TESIS DOCTORAL

Swarm Intelligence in Computer Vision: An Application to Object Tracking.

Luis Antón Canalís

Las Palmas de Gran Canaria - Marzo 2010

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

Instituto Universitario de Sistemas Inteligentes
y Aplicaciones Numéricas en Ingeniería

Para Judith,
por su infinita paciencia para escuchar,
o hacer que escucha,
todo lo que le cuento.
Y por muchísimo más.

Agradecimientos

La principal culpable de que me encuentre escribiendo estas líneas es, sin lugar a dudas, la Dra. Elena Sánchez Nielsen, co-directora de esta tesis. Tras tutorizar mi proyecto de fin de carrera me propuso continuar en la esfera universitaria realizando un doctorado. Entre aquel momento y hoy han pasado seis años y muchas horas en barco.

Si bien no estaba en mis planes, estos cuatro años viviendo en Gran Canaria no han sido tan duros como mis amigos tinerfeños vaticinaban. He de dar las gracias a los miembros del Grupo de Inteligencia Artificial y Sistemas de la Facultad de Informática de la Universidad de Las Palmas de Gran Canaria por conseguir un ambiente de trabajo tan agradable.

Gracias a los doctores Modestro Castrillón Santana, Javier Lorenzo Navarro y Jorge Cabrera Gámez por compartir impresiones profesionales y personales. Sus distintos puntos de vista en cada conversación mantenida han sido muy edificantes.

Quiero dar las gracias especialmente a Francisco Mario Hernández Tejera por dirigir como lo ha hecho todo mi trabajo. Cuando uno pasa con su director de tesis tanto tiempo discutiendo aspectos formales del trabajo como escuchando historias de todo tipo, la mayor parte nada serias, puede estar seguro de que se encuentra ante alguien excepcional. Quiero agradecerle especialmente que me diera libertad para investigar aquello que más me interesara en cada momento,

comprendiendo perfectamente mi no siempre sana necesidad de cambiar de registro de vez en cuando.

Debo agradecer al antiguo Ministerio de Educación y Ciencia la concesión de la ayuda para la Formación de Personal Investigador BES-2005-8272 que he disfrutado al amparo del proyecto TIN2004-07087, financiado por el mismo Ministerio y el Fondo Europeo de Desarrollo Regional (FEDER).

Finalmente, quiero agradecer a mis padres y a mi hermana todo el cariño y el apoyo incondicional recibido desde siempre. Aunque algo así no se agradece, ha de ser devuelto, y soy consciente de que desde lejos no lo hago demasiado bien.

Foto de portada: *Starling flock over Gretna, Scotland*. Photograph by Fi Exon. www.flickr.com/photos/heandfi, www.heandfi.org, www.youtube.com/heandfi

Contents

List of Figures

Resumen

La conocida como Inteligencia de Enjambres aparece bajo diversas formas tanto en la cultura popular (libros, películas y videojuegos) como en entornos de investigación y desarrollo. El concepto, surgido a partir del comportamiento de animales eusociales, propone formas de inteligencia colectiva emergente a partir de la interacción entre entidades relativamente simples. Como grupo, consiguen abordar y resolver problemas a los que, como individuos, difícilmente encontrarían solución.

A pesar de que las soluciones numéricas basadas en Inteligencia de Enjambres fueron propuestas hace más de dos décadas, es ahora cuando empiezan a ser aplicadas en campos diversos. Puesto que los individuos que forman un enjambre realizan su actividad de manera independiente, los enjambres artificiales son altamente paralelizables, lo cual resulta muy adecuado para ser implementado en los actuales ordenadores multi-núcleo.

Resulta igualmente atractiva la relativa simplicidad de cada miembro de un enjambre, ya que su diseño puede resultar más abordable que el de otras metodologías monolíticas y complejas.

El uso de estas técnicas en Visión por Computador parece limitado a su aplicación en problemas de optimización, con algunas excepciones puntuales. Dado que este campo está abierto a la experimentación, gracias en parte a la

inexistencia de un modelo único de visión artificial, la adopción de técnicas basadas en enjambres puede dar lugar a soluciones originales a problemas conocidos.

Este trabajo aborda el problema del seguimiento de objetos en secuencias de vídeo desde la perspectiva ofrecida por la inteligencia de enjambres. Siguiendo como máxima la simplicidad de cada individuo y su independencia frente al resto de componentes del grupo, independencia relativa al procesamiento de la información pero no a su actividad 'social', se proponen dos soluciones basadas en enjambres. Se mostrará cómo el comportamiento que emerge de la interacción entre individuos va más allá de las capacidades individuales.

El presente documento se organiza de la siguiente manera. El primer capítulo ofrece una introducción a la Inteligencia de Enjambres desde los modelos biológicos más conocidos y a la Visión por Computador, así como una breve descripción de las contribuciones de este trabajo. El segundo capítulo revisa la definición de objetos en imágenes digitales y el estado del arte de su seguimiento en secuencias de vídeo. El tercer capítulo aborda los enjambres artificiales inspirados en modelos biológicos, así como su aplicación en Visión por Computador. En el cuarto y quinto capítulo se presentan las aportaciones de este trabajo, dos soluciones de seguimiento inspiradas en enjambres. El sexto capítulo evalúa ambas soluciones aplicándolas sobre distintas secuencias de vídeo. Finalmente se ofrecen las conclusiones alcanzadas y las futuras líneas de investigación.

Abstract

The so-called Swarm Intelligence appears in various forms in popular culture (books, movies and video games), research and development environments. This concept, coined from the observation of the behavior of eusocial animals, proposes forms of collective intelligence emerging from the interaction between relatively simple entities. As a group, they manage to tackle problems that, as individuals, would be difficult to solve.

Although numerical solutions based on Swarm Intelligence were proposed over two decades ago, it was not until recently that different fields begun adopting them. Since individuals belonging to a swarm carry out their work independently, artificial swarms are highly parallelizable, which is very suitable for their implementation in current multi-core computers. The relative simplicity of swarm members is equally attractive, being their design more attainable than other monolithic and complex methodologies.

The use of these techniques in Computer Vision seems limited to their application to optimization problems, with some isolated exceptions. Since this field is open to experimentation, thanks in part to the lack of a unique model of machine vision, the application of techniques based on swarms can lead to novel solutions to known problems.

This work addresses the problem of object tracking in video sequences from

the perspective offered by the intelligence of swarms. The main contributions of this work are two solutions that were designed following the principle of individual simplicity and individual independence between swarm members, independence relative to their information processing activity. The first solution considers a group of free-roaming particles that are guided by simple steering behaviors. In the second solution, particles are linked with distance constraints, shaping an articulated rigid body. In both cases, particles are able to extract information from images and react according to a given visual task like object detection or tracking. It will be shown how the behavior that emerges from the interaction between individuals goes beyond individual capabilities.

This document is organized as follows. The first chapter provides an introduction to Swarm Intelligence and Computer Vision, as well as a brief description of the contributions of this work. The second chapter reviews the definition of objects in digital images and the state of the art in object tracking in video sequences. The third chapter presents biological models and their application to Computer Vision. In the fourth and fifth chapters the two main contributions of this work are presented: two tracking solutions inspired by swarms. In the sixth chapter both solutions are empirically evaluated. The last chapter concludes this work summarizing conclusions and offering future research directions.

Part I

Swarm Intelligence in Computer Vision. An Application to Object Tracking

1 Introduction

FROM observation and imagination comes inspiration. Ever since Da Vinci tried to mimic the wings of bats and large birds in his Flying Machines, and surely even before, engineers, designers, inventors, architects, artists and creative individuals in general have looked into Nature's designs for an answer. Although *Bionics*, the application of biological methods and natural systems to the study and design of engineering systems and modern technology, is a relatively new concept (coined by Jack E. Steele in 1958), it has been constantly present in mankind's technological developments.

The field of Artificial Intelligence, or AI, attempts to understand and build intelligent entities (Russell and Norvig, 2003). Philosophers, psychologists, linguists, mathematicians and computer engineers, among many others, study intelligence and discuss its roots in the struggle to tackle a wide variety of topics: problem solving, reasoning, planning, learning, natural language processing or perception, to name a few. Many tools have been traditionally used in AI to deal with such a number of tasks, from First-Order Logic to Fuzzy Logic, Informed Searches, Optimization, Classifiers, Probabilistic Reasoning and many more.

Naturally, as the ultimate goal of AI consists on replicating Intelligence, there exist many bio-inspired approaches. Probably, the most well known example, and a whole branch within AI, would be the use of artificial neural networks that mimic their biological counterpart. However, not until recently have

AI researchers turned their attention towards social animals in their quest for new approaches to solve old problems more efficiently.

As biologists understand the behavior of social animals and how relatively unintelligent creatures manage to accomplish complex tasks, their findings encourage researchers from many fields to adopt solutions based on what is known as Swarm Intelligence.

1.1 Swarm Intelligence

In our collective minds, the Queen of a swarm still acts as some kind of Overmind that controls the hive. Many science fiction novels depict alien swarm-like creatures governed by a single mind. In Robert A. Heinlein's *Starship Troopers*, the Pseudo-Arachnids were controlled by the Brain Bug (Heinlein, 1987). If it died, the colony died. Similarly, in Orson Scott Card's *Ender's Game*, the Hive Queen controlled the entire Bugger race as a single collective supermind (Scott-Card, 1986). Far from being just fiction tales, these ideas have their origins in the hypotheses of late 19th and early 20th century naturalists.

In 1905, Edmund Selous, ornithologist and confirmed Darwinian, wrote when observing tens of thousands of starlings coming together to roost:

> "*They must think collectively, all at the same time, or at least in streaks or patches a square yard or so of an idea, a flash out of so many brains.*"

The first hypotheses about the mechanisms underlying the collective behavior of insects were clearly anthropomorphic (Buchner, 1881) (Forel, 1921) (Thorpe, 1963). Actually, it seems that the ant or bee queen neither rules the colony or hive or acts as a central peacemaker or coordinator of workers' activities (Reeve and Gamboa, 1983) (Reeve and Gamboa, 1987). Most social animal societies are not hierarchical nor centralized and there exists no supervision, as many works in the last decades reveal (Theraulaz et al., 1998) (Jha et al., 2006).

Figure 1.1: Millions of European starlings migrate through the marshlands of western Denmark, where they appear in mass formations before sunset each day (Earth Science Picture of the Day, author/s unknown.)

More amazing than the hypothesis of an Overmind is the fact that insect colonies are composed of autonomous units acting locally. Following simple probabilistic stimulus-response behaviors (Denebourg et al., 1983), they are unable to assess a global situation and have no knowledge of the global pattern. While this behavior applies to collective decision-making and movement coordination in animal groups like flocks, herds, schools and even human groups (Miller, 2007) (Gordon, 2007) (Zimmer, 2007), it is critical in the development of social units of eusocial animals, usually known as superorganisms (Tautz, 2008).

The key concept behind Swarm Intelligence seems to be Stigmergy, a form of self-organization. This concept was introduced in 1959 by French biologist Pierre-Paul Grass to refer to termite behavior (Grass, 1959). He defined Stigmergy as *"The stimulation of workers by the performance they have achieved."*, after his observations of the coordination and the regulation of building activities in termite colonies. Each action of a termite leaves signs in the environment, signs that termites sense and that determine their subsequent actions. When a termite worker scoops up a mud ball from its environment, it invests the ball with pheromone. This chemical attracts nest mates, so other termites are likely to drop their own mud balls next to previously deposited mud balls. Over time, complex structures

grow and form the nest.

Most social wasps build their nests similarly, using chewed wood pulp and plant fibers, cemented together with oral secretions (Ross and Matthews, 1991). Like termites, the building activity of wasps is also driven by the local nest structure (Karsai and Theraulaz, 1995). Apparently, wasps detect the configuration of nearby nest cells with their antennae, and add new cells with higher probability to corner areas where three adjacent walls are already present (Camazine et al., 2001).

A similar behavior is observed in numerous ant species when carrying their dead out of the nest, which has been studied under controlled conditions with Messor Sanctus ants (Theraulaz et al., 2002) (Jost et al., 2006).

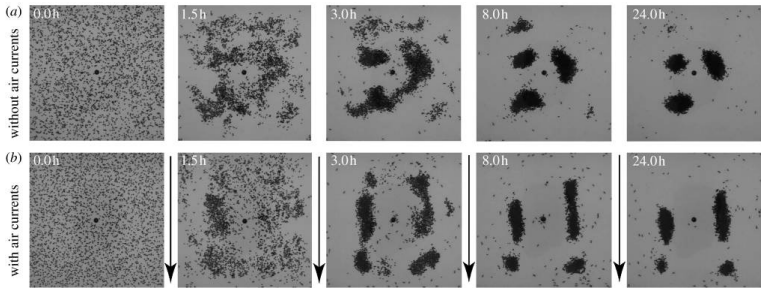


Figure 1.2: Typical spatio-temporal dynamics of corpse clustering by ants filmed from the top through the acrylic glass covering the experimental chamber, without (a) and with (b) air currents. Black dots are ant corpses and black arrows indicate the air flow direction (Jost et al., 2006).

From an initial setting where corpses are spread over the whole surface of an arena, ants are able to collect and aggregate them within a few hours in a low number of piles. An ant will pick up or drop a corpse with a probability that depends on the density of corpses in the vicinity. A high density of corpses will lead to a higher probability of dropping the one the ant is carrying and a lower probability of picking up a corpse, and vice versa. This way, without a central coordination system, ants will slowly pick up corpses and drop them in growing piles (see Figure 1.2)

One of the most classic examples of stigmergy is the way ants create

pheromone trails to guide nest mates (Hölldobler, 1990), and how this process leads to the discovery of the shortest path between two points. When an ant finds a food source, she returns to the nest laying down a pheromone trail. Ants are attracted by this chemical, which will guide them towards the food source. As more ants find the food and return to the nest, laying down more pheromone on the same trail, the signal is reinforced. The more ants use a trail, the more attractive it becomes. But pheromones evaporate with time, so quiet trails will become uninteresting. From all the available paths between two points, the shortest one will be visited by a higher number of ants, as it takes less time to traverse it back and forth. Thus, its pheromone concentration will become stronger than that of longer paths, attracting more ants.

Social animals are able to assess certain group states from local interactions or observations, and adapt their activity accordingly (Fourcassi and Deneubourg, 1994). Locust infer the size of the colony from the number of individuals in their surroundings. They start cannibalizing nearby mates when they feel the local population rises above a certain threshold. As they avoid nearby mates, they align their own movements with any neighbor. This way, they create huge bands that move in the same direction, pushed by fleeing individuals that avoid being eaten by those mates on their back (Bazazi et al., 2008).

For an in-depth introduction to Swarm Intelligence please refer to (Miller, 2007) and (Zimmer, 2007).

1.2 Artificial Vision

Computer Vision is the branch of Artificial Intelligence within Computer Science that is concerned with the theory for building artificial systems that obtain information from images. Using image capturing devices, machines perceive the visual world around them, analyze it and react accordingly. See (Forsyth and Ponce, 2003) for a reference document.

Computers may obtain visual information through a wide variety of visual sensors that are sensitive to certain wavelength ranges within the electromagnetic

spectrum. Depending on the type of sensor, resulting image data may be a bidimensional image or a three-dimensional volume, and it is commonly stored in matrixes which cell values correspond to radiation intensity in a certain spectral band. Light-sensitive cameras, the most commonly used sensors, capture wavelengths within the visible spectrum, ranging from $400nm.$ to $700nm.$ (although some cameras also perceive infrared light, these ranges are usually filtered).

It is a common approach in Computer Vision, due to the nature of visual sensors, to construct a symbolic representation of the real world, or distal space, in an internal representation, or proximal space. This proximal space is usually represented by bidimensional images, projections of the distal space created by light-sensitive cameras. It is in this proximal space where computers analyze and interpret the symbols associated to objects in the distal space, which complexity is otherwise intractable (Edelman, 1999). These images are interpreted with different purposes, like the detection, classification, tracking or segmentation of objects projected from the distal space.

Back in the early years of Computer Vision, in the 1950s and 1960s, it was believed that building perceiving machines would take no more than a decade. After all, seeing is something we accomplish without effort. However, it was soon understood that vision was a task far more complex than imagined. From the physical and biological description of our own visual system to actually understanding what we are seeing, there still exist many unknown processes that involve many different fields like Physics, Neuroscience or Psychology. Still today, Computer Vision is an immature field plagued with specific solutions to concrete problems, as there exists no standard formulation to the computer vision problem. Nevertheless, successful specific solutions may be combined in complex systems in an effort to solve broader problems. There exist commercial solutions for many visual tasks like Image Processing, Object Detection and Recognition, Image Retrieval, Optical Character Recognition or Tracking, to name a few.

While the underlying problem of the current approach to Vision may reside in the Von Neumann architecture of current computers, Computer Vision is still an interesting challenge that deserves attention.

1.3 Contribution

The present work explores the application of Swarm Intelligence to Computer Vision, and proposes two contributions to visual object tracking in video sequences. The first one consists of a Swarm Intelligence metaphor where particles behave like predators hunting a prey, being the prey the object to track. While the patterns of motion remind of a chaotic flying swarm, tracking emerges from the trajectory of the group centroid.

The second contribution considers how to include structural constraints in a group of independent tracking particles, adopting a solution from the video-game field known as Ragdolls, suitable for real time processing. As each particle follows a part of the tracked object, its position is corrected in order to satisfy structural constraints. All together, linked particles create an elastic articulated rigid body that may deform to adapt itself to the underlying object, while still being able to recover its original shape.

1.3.1 Tracking using boids

Boids, by Craig Reynolds (Reynolds, 1987), is a computer model designed to achieve natural coordinated animal motion in computer animations. Boids exists in a virtual environment that simulates an Euclidean three-dimensional or two-dimensional flying space, which may include virtual entities like obstacles, food, hazards, friendly or enemy units or whatever the designer comes up with. Boid groups navigate their virtual world in a manner that resembles real life animal groups, even though their reactions are ruled by simple vector arithmetic.

Boids are able to wander realistically, decelerate when approaching a target, adjust their movement to a path defined by a curve, adjust their orientation to avoid future collisions with static obstacles, steer towards a target or away from it, stick close to local flockmates or imitate their movement, and almost any other behavior that can be defined using an orientation and a velocity vector computed strictly from local information. The combination of these *steering behaviors* creates

complex but natural responses like the ability to follow a leader orderly, traverse a crowded path avoiding collisions or queue to go through a narrow doorway.

While playing with Reynolds's boids, the next questions were raised: what if the space where boids evolve were defined by the image frames that compose a video sequence, and boids were able to interact with them? What if boids could look for a certain feature in the landscape created by an image? Would boids be able to follow an image patch as it traverses the proximal space and, if so, could interacting individual efforts result in an emerging object tracking behavior?

1.3.2 Tracking using Ragdolls

The beauty of boids, the independence of particles and the emerging behavior, is both their strength and their weakness. The relative position of each particle to other particles is irrelevant, the swarm lacks an organized structure.

How could structural constraints be introduced while keeping particle behavior independence?

Chronic Logic's Gish featured in 2005 an improbable hero for a videogame: a ball of tar. Two years later, in 2007, Cryptic Sea acquired the rights to Gish from Chronic Logic and announced the awaited sequel. While it has not yet been released (up to late 2009), their creators offered a teaser trailer of the new Gish three weeks into development, focusing on the new physical structure of the main character. The new ball of tar was composed by more than a hundred particles joined by elastic links, creating a viscous slime that was able to stretch or shrink, breaking into smaller slimes when links were broken. It made a very good impression.

But, moreover, it was the answer to how independent tracking particles could be linked together effortlessly, computationally speaking, to create a single shape-preserving elastic entity.

Even before *Gish 2*, many award-winning videogames have used similar physic engines. But it was Thomas Jakobsen, working for IO Interactive's

Hitman: Codename 47, who first proposed a comprehensive, fast and stable physics system, commonly known as *Ragdoll physics* because human-like structures governed by these systems behave like stringless marionettes (Jakobsen, 2001).

The second contribution of the present work consists on the application of Ragdoll physics to Computer Vision tasks, modeling articulated bodies as particles with constraints, using a Verlet integration scheme (Verlet, 1967) to control particle dynamics and solving constraints using relaxation. Thanks to the relative simplicity of Ragdoll physics, elastic structures can be integrated into complex Vision tasks like clustering, articulated body pose analysis or tracking.

2 Object Tracking

OBJECT tracking, the action of following visual entities autonomously as they traverse a video sequence, is a fundamental problem in computer vision (Yilmaz et al., 2006). While tracking under controlled or known environments can be considered a solved problem, a general tracking solution still remains a challenging task. Object tracking is thus a widely researched topic in computer vision.

Tracking is basically a low level, pre-categorical task: objects may be followed no matter their class. Babies are able to follow colorful toys even before they learn what they are, relying on low level cues like color, textures and movement. Once an object is learnt, the tracking activity may make good use of known information from the target like movement behavior or possible appearances to improve robustness. However, this work will avoid the loaded nature of visual objects. No specific object classes are considered and thus no class-related information will be used.

2.1 Object definitions in a 2D space

There exists no single definition of what an object is in the proximal space (the space that stores a projection of the real world, or distal space, created by a visual sensor) (Edelman, 1999). If any, it would be anything that is of interest for the

task at hand.

Objects are characterized by their appearance, using features computed from images to describe their visual aspect, but also by their geometry, motion and possible relations with other objects, which defines their shape and degree of non-rigidity. How objects are represented determines how to work with them, and the nature of the solution to a visual problem is tightly related to the chosen object definition.

2.1.1 Object representations

Different representations are adopted depending on the degree of non-rigidity of an object and whether it is defined by a single or multiple parts. While rigid objects require simpler descriptions, non-rigid motion is exhibited by most real world objects.

Non-rigid motion is generally classified in three groups: the motion of rigid parts, where parts of an object move independently one of another (categorized as articulated motion), motion of coherent objects and motion of fluids. In computer simulations, coherent motion is usually estimated through discrete articulated rigid bodies composed by multiple infinitesimal parts. An object classification scheme based on non-rigidity degrees is proposed in (Kambhamettu et al., 1994). The following definitions are given:

- Rigid motion: all distances and angles are preserved and it has no associated non-rigidity.
- Articulated motion: rigid parts conform to rigid motion constraints, but overall motion is not rigid.
- Quasi-rigid motion: deformations are restricted to be small. Any general motion is quasi-rigid when viewed in a sufficiently short time window.
- Isometric motion: is defined as motion that preserves distances along the surface and angles between surface curves.

- Homothetic motion: motion with a uniform expansion or contraction of the surface.
- Conformal motion: non-rigid motion which preserves the angles between surface curves, but not distances.
- Elastic motion: non-rigid motion whose only constraints is some degree of continuity or smoothness.
- Fluid motion: violates the continuity assumption. It may involve topological variations and turbulent deformations.

Knowing how an object evolves in a video sequence, whether it is rigid or elastic, dictates how it may be described. The simplest representation of a visual object in a bidimensional space is a single point, which may coincide with the object's centroid. Simple geometric shapes (i.e. rectangles or ellipses), can be used to define image regions around specific points, which may enclose the whole object or represent distinct parts at different scales. Affine transformations can be considered in order to correctly adapt these shapes to rotations, shearing, translations and scaling. A collection of points and/or regions may be combined to define compound objects. In order to deal with deformable targets, many works consider silhouettes, contours and skeletons due to their versatility to describe shapes. Figure 2.1 shows some possible object representations.

Once the shape or structure of an object is defined, it must be described using information extracted from image values.

2.1.2 Object descriptors

Digital images are composed by thousands of values, called pixels, stored in a bidimensional grid that represents radiation intensities captured by a sensor in a certain spectral band. Common photographic digital cameras use a CMOS or CCD image sensor that operates with some variation of the RGB model, capturing wavelengths from the three additive primary colors: red, green, and blue. Thus,

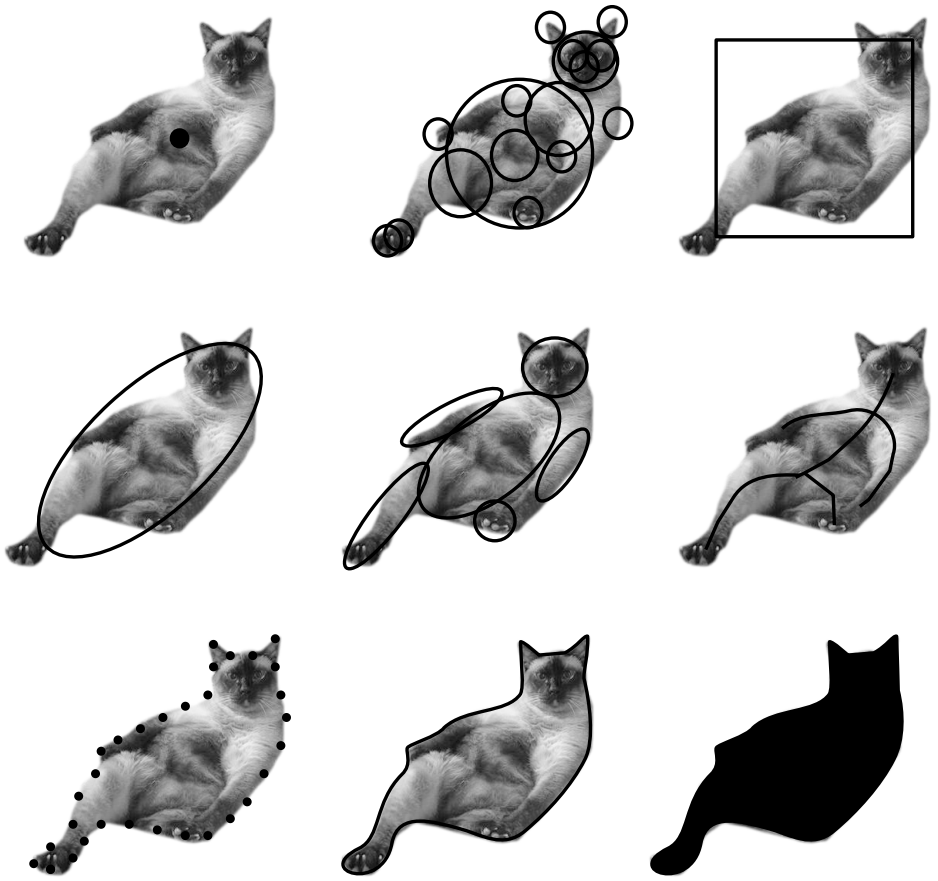
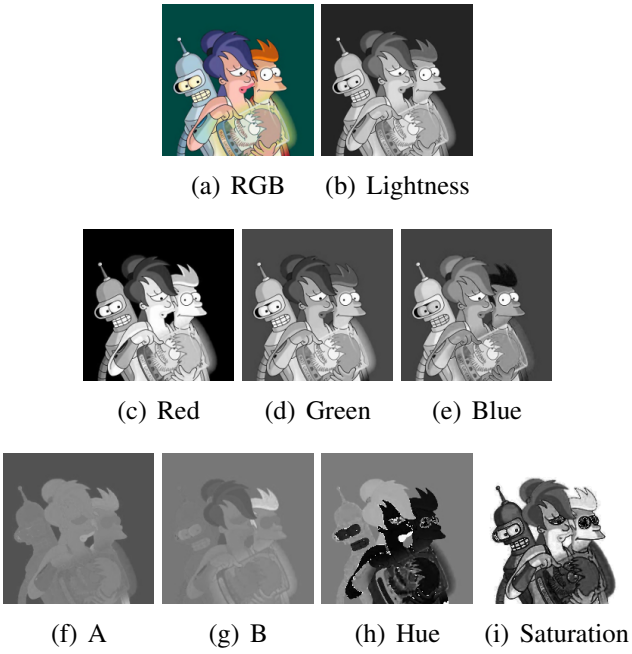


Figure 2.1: Bidimensional objects may be represented in many ways: by a centroid, a collection of interest points (with or without scale or affine transformations), rectangles, fitted ellipses, a collection of parts, skeletons, points in their contour, complete contours, silhouettes...

individual pixels just contain color information projected from the distal space. In order to define a low level descriptor of an object in a digital image, relations between neighboring pixels must be described in terms of spatial and frequential measurements.

Color spaces

Light-sensitive cameras generally use the RGB aditive color model, where the three primary additive colors red, green and blue are added to create a wide range of pigments. However, the RGB color space is not perceptually uniform. A system is perceptually uniform if a small perturbation to a component value is approximately equally perceptible across the range of that value. In the RGB color space, distances between colors that are visually different may be numerically similar, and vice versa.



(a) RGB (b) Lightness

(c) Red (d) Green (e) Blue

(f) A (g) B (h) Hue (i) Saturation

Figure 2.2: From a given RGB image other color spaces can be computed, like HSL (Hue, Saturation and Lightness) or CIELab (Lightness, a, b). Channels are shown independently. Their values were normalized and remapped to grayscale for visualization purposes.

Many RGB non-linear transformations have been proposed in an attempt to obtain a reasonable perceptually-uniform color space. The following examples are color spaces that create independent channels for chrominance and luminance information:

- Normalized Color Coordinates: also known as red-green normalized, this color space only represents chromatic information. It has provided good results even under wide lighting variations (Fritsch et al., 2002) (Schmidt and Castrillon, 2008).

- HSL: it stands for Hue (color or, according to (Smith and Guild, 1931): *the attribute of a visual sensation according to which an area appears to be similar to one of the perceived colors, red, yellow, green and blue, or a combination of two of them*), Saturation (*colorfulness of an area judged in proportion to its brightness* (Smith and Guild, 1931)) and Luminance (*perceptual response to lightness* (Smith and Guild, 1931)). Hue and Saturation channels are independent from Luminance, so chromacity may be defined using just Hue and Saturation.

  However, this color space suffers two main problems: Hue values are defined by angles in a circle, but 0° and 359° both correspond to the same red color; and they are unreliable in low Luminance or Saturation regions, showing noisy values. Both problems make computing distances between two colors in this space more difficult: while Hue distances can be computed from the length of the chord between two Hue angles, noisy Hue values are hard to deal with, requiring the definition of dark and low saturation levels. Alternatively, an HSL variant where Hue is non cyclic can be used, requiring a conversion of the Hue value from polar to Cartesian coordinates (Mundhenk et al., 2005).

- CIE L*a*b and L*u*v: two CIE (International Commission on Illumination) standardized systems that, according to (Smith and Guild, 1931), *"improve the 80:1 or so perceptual nonuniformity of XYZ to about 6:1"*. Luminance (L channel) represents lightness, while a*b* and u*v channels store chromatic information. This color space demands more computation than the previous two, but it is more descriptive than Normalized Color Coordinates and color distances can be easily computed using the Euclidean Distance. CIELuv, however, seems to behave much better than CIELab in darker regions.

Figure 2.2 shows an RGB image decomposed in most previously mentioned channels.

Gradients and edges

Proximal object boundaries usually generate strong changes between corresponding neighboring pixel values. Because images can be considered two-variable functions, change strength can be obtained computing partial derivatives in the horizontal and vertical directions. Operators based on the first derivative of an image, its gradient, include Roberts, Prewitt, Sobel and Frei-Chen. (Forsyth and Ponce, 2003)

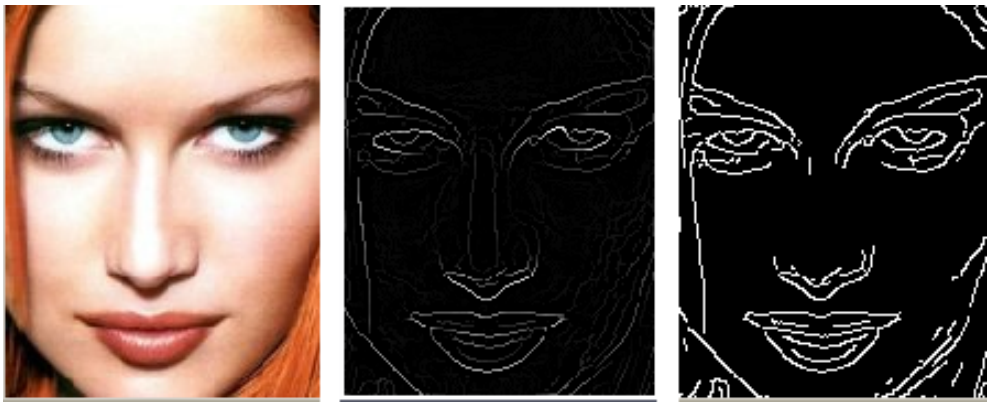


Figure 2.3: From the left-most RGB image, a Sobel operator computes vertical and horizontal gradients, from which the gradient magnitude image shown in the middle is created. Binary edges in the right-most image are computed suppressing non-maxima values from the gradient image.

Edges are binary maps where active pixels represent boundaries. Edge points of an image can be detected by finding the zero crossings of the second derivative of Gaussian filtered image intensities (Laplacian of Gaussian). Classic edge detectors like Canny (Canny, 1986) and Deriche (Deriche, 1987) rely on non-maxima suppression from image gradient values followed by an hysteresis process to find representative edges. While both methods return complete edges (i.e. showing no gaps), they are not very stable in video sequences if parameters are not properly set (Antón-Canalís et al., 2006a). Edges can also be computed

at different scales, adding robustness against noise, as proposed by (Perona and Malik, 1990). For an in-depth review of edge detection see (Ziou and Tabbone, 1998).

Textures

A texture is created by the regular repetition of pixel values on an image region, quantifying properties such as smoothness and regularity. Compared to color and edges, textures require a more expensive preprocessing step to generate appropriate descriptors. Textures may contain color information, buy they are less sensitive to changes in illumination than color values because they also contain structural information.

The list of available texture descriptors is long (Tuceryan and Jain, 1993) (Puig and Garcia, 2006). The most frequent techniques include Gabor filters (Fogel and Sagi, 1989), steerable pyramids (Greenspan et al., 1994) and Local Binary Patterns (LBP) (Pietikäinen, 2005).

Local Binary Patterns (LBP) are image descriptors commonly used for classification and retrieval. Introduced by Ojala et al. (Ojala et al., 2002) for texture classification, it is characterized by its invariance to monotonic changes in illumination and low processing cost, which makes it suitable for real-time processing.

Given a pixel, the LBP operator first obtains a binary pattern comparing the gray value of the pixel with those of its neighbors in a circular neighborhood. Then the pixel is assigned a value computed from the resulting pattern according to a weighting mask, as shown in Equation 2.1. A region is described from the concatenation of local LBP histograms. Rotation invariance is achieved considering circular local binary patterns.

$$LBP_{P,R}(x_c, y_c) = \sum_{k=0}^{P-1} s(g_p - g_c)2^k \ , \ s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{2.1}$$

LBP can also be used as a preprocessing method, having the effect of

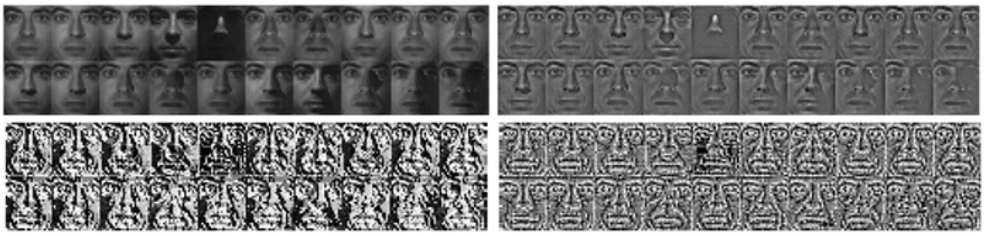


Figure 2.4: Examples in Yale database B: left above - the original face images [2]; right above - preprocessed by linear high-pass filter, left below - preprocessed by the LBP with 256 patterns [15]; right below - preprocessed by the simplified LBP. Dynamic ranges are edited for visualization purposes.

emphasizing edges and noise. To reduce noise influence, a modification to Equation 2.1 has been proposed (Tao and Veldhuis, 2007). Instead of assigning each neighbor a different weight, they are equally weighted, so each pixel has a maximum of nine different labels (in a $3x3$ neighborhood). This approach has shown some benefits when applied to facial verification, due to the fact that images become more robust to illumination changes. See Figure 2.4 for an example of LBP used as a preprocessing step.

Discriminative Features

The discriminative power of a region can be exploited between the region and its local background. Psychology research results (Shim and Cavanagh, 2004) suggest that a human vision system does shift its attention among different local regions during the observation of an object. Which features should be used to describe an object seems to depend on the context in which the object exists, and they should change if context changes (Chen and Yang, 2005).

In (Collins and Liu, 2003) a relatively simple on-line feature detector is proposed: object are compared with their surroundings performing center-surround histogram analysis. The set of candidate features is composed of linear combinations of RGB pixel values (see Equation 2.2).

$$F \equiv w_1 \cdot R + w_2 \cdot G + w_3 \cdot B, w_* \in [-2; -1; 0; 1; 2] \tag{2.2}$$

where weights are integer coefficients between -2 and 2. Pruning linear combinations and all-zero weights, 49 features are left. Comparing center-surround histograms for each feature, the most discriminative one is chosen as the current object feature. While it was only applied to color channels, the same principle may be used for other features like textures or gradient. See Figure 2.5 for an example where the most discriminative features for two different objects in the same frame of a video sequence are shown.

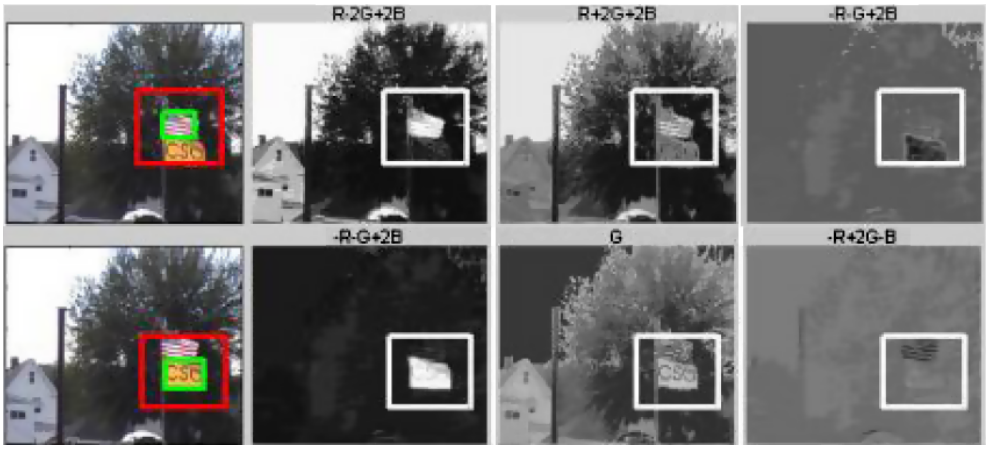


Figure 2.5: Left column: Frame with labeled object (green box) and background pixels (red box). Second through fourth columns: weight images corresponding to tuned features with highest score, median, and lowest variance ratio scores, respectively. ((Collins and Liu, 2003))

2.1.3 Object detection

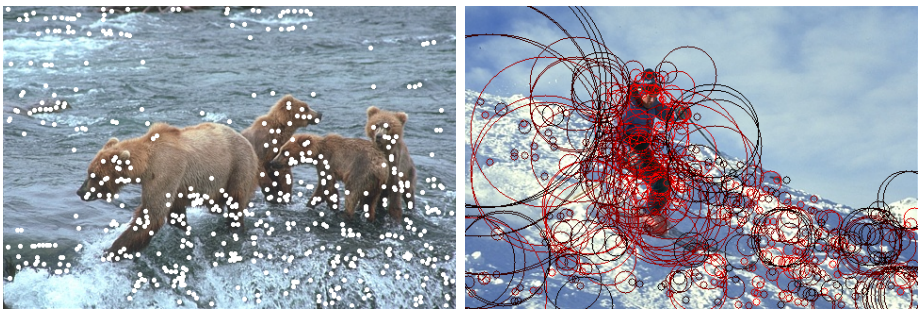
Object trackers need a target to follow. While the target can be defined manually, it can also be detected automatically by a variety of means. Object detection is an important branch within Computer Vision, and it refers to the process of detecting visual objects in images. Detected objects may belong to a previously known class, like pedestrians, cars or faces, although pre-categorical bottom-up object detectors are able to find interesting image regions that may deserve further processing.

Interest Point detection

While an image contains thousands of pixels, most of them are not significative per se. Interest point detectors are bottom-up processes that find those points in an image which local image structure contains rich information (See Figure 2.6)

Since Moravec's interest operator (Moravec, 1979), which detected corner-like structures, point detectors have been in constant evolution. Harris corner detector (Harris and Stephens, 1988) improved Moravec's detector, and the Shi and Tomasi detector (also known as Kanade-Lucas-Tomasi or KLT) (Shi and Tomasi, 1994) is strongly based on Harris' detectors. Lindenberg introduced the scale-space representation and automatic scale selection (Lindeberg, 1993) (Lindeberg, 1998), using the determinant and the trace of the Hessian matrix to detect center-surround (blob) structures with their own characteristic scale. Mikolajczyk and Schmid (Mikolajczyk and Schmid, 2001) proposed a multi-scale Harri's detector, adding scale invariance and higher repeatability, and Lowe (Lowe, 1999) proposed a fast approximation to the Laplacian of Gaussians (LoG) by a Difference of Gaussians (DoG) filter.

While many different detectors have been proposed (Mikolajczyk et al., 2005) (Mikolajczyk and Schmid, 2005), recent approaches still seem to outperform previous schemes, like the SURF detector (Bay et al., 2006), in terms of invariance, repeatability, stability, accuracy and speed of execution.



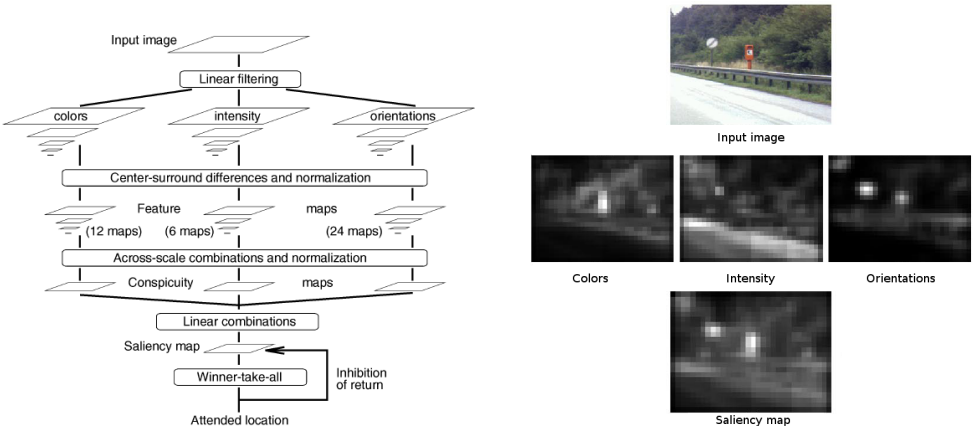
(a) (b)

Figure 2.6: a) Kanade-Tomasi salient points are marked with white dots. b) Circles are drawn around center-surround salient points, where circle radii represent scale.

Saliency-Based visual attention

Similarly to salient point location, saliency-based visual attention methods compute saliency maps mimicking how biological attention architectures supposedly work. According to (Itti et al., 1998), *"In order to interpret complex scenes in real time, intermediate and higher visual processes appear to select a subset of the available sensory information before further processing. This selection appears to be implemented in the form of a spatially circumscribed region of the visual field, the so-called focus of attention, which scans the scene both in a rapid, bottom-up, saliency-driven, and task-independent manner as well as in a slower, top-down, volition-controlled, and task-dependent manner."*.

Approaches based on saliency decompose images into different feature maps (e.g. color, intensity, orientations, textures...) at different scales (see Figure 2.7). Each map is processed independently looking for salient features, which are finally combined to create a saliency map (Park et al., 2002) (Liu et al., 2007). These saliency maps can be scanned orderly starting from the most salient regions, in a way that seems to imitate human visual attention.



(a) (b)

Figure 2.7: a) Saliency-based visual attention model proposed in (Itti et al., 1998). b) Saliency map computed from an input image.

Background subtraction

In many contexts, video cameras are fixed or their motion is limited. Under these assumptions, the background scene does not change significantly or shows some regular behavior that can be modeled. If the system knows how a scene looks like, new objects can be detected as those regions of the scene not belonging to the background model.

A complete visual comparison of early background subtraction methods like Eigenbackgrounds, Normalized Block Correlation, Wallflower or Mixture of Gaussians is presented in (Toyama et al., 1999). Some methods have been improved, like a new Adaptive Gaussian Mixture Model (Zivkovic, 2004) and new methods have been proposed, like texture-based LBP (Heikkila, 2006).

Segmentation

Image segmentation consists on dividing an image into significative partitions. Segmentation, however, is a task that is highly dependent of higher level processes, and different segmentations may arise depending on prior knowledge and clustering conditions. In (Bagon et al., 2008), for example, visually meaningful image segments are found from a given single point-of-interest, specified by the user.

Bottom-up segmentations must rely in lower level features like image gradients, generally used in watershed-based segmentations (Meyer and Beucher, 1990), color and textures (Chen et al., 2003). Robust segmentations can be obtained using clustering techniques based on Mean Shift (Comaniciu and Meer, 2002), Normalized Graph Cuts (Shi and Malik, 2000) and even combining methods (Tao et al., 2007), although they are computationally expensive.

Segmentation can be used to reduce image information, grouping pixels in bigger structures known as *superpixels* (Ren and Malik, 2003). Pixels are not natural entities, they are merely a consequence of the discrete representation of images. When grouped together in local and coherent structures, most information is preserved. These structures may be used to guide higher level segmentations

(a) (b)

Figure 2.8: a) Clustered pixels creating *superpixels*. b) Image synthesized using its distance transform maxima.

(Mori et al., 2004) and model searches (Mori, 2005). A fast approach that only considers color cues, suitable for real-time applications, was presented in (Antón-Canalís et al., 2007). In that work, superpixels were created from maxima in distance transform images. Two examples are shown in Figure 2.8.

Supervised Learning

Interest point detectors, saliency-based methods, background subtraction and segmentation are generally pre-categorical detectors, unless some top-down class-specific method drives the process (Borenstein and Ullman, 2002) (Borenstein and Ullman, 2004). Categorical object detectors are able to find instances of previously learned object classes. These detectors commonly require a set of positive and negative samples in order to find discriminative features from the class they are being trained to detect.

Most categorical object detectors compute feature vectors from visual objects and apply machine learning techniques (Duda et al., 2001) like Support Vector Machines, Adaptive Boosting, Decision Trees or Principal Component Analysis, to obtain suitable classifications.

In (Leibe et al., 2008), objects are described using collections of appearances (codebooks) in structures that include both representative image

patches from the most salient object parts and a model of their possible positions in relation to the object centroid. That way, wherever a known patch is found, it votes for object centroid location hypothesis. Centroid locations with enough votes are verified looking for the presence of all those patches that should have voted for them. This system not only detects but also segments instances of known objects (see Figure 2.9).

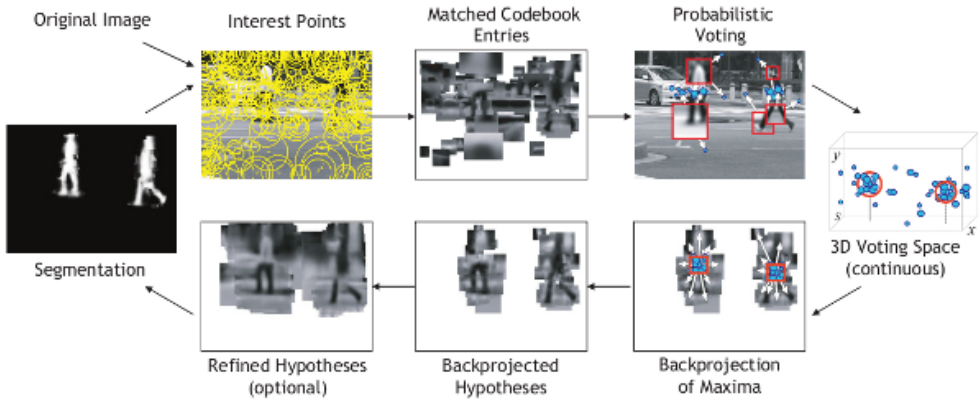


Figure 2.9: Interleaved object categorization and segmentation.

In (Shechtman and Irani, 2007), the internal layout of local self-similarities is computed from a single training image, which is used to describe a whole object class. Self-similarity is a powerful mechanism that allows describing objects from their shape, overcoming color and textures. Even a sketched figure may lead to the detection of objects with the same shape (see Figure 2.10).

Even when a reliable class detector is available and a certain object class is robustly detected, object tracking may still be necessary to maintain temporal coherence and follow object instances as they traverse video sequences.

2.2 Object tracking survey

Object tracking consists on consistently following the position of visual object instances over time. Tracking can be performed in collaboration with an object detector algorithm, defining correspondences between object instances

Figure 2.10: Matching local self-similarities across images and videos.

across frames. But also autonomously, iteratively updating object location and appearance from previous frames, given an initial object instance.

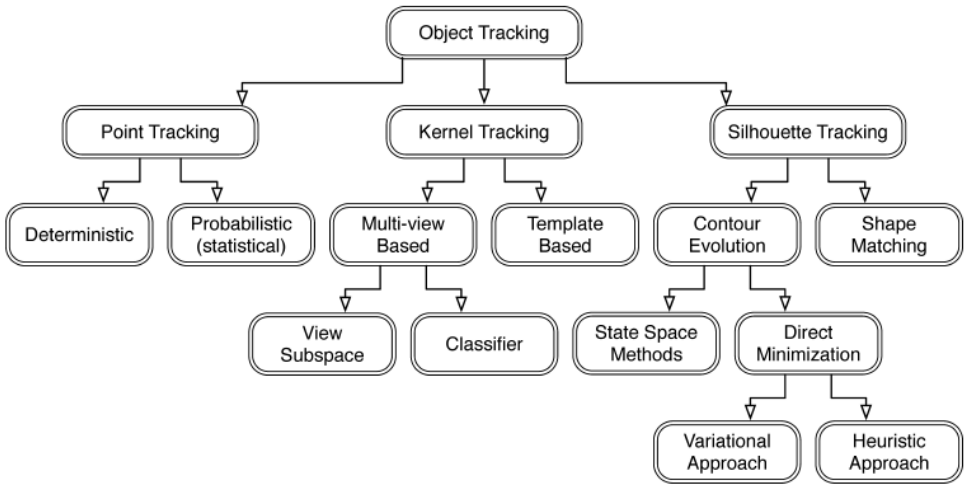


Figure 2.11: Taxonomy of tracking methods, proposed in (Yilmaz et al., 2006).

A taxonomy of tracking methods according to object representations is offered in (Yilmaz et al., 2006). Tracking approaches are grouped in three big groups: Point tracking, Kernel tracking and Silhouette tracking, as seen in Figure 2.11. However, most proposed solutions in the literature consist of a mixture of methods in order to increase robustness.

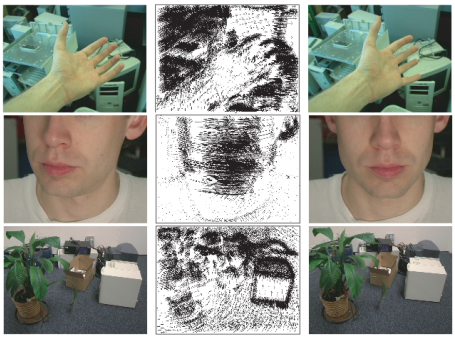
2.2.1 Point tracking

Points are detected by a variety of means, as explained in Section 2.1.3, and tracked using deterministic or statistical methods that find point correspondences between consecutive frames.

Deterministic methods define point correspondence cost functions that are minimized to find one-to-one assignments. Trajectories are generated by a combination of constraints like proximity, maximum velocity, small velocity changes, common motion, rigidity or proximal uniformity. Two points in consecutive frames are considered the same one if their appearance vector is similar and the trajectory that leads from one to the other satisfies imposed constraints. See (Yilmaz et al., 2006) for a review of several methods.

In (Sand and Teller, 2006), long-range point trajectories are computed using pixel appearance and optical flow to estimate dense particle motion. This method has been successfully applied to video edition in (Goldman et al., 2008). However, although its results are impressive, two passes through the whole video sequence are required, not being suitable for real time applications (see Figure 2.12).

Statistical approaches model object movement taking into account noisy measurements and random perturbations, using predictive filters to estimate object positions. Correspondences are defined statistically from existing points towards new object detections, defining relationships between measurements and object states. The Kalman filter (Kalman, 1960) is a recursive solution to the discrete-data linear filtering problem that has been extensively used in object tracking. In order to use the Kalman filter to estimate the internal state of a process given only a sequence of noisy observations, the underlying linear dynamical system, process noise and measurement noise must be properly modeled. This requisites are suitable for object tracking, because many common object dynamics can be described using Euler's movement equations. The Extended Kalman Filter performs recursive non-linear estimations, but the state distribution is still approximated by a Gaussian random variable, which is not optimal if multiple detection hypotheses are considered. Kalman filters usually fail if the tracked object experiences sudden velocity changes.



(a)

(b) (c)

Figure 2.12: a) Optical flow, b) Dense particle maps, c) Video labeling from long-range point trajectories.

The robustness of Kalman filters is improved by the ability of Particle Filters (Maskell and Gordon, 2002) (Arnaud et al., 2008) to deal with non-linearities and non-Gaussian distributions. Instead of a using a single Gaussian function, states are represented by a set of samples, or *particles*, that shapes the probability density function (PDF) of the system.

2.2.2 Kernel tracking

Objects defined by a primitive geometric shape like a rectangle or an ellipse may be tracked using an iterative localization procedure based on the optimization of a similarity measure, a method known as Pattern Matching. Because this process is computationally expensive, it is usually limited to a search window around the last known position.

There exists a large variety of pattern descriptors used for tracking. Feature statistics, like color mean within the defining shape (Fieguth and Terzopoulos, 1997), are probably the simplest descriptors, but they are not very discriminative. Image patches can be directly used as template windows of pixel intensities or color values in any color space, explicitly capturing spatial and appearance information (Shi and Tomasi, 1994) (Edelman, 1999) (Guerra, 2002). However, they are not robust against deformations and they only represent a single object view. Color histograms, on the other hand, completely discard spatial information, being much more robust against local deformations. They have been used for tracking in many approaches (Comaniciu et al., 2003) (Collins, 2003) (Zivkovic, 2004) (Hager et al., 2004). Spatiograms (Birchfield and Rangarajan, 2005) offer a solution somehow between templates and histograms, adding spatial information to histograms by incorporating spatial means and covariances to each color histogram bin. A similar solution is offered in (Zhao and Tao, 2009), using simplified color correlograms. A combination of color histograms, correlograms, LBP textures and motion is used in (Takala and Pietikainen, 2007) to track multiple targets.

Other common region descriptors include Cross Correlation, Moment Invariants, Steerable Filters, Differential Invariants, Complex Filters and Scale Invariant Feature Transform (SIFT) descriptors. An analysis of these methods in (Mikolajczyk et al., 2005) concludes that SIFT (Lowe, 1999), which describes local features using sets of orientation histograms, outperforms all other descriptors. However, SIFT is still computationally expensive and can only be applied to real time tracking of previously detected interest points (Zhou et al., 2008). DAISY, a promising fast local descriptor based of SIFT and GLOH, is proposed in (Tola et al., 2008). It is apparently much more robust than previous methods, and it can be computed much more efficiently for dense matching purposes.

An object can be defined by multiple regions. In (Adam et al., 2006), several rectangular sections are described by histograms and combined to track objects under severe partial occlusions. In (Ren and Malik, 2007) objects are defined by an ensemble of triangular regions computed from a Delaunay triangulation, and

tracking is performed in a figure/ground segmentation-tracking cycle.

Figure 2.13: Mean shift tracking, proposed in (Collins, 2003).

Predictive filters are not exclusively used for point tracking. In (Yang et al., 2005), a particle filter tracks objects characterized using color and edge orientation histogram features. Particle filters are especially handy when dealing with multi-modal probability density functions. However, many works use a common mode-seeking algorithm known as Mean Shift, which has achieved considerable success in object tracking due to its simplicity and robustness (see Figure 2.13). It is a gradient climbing, non-parametric technique first proposed in (Fukunaga and Hostetler, 1975) (Cheng, 1995) that finds the peak of a given distribution. Camshift (Bradski, 1998) was proposed as a mean shift modification that adapts dynamically to the probability distribution it is tracking, dealing with appearance and scale changes. Current trends indicate a shift towards Particle Swarm Optimization as a fast and robust mode seeking process (Thomas and Kambhamettu, 2006). It will be explained in detail later.

2.2.3 Silhouette, Contour tracking and Flexible models

Silhouettes of objects and their contours are powerful clues that allow the sequential segmentation and tracking of deforming objects or regions of interest (ROI). Deformable objects like human bodies or hands, or objects suffering shape changes due to rotations, like a turning car, show highly variable shapes that can be defined more robustly using their contour or silhouette than using a single rigid pattern.

Silhouette tracking is similar to template matching. Known shapes are matched against an image, looking for the extreme value of a similarity function. The Hausdorff distance (Huttenlocher et al., 1993), the Chamfer distance

(Borgefors, 1988) or Shape context (Belongie et al., 2002) are used to define distances between point sets that allow constructing correlation surfaces from which an extremum is selected as the new object position and shape. Based on the Kalman Filter, the Condensation algorithm (Isard and A., 1998) tracks outlines and features of foreground objects defined by B-splines, allowing contour deformation up to a certain degree.

Figure 2.14: Object contour tracking using level sets, proposed in (Yilmaz et al., 2004).

Contour tracking methods follow object boundaries as they translate and deform between frames. Active Contour Models, known as Snakes, were introduced in (Kass et al., 1988). Contours are defined by splines that continuously evolve to adapt to object boundaries. Similar solutions are applied to segmentation and tracking, like the Level Sets approach (Yilmaz et al., 2004), shown in Figure 2.14. Contour control points may also include appearance information as proposed in the Active Shape Models (Cootes et al., 1995) (Davies et al., 2001) and Active Appearance Models (Cootes et al., 1998).

3

Engineered Swarms

THE ability of swarming creatures to achieve complex tasks no matter their own individual simplicity has already inspired many swarm-based solutions in different fields. Swarm intelligence, an expression that was introduced by Gerardo Beni and Jing Wang in 1989 (Beni, 1989) in the context of cellular robotic systems, is currently considered a type of artificial intelligence based on the collective behavior of decentralized, self-organized systems. Swarm Intelligence may be represented by the concept of *Synergy*, meaning *"the interaction or cooperation of two or more organizations, substances, or other agents to produce a combined effect greater than the sum of their separate effects"*. Or, as a famous quotation attributed to Aristotle states, *The whole is greater than the sum of the parts* (von Bertalanffy, 1975).

But before the ability of insect colonies to solve optimization problems was modeled and applied to solve numerical problems (Dorigo and Stützle, 2004), many systems already existed showing self-organizing, evolving and emerging behaviors.

3.1 Artificial Life

> *A theory has only the alternative of being right or wrong. A model has a third possibility; it may be right, but irrelevant.*
>
> Manfred Egan

A computational model attempts to simulate a particular system described mathematically and/or algorithmically. Since the late 1940's, when John von Neumann himself postulated the potentials of Artificial Life, many systems have simulated the behaviour of living creatures in their ecosystems. One of the first examples of a system with surprising emergent properties is Conway's *Game of Life*, a cellular automaton created by John Horton Conway in 1970 that successfully applies von Neumann's ideas in a simplified setting.

Conway's *Game of Life* is the oldest and simplest example of a complex environment showing emergence, and it can be considered the precursor of bio-inspired computational models. An infinite bidimensional matrix is defined, where each cell has two possible states, live or dead. In each simulation step, every cell interacts with its eight neighbors, and its own state is defined by the perceived number of living neighbors. Underpopulation and overcrowding kills a cell, but they revive if a certain number of neighbors are alive. Even from a random initial state, where each cell is set to live or dead randomly, different stable cell configurations arise after a few iterations. As they stay still, oscillate or even translate, existing configurations may interact and evolve, being destroyed or creating new configurations. Some structures are shown in Figure 3.1

Artificial Life (alife) studies the simulation of any aspect of life, as through computers (soft alife), robotics (hard alife), or biochemistry (wet alife). Since the beginnings of computation, the possibility of creating artificial life with a computer program has been a daunting challenge. May a *living* creature be described with some thousand lines of code? The Birth-Growth-Reproduction-Death cycle has been simulated in virtual creatures that wander in virtual worlds, interacting with their environment and other creatures, evolving and successfully

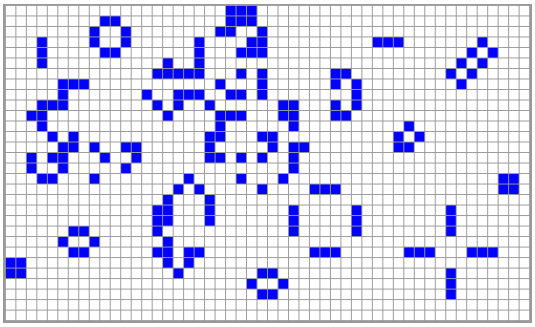


Figure 3.1: Conway's Game of Life

representing simple ecosystems. But whether these creatures are alive or not still generates debate.

Alife simulates life at different scales. Some systems study ecosystems as a whole, simplifying creature movement mechanics in order to analyze the phenomena of living systems. Tierra (Ray, 1991), Cosmos (Taylor, 1997) and Avida (Ofria and Wilke, 2004) are examples of systems designed to analyze biologic processes, interactions between synthetic creatures and their evolution.

Other simulations focus on the complex morphology of single virtual creatures. In (Sims, 1994), DNA-like codes were used to create a genotype of joints, limbs and actuators, producing different locomotive abilities. A population of several hundred of virtual creatures was created within a supercomputer. Each creature was tested for its ability to perform a given task like swimming in a simulated water environment: those that were most successful survived and reproduced, adding their virtual genes to the new population. Not surprisingly, after some time, creatures with successful behaviors emerged. Some even showed a strikingly resemblance to real life animals, both in their morphology and their movements, like the snake-like one shown in Figure 3.2.

3.1.1 Steering Behaviors: Boids

Alife studies complete synthetic ecosystems and organisms, evolution and life processes. As previously stated, life is simulated at different scales and

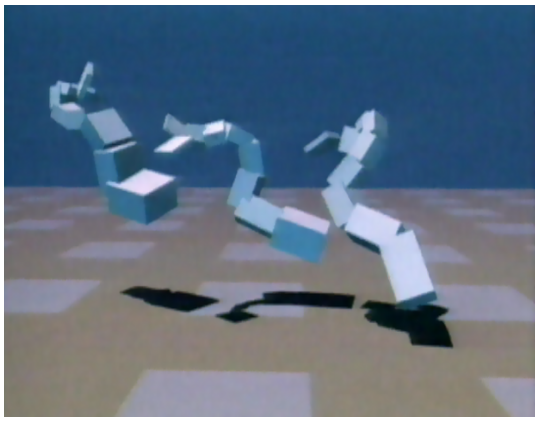
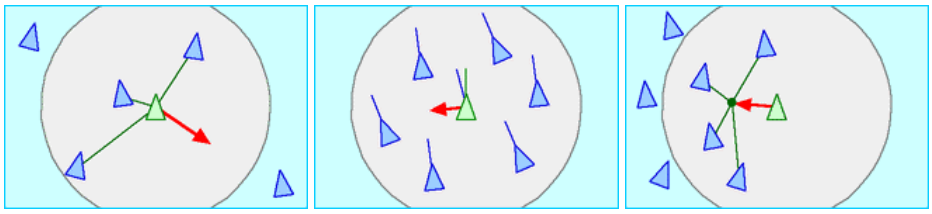


Figure 3.2: Evolved digital creature (Sims, 1994).

complexity levels. While simulations may focus on low level features like the neural networks that govern virtual creatures or the DNA-like structures which alteration defines new phenotypes, life-like behaviors can be obtained in a true swarm intelligence manner from the unexpected interactions of multiple individuals following simple rules.

Back in 1986, Craig Reynolds proposed a computer model of coordinated group motion largely independent of the particulars of the individual's means of locomotion (Reynolds, 1987). He called his virtual creatures *Boids*, and their basic flocking abilities where defined by just three rules, named Cohesion, Alignment and Separation, using vector arithmetic. Thanks to these three rules, each boid was able to maneuver according to the positions and velocities of nearby flockmates (see Figure 3.3). Boid movement rules were latter supported by experiments in real fishes (Partridge and Pitcher, 1980).

Movement rules simply define velocity vectors. The cohesion rule computes a vector from the position of a boid towards the perceived centroid of nearby swarm mates (that is, the average position of all neighboring boids). The alignment rule is the average of neighboring boids' velocities. The separation rule is a vector from the centroid of all flockmates moving at close range towards the position of the boid. In order to obtain a natural a flocking behavior, these three rules are weighted and added to a fraction of the current boid's velocity, and then its position is updated. Obstacle avoidance can be included in this scheme



(a) (b) (c)

Figure 3.3: Reynolds's boids follow three basic steering behaviors, which lead to realistic flocking movement. a) Separation: steer to avoid crowding local flockmates. b) Alignment: steer towards the average heading of local flockmates. c) Cohesion: steer to move towards the average position of local flockmates.

by simply adding a fourth rule computing a vector that moves a boid away from nearby obstacles.

The following expressions define how a given boid p would move in a system with n rules:

$$\begin{aligned} \vec{v_p}(t) &= \omega_0 \cdot \vec{v_p}(t-1) + \\ &\quad \omega_1 \cdot (\vec{rule}_1) + \\ &\quad \omega_2 \cdot (\vec{rule}_2) + \\ &\quad \omega_3 \cdot (\vec{rule}_3) + \\ &\quad \dots \\ &\quad \omega_n \cdot (\vec{rule}_n) \end{aligned} \tag{3.1}$$

$$\vec{x_p}(t) = \vec{x_p}(t-1) + \vec{v_p}(t). \tag{3.2}$$

where:

- $v_p(t),\ v_p(t-1)$ are the boid's velocity at current and previous iteration.
- $x_p(t),\ x_p(t-1)$ are the boid's position at current and previous iteration.
- $\vec{rule}_1..\vec{rule}_n$ are flocking behaviors, vectors computed to satisfy certain conditions: cohesion, separation, alignment, leader following, random

movement...

- ω_0 is the weight that controls momentum.
- $\omega_1..\omega_n$ are weights that control the influence of each rule.

The beauty of this model lies in the unpredictable nature of emerging group dynamics. As Reynolds states,

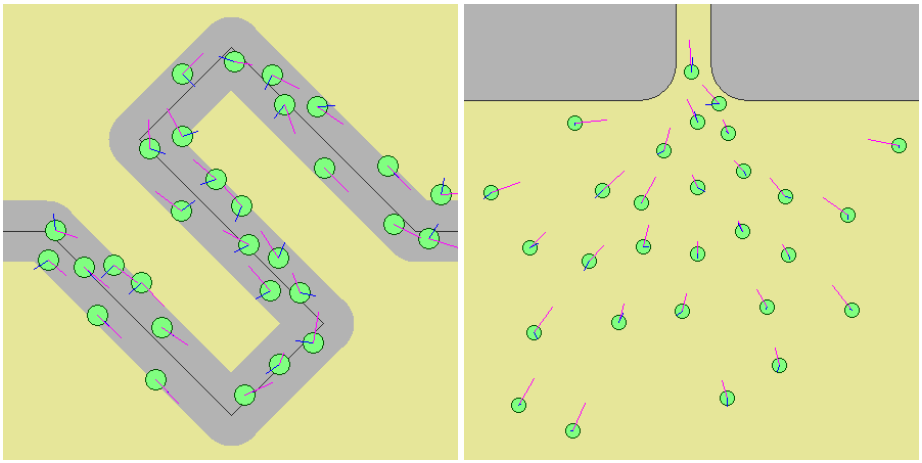
> *Flocking is a particularly evocative example of emergence: where complex global behavior can arise from the interaction of simple local rules.*

Indeed, Reynolds's boids wander in their virtual environment and do look like a swarm, a flock, a school or a herd, depending on the weights assigned to each rule. Combined with computer generated graphics, the effect is truly attractive, and it has been successfully applied to computer animations. Tim Burton's *Batman Returns* (1992) featured simulated bat swarms and penguin flocks that were based on Reynolds's boids; Disney's *The Lion King* (1994) included a wildebeest stampede that also used boids to guide dozens of animals in their frenzied rush; more recently, in *The Lord of the Rings* trilogy (2001-2003), a similar technology was applied to battle scenes. Their use is nowadays widespread in visual media when coordinated group dynamics are needed.

The boids model allows additional rules to be added, defining interactions with other virtual creatures and their simulated environment. Boids may follow a leader, flee from predators, look for food... It is just a matter of including new rules to the original set using vector arithmetic. In Figure 3.4 boids follow a crowded path and cross a gate avoiding collisions.

3.2 Bio-inspired Computing

Bio-inspired Computing relies on the fields of Biology, Computer Science and Mathematics, and it is tightly related to Artificial Intelligence, involving



(a) (b)

Figure 3.4: Different steering behaviors combinations lead to complex group dynamics. a) Crowd path following, which combines path tracking and separation, and b) Queuing (crossing a doorway), which combines seek, avoid and separation.

Connectionism, Social Behaviour and Emergence. Unlike traditional AI, where problems are solved from the programmer's perspective, relying on his/her knowledge and ability to design a suitable algorithms, bio-inspired computing applies solutions found in nature to problems similar to the one being tackled, using the analogy principle.

The corpse gathering strategy shown by Messor Sanctus ants explained in Chapter 1 (see Figure 1.2), can be easily mimicked in a virtual environment with independent and simple agents. They are programmed to walk around randomly, picking objects with a probability that depends on the presence of nearby objects and droping them next to other objects. Figure 3.5 shows how from an initial setting where objects are regularly placed on the virtual field, a low number of piles appear after some simulation steps. Indeed, this corpse gathering behavior has been successfully applied to data clustering (Handl et al., 2003).

Is it possible to apply biologic metaphors to solve numerical problems? Indeed, what eusocial animals do constant and efficiently is solving optimization problems: from task allocation to finding shortest paths between two points, insects apply their sheer numbers and decentralized control to find optimal

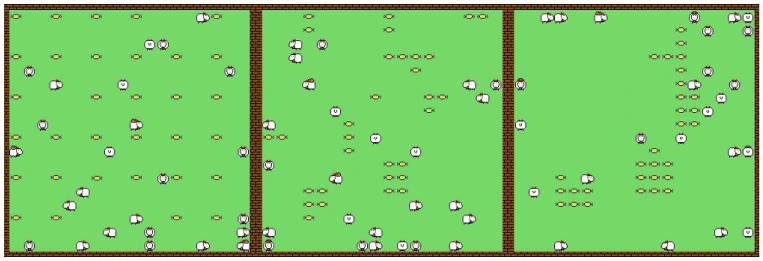


Figure 3.5: The corpse gathering strategy followed by Messur Sanctus ants (Jost et al., 2006) can be simulated with simple and independent agents.

solutions.

But even before insects appeared, evolution itself managed to find optimal solutions to adaptation and speciation, modifying genotypes, and thus phenotypes, in ways that permitted species to succeed in their environment.

3.2.1 Genetics

Biologic evolution has inspired a whole branch within Artificial Intelligence, called Evolutionary Computation or Evolutionary Algorithms, that aims to solve optimization and search problems using the same principles of evolution: selection, reproduction and mutation.

Genetic Programming proposes pools of randomly initialized computer programs that have a certain goal. Using appropriate operators, the best programs in each generation are evaluated and suffer crossovers and mutations (reproduction and variation), creating new programs for the next generation that will eventually improve their output (Koza, 1992) (Koza, 1994) (Koza et al., 1999) (Koza, 2003).

Genetic algorithms (Holland, 1992) (Mitchell, 1998), categorized as global search heuristics, are used to find solutions to optimization and search problems. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination), combining exploration (mutation) and exploitation (crossover). Genetic Algorithms use a population

of abstract representations (called chromosomes or the genotype of the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem.

Evolution happens in generations. In each one, individuals are first evaluated using a fitness function. Then, they are stochastically selected and crossed to populate the next generation with their offspring, which may randomly mutate. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations have been produced or a satisfactory fitness level has been reached. Parallelism is implicitly introduced by genetic algorithms, as each individual can operate independently.

The three most important aspects of using genetic algorithms are the definition of the fitness function, the definition and implementation of the genetic representation and the definition and implementation of the genetic operators. Beyond that, many different variations may be implemented to improve performance, like finding multiple optima (species) if they exist.

It is important to understand that the nature of such algorithms does not guarantee success. Being a stochastic system, the genetic pool may be still too far from the solution when the algorithm terminates, or a too fast convergence may halt the evolution process in a local extremum. These algorithms are nevertheless extremely efficient, and are used in fields as diverse as stock exchange, production scheduling or programming of assembly robots in the automotive industry.

3.2.2 Ant Colony Optimization

Ant Colony Optimization (ACO) was initially proposed by Marco Dorigo in 1992 in his PhD thesis (Dorigo, 1992), and it is still being studied and improved (Dorigo and Stützle, 2004). It can be considered one of the first applications of Swarm Intelligence to problem solving. In his work, Dorigo was inspired by the pheromone trails used by ants to create minimum paths towards food sources. He proposed a probabilistic technique for solving computational problems which can

be reduced to finding good paths through graphs. However, the original idea has since diversified to solve a wider class of numerical problems, drawing on various aspects of the behavior of ants.

The original ACO algorithm populated graphs with *virtual ants* that walked around mimicking their real-life counterpart behaviour. In the real world, ants initially wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely to stop travelling at random and follow the trail instead, returning and reinforcing it if they eventually find food. Over time, however, pheromones trail evaporate, thus reducing their attractive strength. In ACO, pheromone lay and evaporation rates were modeled and the attraction of ants by pheromones was statistically defined, creating a realistic simulation of foraging behavior.

When an ant finds a solution, pheromone is updated in the travelled graph path using the following expression:

$$\tau_{i,j} = (1 - \rho) \cdot \tau_{i,j} + \Delta\tau_{i,j} \tag{3.3}$$

where:

- $\tau_{i,j}$ is the amount of pheromone on a given edge i, j
- ρ is the pheromone evaporation rate
- $\Delta\tau_{i,j}$ is the amount of deposited pheromone, typically given by

$$\Delta\tau_{i,j}^{k} = \begin{cases} 1/L_k & if\ ant\ k\ travels\ on\ edge\ i,j \\ 0 & otherwise \end{cases} \tag{3.4}$$

where L_k is the cost of the k ant's tour (typically the path's length)

Ants traverse graphs choosing edges with a probability given by the following expression:

$$p_{i,j} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum (\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})} \tag{3.5}$$

where:

- $\tau_{i,j}$ is the amount of pheromone on a given edge i, j
- α is a parameter to control the influence of $\tau_{i,j}$
- $\eta_{i,j}$ is the desirability of edge i, j, typically $1/d_{i,j}$
- β is a parameter to control the influence of $\eta_{i,j}$

Some random movement factor can be also included to promote searches.

Ant colony optimization algorithms have been applied to many combinatorial optimization problems, ranging from quadratic assignment to protein folding or routing vehicles and a lot of derived methods have been adapted to dynamic problems in real variables, stochastic problems, multi-targets and parallel implementations (Dorigo and Stützle, 2004). In practice, the use of an exchange of information between agents via the environment (*Stigmergy*) is deemed enough for an algorithm to belong to the class of ant colony algorithms.

ACO, like Genetic Algorithms, is a stochastic population-based scheme. The main advantage of ACO over Genetic Algorithms is the ability of virtual ant colonies to adapt to dynamic environments (Ramos and Almeida, 2000).

3.2.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a stochastic, population-based algorithm for problem solving. First described in 1995 by James Kennedy and Russell C. Eberhart (Kennedy and Eberhart, 1995), although still very popular (Kennedy and Eberhart, 2001) (Dehuri and Cho, 2009), it was inspired by the social behavior of animal groups and the works of Reynolds (Reynolds, 1987) and Heppner (Heppner and Grenander, 1990). In PSO, however, group dynamics are not

modeled to simulate life-like creatures. Instead, PSO was designed explicitly to find solutions to optimization problems using social-psychological principles. PSO could be considered a specialized boids system flying in the solution space of a given problem.

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA) and Ant Colony Optimization (ACO). The system is initialized with a population of random solutions and searches for optima by updating generations (iterations). However, unlike GA, PSO has no evolution operators such as crossover and mutation. Instead, social influence and social learning enable particles to maintain cognitive consistency. Each particle represents a feasible solution, similarly to individuals in GA, flying through the multidimensional solution space. As individuals share their findings about a given problem, they reach an optimal solution as a group.

The swarm is modeled as a group of particles that have a position, a velocity and a memory. This memory stores the best position of a particle so far, and knowledge of the global and/or neighborhood best solution found by swarm mates ($lbest$, $gbest$ and $nbest$). Members of a swarm communicate good positions to each other and adjust their own position and velocity accordingly, moving through the solution space towards a better position. In its simplest form, a particle's update equations are similar to those used by boids (see Equation 3.1):

$$\begin{aligned} \vec{v_p}(t) &= \omega \cdot \vec{v_p}(t-1) + \\ &\quad \varphi_1 \cdot \vec{r_1} \cdot (\vec{gbest} - \vec{x_p}(t-1)) + \\ &\quad \varphi_2 \cdot \vec{r_2} \cdot (\vec{lbest_p} - \vec{x_p}(t-1)) + \\ &\quad \varphi_3 \cdot \vec{r_3} \cdot (\vec{nbest_p} - \vec{x_p}(t-1)) \end{aligned} \tag{3.6}$$

$$\vec{x_p}(t) = \vec{x_p}(t-1) + \vec{v_p}(t). \tag{3.7}$$

where

- ω is the momentum weight, which value should be slightly less than 1.0.

- $gbest$ is the best position found by any member of the swarm so far.
- $lbest_p$ is the best solution found by particle p so far.
- $nbest_p$, not always present, is the best solution found by swarm members in a neighborhood around p so far.
- φ_1, φ_3 represent the *"social"* component and φ_2 the *"cognitive"* component. Their values are usually around 2.0.
- $\vec{r_1}, \vec{r_2}$ and $\vec{r_3}$ are random vectors with each component being generally a uniform random number between 0 and 1.

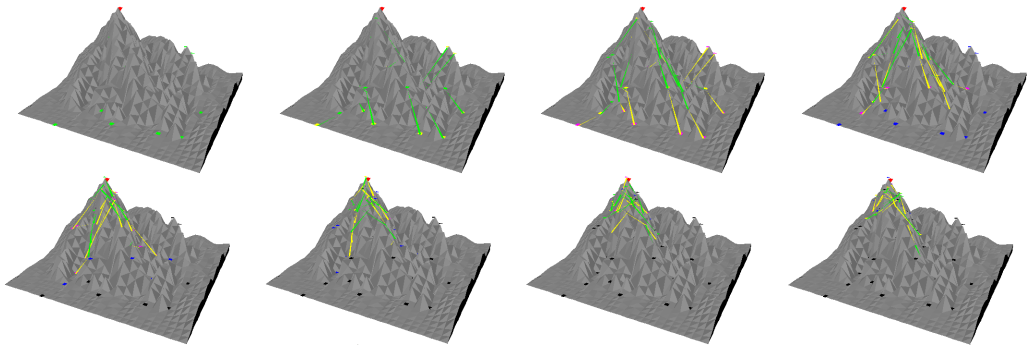


Figure 3.6: Particle Swarm Visualization : A randomly generated particle swarm of 12 particles attempts to find the global maximum on a 3D landscape. Yellow and green lines show previous and current movements, respectively. Note that, although this swarm overcame local maxima, this ability depends on the current implementation.

Compared to Genetic Algorithms, the advantages of PSO are many. It is easier to implement and there are fewer parameters to adjust, for instance. PSO can be seen as a basic search strategy that can be adapted as needed for the problem at hand. The adaptability of PSO is considered a strength over other robust evolutionary optimization mechanisms. One version, with slight variations, works well in a wide variety of applications, as well as for applications focused on a specific requirement.

Although its original formulation prevents PSO particles from adapting to dynamic functions, different approaches have been proposed to overcome this

Project Computing, 2004. http://www.projectcomputing.com/resources/psovis/index.html

problem (Hu and Eberhart, 2002). Another common modification consist on adding a random velocity vector, allowing particles to overcome local extrema thanks to stochastic exploration.

Over the years, many heuristics and variants determined to be better with respect to convergence speed and robustness, introducing variations like self-tuning parameters (Zhang et al., 2008). Very frequently, the values of φ_i are taken to decrease over time in order to facilitate exploitation over exploration in later states of the search. There are also other variants of the algorithm, such as discretized versions for searching over subsets of $\mathbb{Z}^n$ rather than $\mathbb{R}^n$ (Consoli et al., 2007). Significant, non-trivial modifications have been developed for multi-objective optimization of problems where it is believed or known that there are multiple global minima which ought to be located (Dehuri and Cho, 2009).

The first practical application of PSO was in the field of neural network training and was reported together with the algorithm itself (Kennedy and Eberhart, 1995). Many more areas of application have been explored ever since, including telecommunications, control, data mining, design, combinatorial optimization, power systems, signal processing, and many others. To date, there are hundreds of publications reporting applications of particle swarm optimization algorithms. For a review, see (Poli, 2008). Although PSO has been used mainly to solve unconstrained, single-objective optimization problems, PSO algorithms have been developed to cope with constrained problems, multi-objective optimization problems, dynamically changing landscapes and multiple solutions. For an in-depth review, see (Engelbrecht, 2005)(Kennedy and James, 2007).

3.3 Swarms in Computer Vision

Swarm Intelligence systems are typically made up of a population of relatively simple individuals interacting locally with one another and with their environment, allowing the emergence of collective behaviors. Decentralization and thus robustness due to swarm members expendability, relative simplicity, low

elemental computational costs and possible parallel implementations are the main advantages of this approach, leading to systems which abilities transcend those of individuals.

Many successful SI techniques have been developed during last years, including Ant Colony Optimization (ACO) (Bonabeau et al., 1999) and Particle Swarm Optimization (PSO) (Shi and Eberhart, 1998) (Hu et al., 2004) (Kennedy and Eberhart, 2001). They were originally designed to solve optimization problems, because that is precisely what social insects do best, but they have been also applied to computer vision tasks (Owechko and Medasani, 2005).

Digital images are interesting habitats for virtual swarming creatures. They provide meaningful environments from which swarming agents may extract useful information. Ant colonies inhabiting digital images are studied in (Ramos and Almeida, 2000) and (Ramos et al., 2005), where the colony adapts the size of its population according to the dynamic structure of three dimensional maps defined by grayscale images. Although single individuals only deal with local pheromone trails, pheromonal fields represent the memory of the recent history of the colony (see Figure 3.7). They are used to initialize a Watershed algorithm for image segmentation.

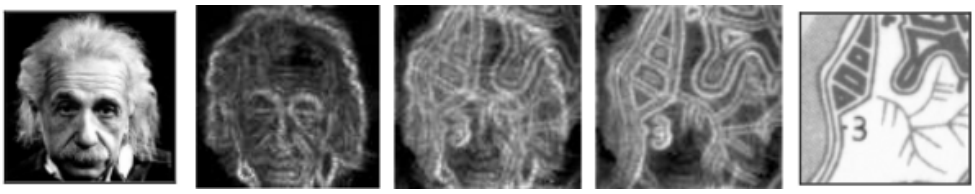


Figure 3.7: A digital ant colony adapts its population size according to the underlying image. After 100 iterations the image changes from Einstein to a Map, and the changing pheromonal field shows how the colony is able to reconfigure its own knowledge about the habitat it inhabits.

Swarms and ant colonies have been proposed for color clustering (White et al., 2004), data clustering and image segmentation (Abraham et al., 2007) (Ouadfel and Batouche, 2007). But not only ants inspire solutions. In (Bourjot et al., 2003) the nest building abilities of certain pseudo-social African spider are modeled and applied to region segmentation, and the *artificial fish swarm*

algorithm is also used for segmenting images in (Jiang et al., 2007). A general decentralized multi-agent system for data discovery and image enhancement is presented in (Jones and Saeed, 2007).

3.3.1 Swarms for object tracking

Object tracking in video sequences is an appropriate workbench for swarming solutions. Where traditional tracking methods may fail due to the unreliable nature of image matching functions, population-based methods may succeed thanks to the collaborative effort of many individuals.

In (Kobayashi et al., 2007), PSO is used to track objects using color features. Particles look for positions in the image that match a target color histogram. However, only one sequence is tested in ideal conditions, and the tracked object, a green ball, is extremely different from its surroundings.

In (Zheng and Meng, 2007), PSO is used to track the location of a target window between frames in a video sequence. Particles represent window hypothesis defined by their position, width, height and orientation. Using a fitness function based on the appearance histogram of the last known window, particles in a new frame evaluate and update their position until the new location, orientation and scale of the tracked object are found.

Similarly, in (Zhang et al., 2008), PSO particles are characterized by a state that includes 2D translation and four deformation parameters. Particle fitness is evaluated using an appearance model based on a Mixture of Gaussians that describes the image region represented by the particle. Additionally, the traditional PSO algorithm is improved adding temporal coherence, a key factor in object tracking. PSO parameters are also changed adaptively according to the fitness value of particles and the predicted motion of the tracked object, instead of using constant parameter values. In both works the swarm finds the region that optimizes the fitness function in a new frame in a mode-seeking process similar to mean-shift, effectively tracking the object in short sequences.

Steering behaviors are used in (Kolsch and Turk, 2005) for tracking a flock

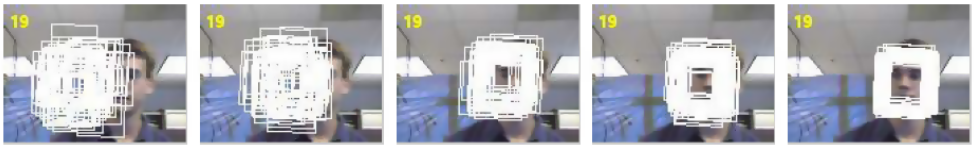


Figure 3.8: Sequential Particle Swarm optimization for visual tracking (Zhang et al., 2008). Particles (white rectangles) converge to the location of the tracked object.

of features in a scheme more similar to Reynolds's boids. Flocking is used to choose which KLT features should be tracked from frame to frame. Movement rules dictate that no two features must be closer to each other than a threshold distance (which would be similar to the *Separation* rule for Boids), and that no feature must be further from the median than a second threshold distance (*Cohesion* rule). Unlike in PSO and Boids, particles in this approach are abruptly relocated onto good color spots that meet flocking conditions.

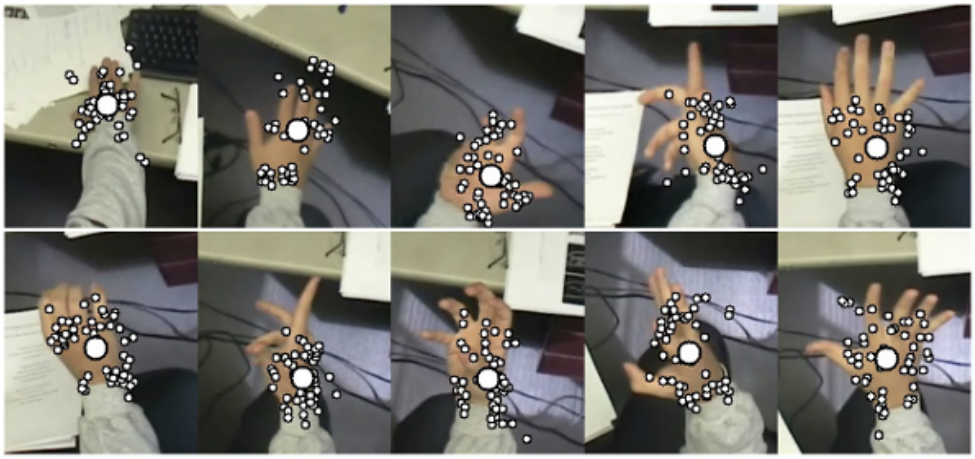


Figure 3.9: Tracking a flock of KLT features. Small dots represent KLT features. Their median position is represented by the big dot.

4 Tracking Swarms

A solution based on a Swarm Intelligence metaphor with a prey-predator analogy is proposed for real time object tracking in video sequences. Based on boids, the seminal proposal by Craig Reynolds (Reynolds, 1987), particles fly in the bidimensional space defined by digital images using combined image features to guide individual movement. Object tracking emerges from the interaction between predator particles and their dynamic environment.

4.1 Introduction

Tracking in Computer Vision, as seen in Chapter 2, is the process of locating and following moving objects as they evolve through time and space in video sequences. Traditional tracking approaches are based on the use of models or templates that represent target features in the spatio-temporal domain and involve the use of two main processes: matching and updating (Yilmaz et al., 2006). Matching corresponds to the process in which a reference template is searched for in an input image to determine its location. Template updating is related to the process of replacing the template that represents the target in such a way that it does not drift away from the object (Matthews et al., 2004) due to possible appearance changes during a sequence.

Tracking moving objects becomes critical in multiple computer vision tasks,

such as vision based interfaces, visual surveillance or perceptual intelligence applications. At present, there are still obstacles in achieving all-purpose and robust tracking systems. Different issues must be addressed in order to carry out an effective tracking approach: dynamic appearance of deformable or articulated targets, dynamic backgrounds, pursuing different target motions without restriction, changing lighting conditions, camera (ego) motion and real-time performance, among others.

4.2 Predator Swarm Model

In (Antón-Canalís et al., 2006c) a precategoric visual tracking approach based in a Swarm Intelligence paradigm was presented, defining the tracking problem as a predator-prey metaphor in the ecosystem created by a video sequence. Inspired by the individual activity of social insects and distributed behavioral models (Reynolds, 1987), tracking is understood as an emergent property of the swarm of predator particles as they individually chase prey pixels across frames. Particles look for prey pixels guided by the intensity of their *scent* (a combination of image features). Neither complete aspect based templates of the visual target nor dynamical models of the object's motion are required. In (Antón-Canalís et al., 2006b) an improvement was proposed, broadening the swarm's perception abilities in order to strengthen robustness and increase stability.

4.2.1 Feature Detection Network

In the initial frame of a video sequence a predator swarm is placed over the herd of prey pixels that will be tracked. Each particle obtains information of the video sequence from a set of feature maps $\vec{F} = f_1, f_2..f_m$ created by a Feature Detection Network. This network is composed by a number of Feature Detectors FD_i that extract and transform values from visual data or feature maps in order to create a new feature map (see Figure 4.1). Without loss of generality, all feature map values should be normalized to range $0..1$.

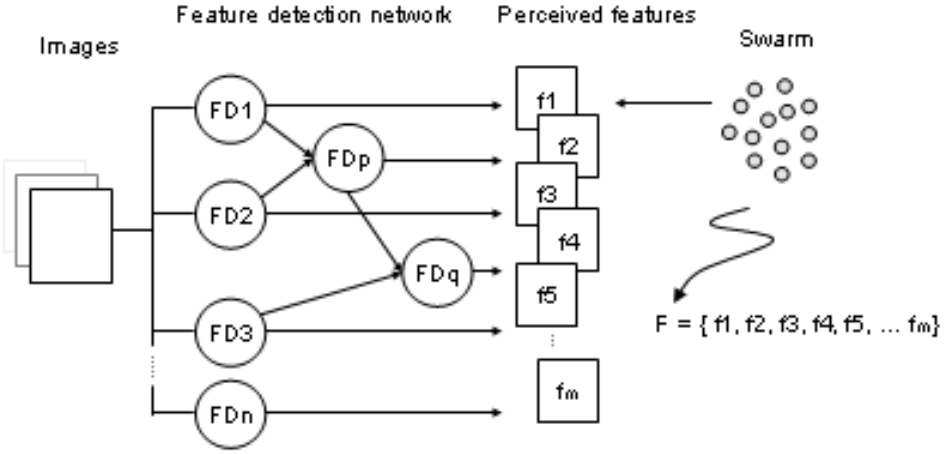


Figure 4.1: Swarm particles get information from feature maps, which are computed from images or created by particles themselves.

These features characterize the *scent* of a prey, which is an abstraction for the combination of transformed image values that particles will chase. From pixel intensities to maps created by the same particles, like pheromonal fields, any possible image transformation can be considered.

Feature maps can also be used to modulate prey *scents*, defining their *intensity*. That way, preys with a certain *scent* become more attractive if they meet certain conditions (e.g. they are placed on an image region where movement is perceived, where a figure stands out from the background or where pheromone deposits have higher concentration levels).

4.3 Swarm Definition

More formally, given a Particle Swarm $\mathbb{S}$ inhabiting a video sequence from which a set of feature image maps $\vec{F}$ are extracted for each frame, each particle p is a 4-tuple: $p =< \vec{x}_p,\ \vec{v}_p,\ c_p,\ \vec{sl}_p >, p \in \mathbb{S}$ where:

1. $\vec{x}_p$: Position vector in the search space.

2. $\vec{v}_p$: Velocity vector.

3. c_p: Particle's comfort, a measurement of its tracking performance in the current time step, with a value between 1.0 (best) and 0.0 (worst). It will be explained in detail later.

4. $\vec{sl}_p$: Scent list, a list of desired prey scents, obtained from $\vec{F}$. It stores a description of the object to be tracked.

The swarm as a whole is characterized by its centroid and its velocity. Both values are computed from the weighted average of each particle's position and velocity, using c_p as a weighting factor.

Tracking emerges from the observation of the trajectory described by the swarms' centroid $\vec{\mathbb{S}}_c$ and its velocity $\vec{\mathbb{S}}_v$ (see Equation 4.1 and Equation 4.2). Each particle contributes to the localization of the tracked object independently, and its contribution is proportional to its tracking performance. Those particles with better comfort values, that is, particles which are closer to their targets, become influential leaders. They are much more relevant to the global behavior of the swarm than those particles that may have lost their prey.

1. Swarm's centroid, which is the Swarm's weighted center of mass:

$$\mathbb{S}_c = \frac{\sum_p \vec{x}_p \cdot c_p}{\sum_p c_p} \; \forall p \in \mathbb{S} \tag{4.1}$$

2. Swarm's velocity, which is a weighted average of all particle's velocity vectors:

$$\mathbb{S}_v = \frac{\sum_p \vec{v}_p \cdot c_p}{\sum_p c_p} \; \forall p \in \mathbb{S} \tag{4.2}$$

Figure 4.2 depicts an example of a recently created swarm that perceives image information using an RGB color map. The swarm is represented as a grid of particles that hovers over the image.

While each predator particle looks independently for interesting preys in its vicinity, trying to move closer to them, it will also try to keep up with the rest of the group following a set of Boid-based flocking rules.

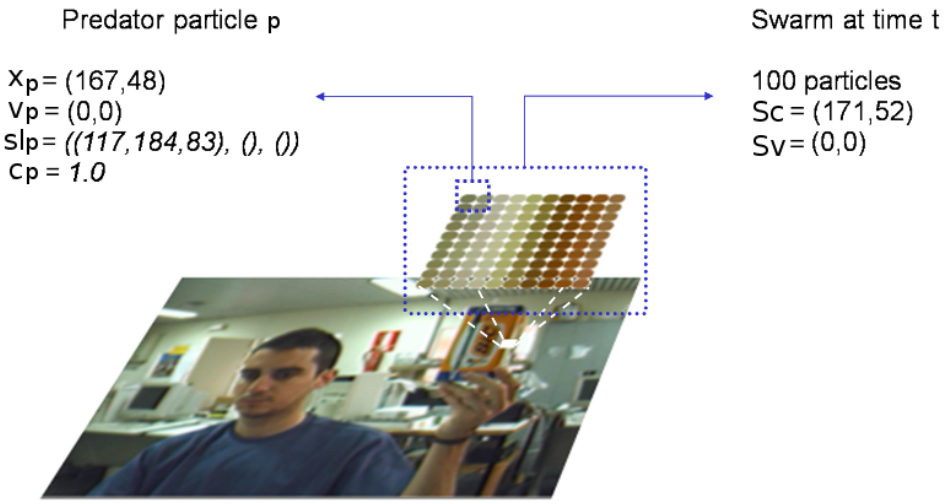


Figure 4.2: A predator swarm is placed over the herd of prey pixels that will be tracked. Each particle is represented by the color of the pixel it will chase. Details for the first particle are shown in the upper-left corner: position, velocity, list of features (RGB triplets) and comfort. In the upper-right corner, some global swarm features are shown, including the number of particles, the swarm's centroid and its velocity (which is zero because it is the initial frame).

4.4 Particle Movement Rules: tracking and flocking

Reynolds's steering behaviors (Reynolds, 1987) are movement rules that define a maneuver strategy for each member of a group of mobile entities. Flocking, Path Following or any other model can be obtained from the linear combination of many steering behaviors. The present tracking swarm was inspired by boids, so particles move following a similar scheme (see Chapter 3, Section 3.1.1).

Swarm movement and preying behavior, and thus tracking activity, emerges from the interaction of each particle moving in response to neighboring preys (pixels) and the rest of the swarm. For each particle, a first movement rule called Hunt defines the tracking behavior, moving particles towards the best prey in their neighborhood. Then, flocking is integrated using a modified version of the three rules that define Reynolds's boids flocking model: Cohesion, Alignment and Separation.

For each predator particle $p, p \in \mathbb{S}$, movement rules are defined as follows:

4.4.1 Rule 1: Hunt

The *hunt* rule defines a vector towards the location of the most interesting preys. This rule is added to the three basic steering behaviors in order to give boid swarms the ability to track objects in video sequences. Following this rule, the swarm perceives and reacts to visual information through each particle's Feature Detection Network. Particles analyze their vicinity in the current image and head towards the most interesting preys. Thus, it acts as a basic local search.

Preys are found in an nxn neighborhood N around p, where they are defined as $q =< \vec{x}_q,\ \vec{s}_q >, q \in N$, being $\vec{x}_q$ the prey's position and $\vec{s}_q$ its *scent*.

If β_{qp} represents the interest that prey q creates in predator p, that is, the similarity between a particle's list of tracked features $\vec{sl}_p$ and a given pixel's set of features, the following expression defines the hunting movement velocity:

$$\vec{V}_1^p = \frac{\sum_q (\vec{x}_q - \vec{x}_p) \cdot \beta_{qp}}{\sum_q \beta_{qp}} \forall q \in N \tag{4.3}$$

How to compute β_{qp} will be explained in detail later. This rule also obtains a particle's comfort c_p from the interest created by the best prey in the neighborhood. A particle placed near suitable preys will get higher comfort values, and thus this variable measures tracking performance.

$$c_p = \max_{q \in N} \{\beta_{qp}\} \tag{4.4}$$

Figure 4.3 depicts the velocity vector computed by this rule.

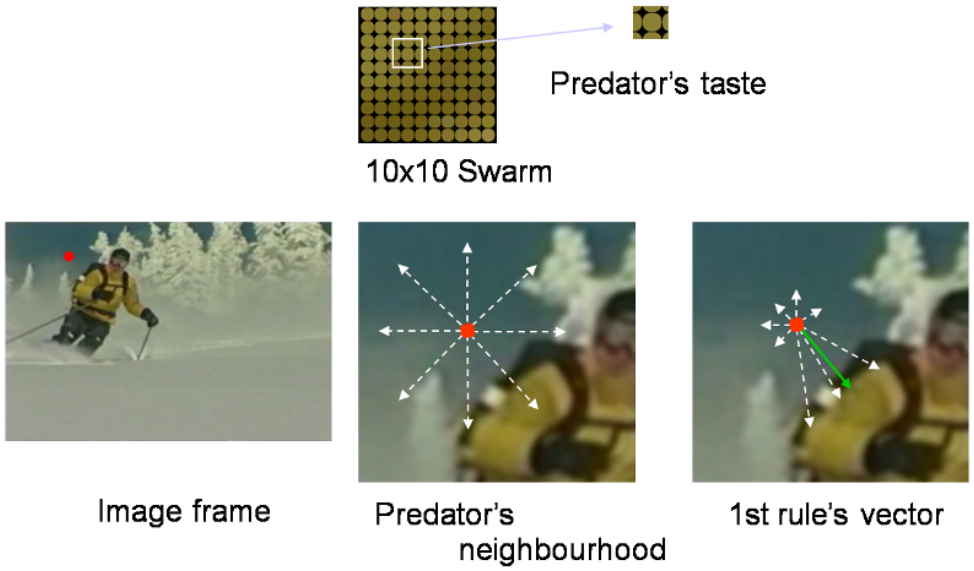


Figure 4.3: A particle (shown by a red dot) analyzes its neighborhood looking for preys that suit its taste. Arrow sizes on the right-most image are proportional to β_{qp}. The resultant of Rule 1, represented by a green arrow, points towards the region containing more interesting preys.

4.4.2 Rule 2: Cohesion

This rule defines a vector from the particle's position $\vec{x}_p$ towards the perceived weighted center of mass of flockmates, $\vec{\mu}$:

$$\vec{\mu} = \frac{\sum_r \vec{x}_r \cdot c_r}{\sum_r c_r}, \forall r \in \mathbb{S}, r \neq p$$

$$\vec{V}_2^p = \vec{\mu} - \vec{x}_p \tag{4.5}$$

Cohesion avoids scattering, keeping swarm members together. It acts as an exploitation strategy in the tracking process. This rule differs from Reynolds's Cohesion in that positions are weighted. Flockmates with higher c_r, those that consider they are properly tracking their target, are more influential leaders.

4.4.3 Rule 3: Alignment

This rule computes the average velocity of flockmates, as perceived by particle p:

$$\vec{V}_3^p = \frac{\sum_r \vec{v}_r \cdot c_r}{\sum_r c_r}, \forall r \in \mathbb{S}, r \neq p \tag{4.6}$$

Particles will imitate flockmates' movement, so this rule acts like a voting system where the majority decides where the swarm should head to. Once again, this rule differs from Reynolds's Alignment in that velocities are weighted, being those flockmates with higher c_r more influential.

4.4.4 Rule 4: Separation

Computes a vector that moves particle p away from flockmates that are too close:

$$\vec{V}_4^p = \sum_r \vec{x}_p - \vec{x}_r, \forall r \in \mathbb{S}, r \neq p, d(\vec{x}_p, \vec{x}_r) < n \cdot 0.5 \tag{4.7}$$

Being n the neighborhood size and $d()$ the Manhattan distance between both particles. This rule avoids crowding, forcing particles to cover wider spaces and favoring exploration.

4.4.5 Final particle displacement

The four resultant velocities are weighted and added to a fraction of the previous iteration velocity for each particle. Finally, positions are updated according to new velocities:

$$\begin{aligned} \vec{v}_p(t) &= w_0 \cdot \vec{v}_p(t-1) + \sum_{i=1}^{4} w_i \cdot \vec{V}_i^p(t), \\ \vec{x}_p(t) &= \vec{x}_p(t-1) + \vec{v}_p(t) \end{aligned} \tag{4.8}$$

This expression is, effectively, a generalization of the one used in Particle Swarm Optimization (see Chapter 3, Section 3.2.3), where only three rules are considered. It is, however, the same expression used in the classic Boid system (see Chapter 3, Section 3.1.1). Figure 4.4 depicts how rules are combined.

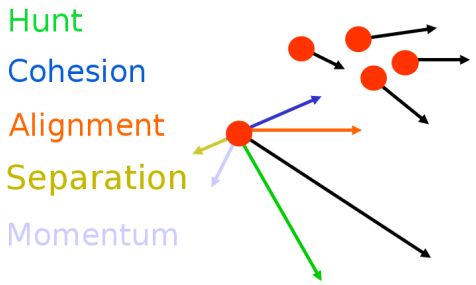


Figure 4.4: The four steering behaviors plus a certain momentum define a particle's velocity on each time step.

4.5 Constants and weighting strategies

The number of required weights and constants is a major drawback of this method, although they are part of its versatility. There are many parameters that must be chosen and tuned to achieve successful tracking, but once they are determined the swarm behaves similarly in most situations. Three constants are defined:

- w_0, momentum, is a fraction of previous velocity. It should be a low value. Otherwise, particles may build up velocity and start acting erratically. $w_0 = 0.1$
- n, measured in pixels, defines the search area around a particle. If it is too

small, particles may not be able to keep up with fast moving objects. If it is too big, search becomes computationally expensive. $n = 11$

- $\|\vec{sl}_p\| = 1$, the size of the scent list. It limits the number of scents that a particle stores. $\|\|vecsl_p\|\| = 1$

While the current approach is not taking advantage of the scent list because it currently contains a single feature vector, it may be used in a future updating mechanism to store the most interesting appearances of a particle's target and avoid drifting. It can be also useful to store a population-based descriptor of prey appearances, keeping a group of sample targets.

With these constants set, two weighting strategies are proposed: the first one considers the same set of static weights for each particle, while the second one considers dynamic random rule weights that change smoothly with each time step.

4.5.1 Static weights

For the considered tracking application, most parameters can be set intuitively:

- Rule 1, Hunt, should be assigned the highest weight, because particles should be allowed to track their target.
- Rule 2, Cohesion, should keep particles together, but no so close that they are not allowed to explore.
- Rule 3, Alignment, pushes particles in the same direction. A high weight creates flock-like movement, while a low weight results in swarm-like movements. Given that particles require a certain freedom to achieve tracking, the swarm model seems more suitable.
- Rule 4, Separation, should not allow particles to converge to the same point, but at the same time it should not push correctly placed particles away.

After empirical tests, the following constants are proposed:

- $w_1(Hunt) = 1.0$
- $w_2(Cohesion) = 0.3$
- $w_3(Alignment) = 0.5$
- $w_4(Separation) = 0.01$

4.5.2 Dynamic weights

While proposed rule weights seem to work well on most situations, it can be argued that manually tuning their values is an ad hoc solution. Thus, instead of finding the optimum set of parameters for each situation, a complete stochastic approach can be adopted.

Each particle, at each time step, shall update the weights of its movement rules. Changing weights promote heterogeneous behaviors: as weights change particles will show different moods, alternating unsocial behaviors that compel particles to stay away from the group with gregarious responses that promote coordinated particle clusters.

Weights could simply change randomly, obtaining their values from white noise. However the output of pseudo-random number generators is too harsh, as seen in the top row of Figure 4.5. Because weights dictate a particle's attitude towards group participation, they should fluctuate smooth and naturally.

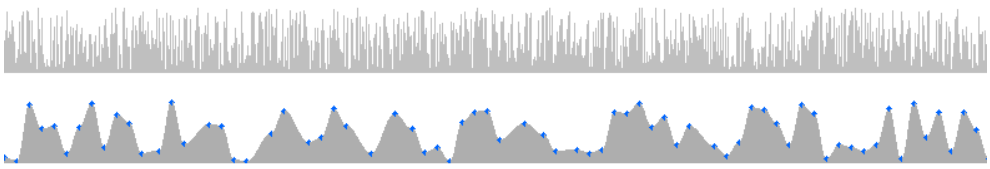


Figure 4.5: Top row: pseudo-random white noise signals are too harsh to appear natural. Bottom row: a noise signal created from the cosine interpolation of random values (blue dots) defined along the dimension of the function.

Perlin Noise functions, proposed by Ken Perlin in (Perlin, 1984), create a noise signal from the composition of smooth noise at different scales, preserving continuity and control over spatial (and temporal) frequency. Since its original form, it has been extensively used to create natural looking random signals (Perlin, 1985) (Perlin and Hoffert, 1989) (Perlin, 2002), and applied in computer graphics for the procedural generation of volumetric clouds, water, terrain, fire, smoke and many other effects.

A noise wave function can be used to create a noise signal that changes smoothly in time. It can be computed in a way similar to the first octave of Perlin Noise, although control over spatial (or temporal) frequency is not required. Instead, a piecewise wave signal function is created, handling one section at a time. Given two random amplitude points a and b, and an interpolation value p, the cosine interpolation between a and b at a fraction p is computed as follows:

$$CI(a,b,p) \quad = \quad a \cdot (1.0 - \frac{1.0 - cos(p)}{2.0}) + b \cdot \frac{1.0 - cos(p)}{2.0} \tag{4.9}$$

Each time this noise wave function is called, the next interpolated value is returned. When the interpolation between the two points that define a section is completed ($p = 1.0$), a new section is created between the current value and a new random point. These key points are shown in Figure 4.5 as blue dots.

The following algorithm is used to generate this kind of random numbers:

Require: $p = 1.0$, $step = 0$, $currentAmplitude = Random(0,1)$, $nextAmplitude = 0$, $currentValue = currentAmplitude$ {Requisites are initialization values, successive calls reuse previous values.}

Ensure: $currentValue \in [0..1]$

$p = p + step$

if $p >= 1.0$ **then**

$nextAmplitude = Random(0,1)$

$step = Random(MINSTEP, MAXSTEP)$

$currentAmplitude = currentValue$

$p = 0.0$
end if
$currentValue = CosineInterpolate(currentAmplitude, nextAmplitude, p)$
return $currentValue$

where $Random(m, n)$ is a pseudo-random number generator that returns a random value between m and n. Using a random $step$ value to increment p in each function chunk is similar to choosing random wavelengths for each section.

Such a signal can be used, for example, to create a natural looking wandering behavior. Instead of choosing a walking direction from a white noise signal, which would result in an awkward trajectory, a virtual creature using a noise wave to choose its next walking direction will look much more natural.

Back to random rule weighting, using a noise wave to alter rule weights results in a more natural behavior progression. That way, a particle that *decides* to steer away from the group and explore nearby regions will have more time to find whether its new location is interesting. Its comfort may increase and thus it may attract other particles towards this new region. White noise would not induce this course of action. Particles using white noise would show unstable behaviors leading to bouncy movements. Figure 4.6 shows the trajectories created by some swarm particles while following the box in the sequence shown in Figure 4.2, using three different weighting policies: fixed values, white noise random values and noise wave values. The latter creates smoother and more compact trajectories.

(a) (b) (c)

Figure 4.6: Particle trajectories using (a) fixed rule weights, (b) white noise random weights and (c) noise wave weights.

4.6 Prey scents and scent intensity

Given a prey q, its *scent* $\vec{s}_q$ is a vector computed from image feature maps. As previously said, a tracking particle will move towards the location of the most interesting preys in its neighborhood (i.e. pixels which features are more similar to those stored by a particle when it was created). The attraction a prey q creates on particle p is given by:

$$\alpha_{qp} = \max_{\vec{s} \in \vec{sl}_p} \{ e^{-(||\vec{s}_q - \vec{s}||)/\delta} \} \tag{4.10}$$

which yields a value between 0 -not interesting- and 1 -most interesting-. $\|.\|$ represents the norm and $||\vec{s}_q - \vec{s}||$ a distance function in the *scent* space. It measures the difference between pixel features and those a particle is looking for. If *scents* are color vectors, then $\|.\|$ may be a L_1 norm in the color space (e.g. RGB space), computing the difference between the color of preys $\vec{s}_q$ and those stored in the particle's feature list $\vec{s}, \vec{s} \in \vec{sl}_p$.

Scent similarity bandwidth is defined by parameter δ. In the present approach, it should be low enough to discriminate between colors in the RGB space. As RGB values are being measured between 0 and 255, $\delta = 5.0$ results in particles being selective.

Scents can become more intense if certain conditions are met, although the use of *scent* intensities is completely optional and task-dependent. Depending on the application, certain features may reveal the presence of an object (e.g. figure-background segmentation or motion detectors in a surveillance system), or regions where other particles have successfully achieved their task (e.g. pheromone maps in an ACO-like solution).

Therefore, once an attraction value α_{qp} is computed, it can be modulated using a number of intensity modifiers $g_i(\vec{x}), i \in 1..m$. These modifiers can be combined using their product, so β_{qp} would quickly decay if one of the values is low (Equation 4.11), the not so restrictive average of modifier values (Equation 4.12), or any other suitable function.

$$\beta_{qp} = \alpha_{qp} \cdot \prod_{i=1}^{m} g_i(\vec{x}_x) \tag{4.11}$$

$$\beta_{qp} = \alpha_{qp} \cdot \frac{\sum_{i=1}^{m} g_i(\vec{x}_x)}{m+1} \tag{4.12}$$

The interest β_{qp} that prey q creates in predator p, introduced in Rule 1, is thus the modulated attraction value that a certain pixel has on a given particle. If no modulation is considered, then $\beta_{qp} = \alpha_{qp}$.

The present work considers the following features to modulate prey *scents*:

1. $(1.0 - |\triangle I(\vec{x}_q)|)$, being $\triangle I()$ the image gradient. A high value of this feature for a given pixel means that it is not placed on a high gradient region. Gradients correspond to object boundaries. Because particles should remain inside the tracked object, it seems sensible to avoid gradients. The opposite strategy is also considered, as proposed in (Antón-Canalís et al., 2006b) and (Antón-Canalís et al., 2006c). That way, high gradient regions ($|\triangle I(\vec{x}_q)|$) are more relevant, considering that there may also exist high gradient regions within the object.

2. $(1.0 + M(\vec{x}_q))$, being $M()$ the amount of movement, computed from the subtraction of consecutive frames: pixels are more interesting if they contain movement (similarly to frogs, which visual system is known to be tuned to detect certain movements in their visual field (Lettvin et al., 1940)). However, this feature only increases a pixel interest value, which is important to allow the swarm to wander over an object that has become static.

3. $\vec{x}_q \in SPN$, SPN standing for Salient Point Neighborhood and representing those pixels which distance to the closest salient point is less than a given

threshold λ. If a pixel does not lie in the vicinity of an image interest point, which are ideally good features to track, its interest value could be halved. λ, the size of salient point neighborhoods in pixels, should be low (around 3.0), because only those pixels that are close enough to a salient point should be considered.

4.7 Other considerations

Many other processes may be introduced in the tracking boids scheme. The following are just some considerations:

4.7.1 Image preprocessing

A wide variety of hand-held cameras, webcams and videos downloaded from Internet have been used in this work, and many of them showed perceptible compression artifacts and noisy colors. In order to smooth the negative effect of noise, three filters were considered: a strong low-pass Gaussian filter, as wide as half the search region, a small median filter and a small bilateral filter with wide frequency range. Bilateral filters (Tomasi and Manduchi, 1998) are faster than same-sized median filters (they do not need to do any kind of sorting) but much slower than bigger Gaussian filters. However, high frequencies are preserved, so images corrected with a bilateral filter conserve most original boundaries. See Figure 4.7.

4.7.2 Weighted search windows and particle tiredness

Points in search windows could be assigned lower weights the further away they are from the center of the window, penalizing larger jumps. A similar solution consists on particles getting *tired* after consecutive long jumps, and slowly recovering with slow movements. That way, tired particles would become worse leaders. See Figure 4.8.

(a) (b) (c) (d)

Figure 4.7: The left-most image (a) is smoothed using b) a Gaussian filter (aperture 11), c) a Median filter (aperture 5) and d) a Bilateral filter (aperture 5, range 64, frequency 3). Bilateral filters preserve original boundaries.

This solution assumes that objects move smoothly. However, it prevents particles from following fast moving objects. It is suitable if the capture device and processing rates are fast enough.

(a) (b)

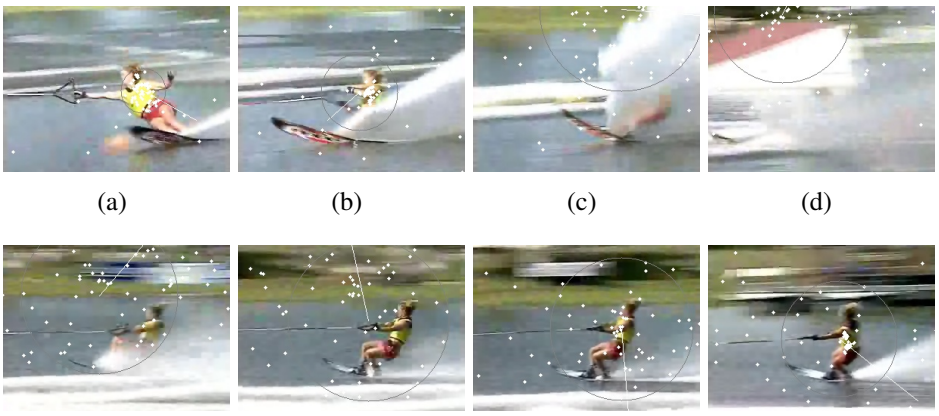
Figure 4.8: a) All particles are fresh when recently created (fresher particles are drawn in whiter color). b) If they move fast, they grow tired, but they recover if they move slowly. Tired particles could become worse leaders.

4.7.3 Improved exploration

In order to allow the swarm to explore wider spaces, random searches can be introduced as a new steering behavior. A random movement vector can be added to the three existing flocking rules. Its magnitude could be inversely proportional to the particle's comfort value: the lower the comfort, the larger the random movement. That way, lost particles would be able to look for their prey far away

from the swarm's centroid.

This behavior allows the swarm to recover from complete object occlusions (see Figure 4.9). However, it may also lead particles to find a similar looking background object. This strategy should be combined with a measure of particle tiredness, seen before, to avoid scout particles from becoming too influential.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.9: In this sequence, the skier is occluded by water spray in c). At that point, most particles have lost their target and thus they start moving randomly until the object reappears in f), after 40 frames. Then they regroup and recover their tracking activity. The circle position on each picture shows the swarm centroid, while its radius represents particle swarm dispersion.

A more deterministic approach to promote exploration consists on particles adopting a hierarchical search strategy using image pyramids (Adelson et al., 1984). Particles could increase the size of their search window as their comfort value decreases, obtaining the search windows from the appropriate image pyramid level. The lower a particle's comfort value, the higher the pyramid level from which the particle obtains its search window. Particles at higher pyramid levels would be able to look for preys further away from their current position.

4.7.4 Pheromone maps

As previously mentioned, an interesting feature map created by the same swarm consist on building a pheromone map similar to those created in ACO solutions. Each Boid could deploy on its position or recent trajectory a certain amount of pheromones, possibly related to its comfort value (or simply a constant value). Consecutive deployments by the same or by any other particle would increase the pheromone accumulation at that point, which would evaporate with time (see Figure 4.10).

Pheromone maps register the memory of the recent history of the swarm, revealing the approximate region where the tracked object lies. This information could be exploited by particles to steer towards stronger pheromone concentrations in order to keep the group together, steer away from them to improve exploration or segment the tracked object.

(a) (b) (c) (d)

Figure 4.10: Pheromone map at different times, computed from a group of particles following a trajectory similar to that shown in a). Values are inverted for visualization purposes, so darker points have stronger pheromone concentration levels.

4.8 Differences with PSO approaches

The present approach differs greatly from PSO, although both methods share the same roots: the social behavior of animal groups and the works of Reynolds (Reynolds, 1987) and Heppner (Heppner and Grenander, 1990). However, while PSO performs global searches in the solution space, boids-based tracking performs local searches.

PSO particles represent solutions that evolve within the solution space, while proposed Boid-like particles fly in the bidimensional space created by digital images. This characteristic allows Boid-like particles to follow heterogeneous goals: while one particle may follow color, another particle may follow edges or textures. Steering behaviors still apply, so even when the goal of particles may be different, they still belong to the same swarm. PSO particles belonging to the same swarm, on the other hand, must be homogeneus. Their position represents a feasible solution, so all particles must have the same structure in order to be able to move towards other swarm members.

PSO approaches perform a number of iterations before finding a single, good solution. Thus, when applied to tracking in video sequences, PSO-based solutions require a certain number of iterations on a given frame in order to find the current target's position before advancing to the next frame. Moreover, although many solutions have been proposed to adapt PSO to dynamic environments, particles are usually forced to *forget* previous positions. Tracking based on boids runs continuously, with particles updating their position independently in each frame, naturally spreading the current swarm's findings to future frames.

Tracking emerges from the state of all particles, locating the tracked object at the swarm's centroid and estimating its velocity from the averaged velocity of all swarm's particles. PSO relies in the goodness of a single particle, the optimal solution. Both approaches are completely different in this aspect: while Boid-like particles evolve independently, optimizing their individual comfort while following flocking rules, PSO particles must converge to a single point, a single solution.

4.9 Discussion

Hunting behavior and cooperative social interactions lead particles towards those areas in the image which are similar to that where the swarm was created, emerging an object tracking behavior for rigid and non-rigid objects where the swarm centroid and velocity describe the position and velocity of the tracked object. Tracking is enhanced through individual comfort optimization: the swarm solves the spatial optimization problem in a greedy way, maximizing each particle's comfort and thus minimizing the swarm's centroid distance to the tracked object.

(a) A single particle may stick to a piece of background easily...

(b) ...but many particles are able to stay within the object.

(c) Flocking behavior may keep particles over the tracked object.

Figure 4.11: Combined steering behaviors may correct misplaced particles.

A single particle would not be able to track an object for a long time. It would be easily attracted to background pixels with features similar to what the particle is looking for (Figure 4.11 a). However, when observing some hundred independent particles trying to follow their correspondent prey pixels, it becomes apparent that some of them are able to stay on the object for longer time periods (Figure 4.11 b). These particles may guide lost particles towards the right object location using flocking behaviors (Figure 4.11 c).

The swarm is able to follow its target in a varied number of cluttered

backgrounds and light conditions. Due to the absence of structural rigidity in the tracked template, objects that deform and change their scale can be easily tracked. Figure 4.12 shows four sequences where the swarm successfully tracks its target.

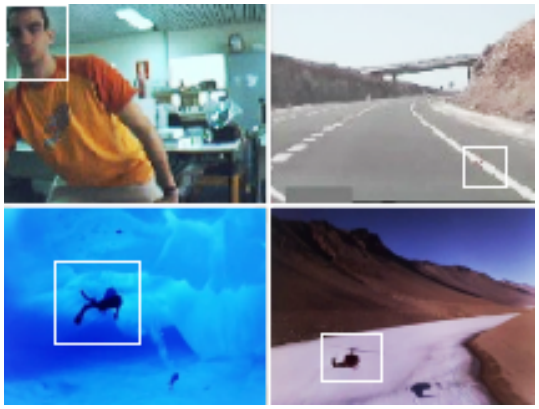


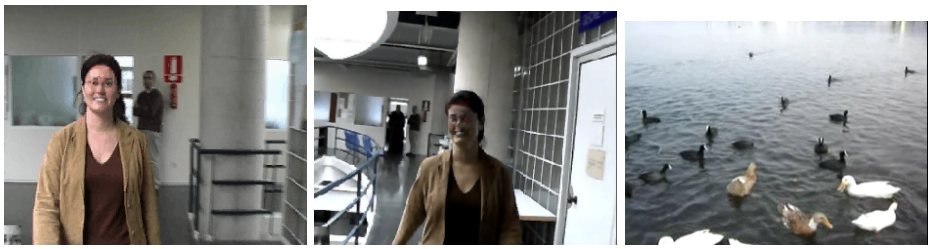
Figure 4.12: successfully tracked objects, enclosed in a white rectangle.

It is important to point out that, precisely because of the absence of structural constraints, the swarm floats freely over tracked objects. If an object has homogeneous features and it is large enough, the swarm's centroid will wander inside the object. In the road line example in Figure 4.12, the swarm's movement had to be restricted vertically.

Figure 4.13: Fast moving object sequence.

Fast moving objects or sudden velocity changes may confuse the swarm. If the capture frequency of the capture device is not high enough, fast moving objects will perform sudden jumps that will leave the swarm behind. If the length of these jumps is larger than the combined particles' search space size, the swarm will lose its target. However, it will still try to pursue it in an attempt to keep up, as shown in Figure 4.13, if enough particles manage to keep the object at range.

Current predator particles do not update their target's appearance. Thus,



(a) (b) (c)

Figure 4.14: Tracking may fail when tracked objects experience appearance changes, like the face in a and b, because no updating mechanism is currently considered. The swarm may also switch its attention to nearby objects if their appearance is similar to that of the tracked object, as seen in c.

if the tracked object suffers sudden chromatic changes, as it happens in Figure 4.14(a,b), particles will not be able to update their target scent and thus they will lose the object. In addition, objects surrounded by similar objects like those in Figure 4.14c will also be difficult to track, because the swarm does not use contextual information.

Chapter 6 will offer detailed numerical results on five representative sequences.

5 Sentient Ragdolls

A solution based on a structure that simulates articulated rigid body dynamics is applied to real time object tracking in video sequences. A number of independent tracking particles follows their targets while subject to distance constraints created between particle pairs. The resulting elastic structure, similar to what is known as ragdolls in the videogame industry, is able to track objects while trying to preserve its original shape.

5.1 Introduction

There exists a plethora of representations and tools that can be applied to images depending on the requirements of currently considered purposes. While descriptions of proximal objects as a whole have been extensively used (Lowe, 1999) (Viola and Jones, 2001), current tendencies consider objects as a collection of parts (Shotton et al., 2008) (Leibe et al., 2008). How structural relations between parts are defined depends on the approach: there may exist no explicit links at all (free roaming particles), a centroid can be used as a hub or different graph-like structures can be used to link particles.

In this context, a constrained particle system that simulates an articulated rigid body which dynamics are ruled by an underlying visual process is presented.

Particle kinematics are computed using a velocity-less Verlet integration scheme, thanks to which the constraint system becomes solvable in a stable way with a very simple and fast approach. The main contribution of this work consists on the application of this method, which has been widely used in the computer games industry, to computer vision problems, controlling particle dynamics through the visual analysis of images.

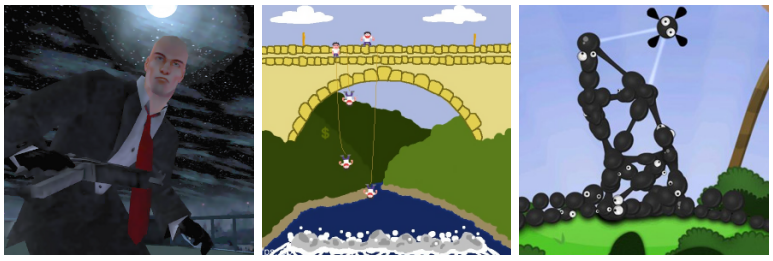
There exists a direct relation between the proposed approach and swarming methods (PSO, ACO...). These biologically inspired techniques maintain a population of simple agents which activity allows the emergence of complex behaviors from local interactions between agents, creating decentralized and self-organized systems. In the same way, the proposed approach uses a collection of simple particles which group dynamics arise from the satisfaction of local constraints between two given individuals, although their individual behavior is completely independent from other group members.

The proposed particle dynamics system is applied to a distributed tracking solution. A team of relatively simple trackers contribute to following an object through a video sequence while a sensible updating mechanism based on context allows the actual object appearance to be registered while minimizing drifting. The elastic structure that is created by the proposed approach is able to follow objects as they traverse a video sequence, correcting misplaced particles as it tries to preserve geometry.

5.2 Previous Work

Constrained particle dynamic simulations are techniques tightly related to the computer games industry. The present approach is adapted directly from the work of Jakobsen (Jakobsen, 2001), who designed the core of a physics system for computer games focusing on stability and speed of execution. Defining an articulated body with solids (particles) and joints (constraints) in a structure that is similar to a graph, Jakobsen proposed a strikingly simple method to simulate its dynamics, based on a velocity-less Verlet integration scheme. Games using this

system are able to handle the simulation of hundreds of rigid bodies in real time, from pieces of cloth to plants and corpses, and many 2D and 3D physics engines already incorporate a similar solution.



(a) (b) (c)

Figure 5.1: The adopted system is very versatile: a) Many elements in IO Interactive's *Hitman: Codename 47*, from main character's tie to victim's corpses, use the adopted Ragdoll physics system, which Jakobsen developed explicitly for the game. b) In *Bungee Manager* (a prototype created by the author of this work), jumping ropes also use a Ragdoll solution. c) In 2DBoy's *World of Goo*, the user is able to create complex structures placing particles wherever she wants, fighting against gravity to achieve a certain goal.

The proposed particle structure is similar to kinematic chains, which are well known structures in systems where shapes must be preserved or inferred. In (Deutscher et al., 2000) human bodies are tracked following limb movement independently, and predefined kinematic chains are used to shape human models, limiting the dimensionality of the tracking problem. In (Mori and Malik, 2006) kinematic chains are used to recover 3D body configurations using shape contexts, matching with a limited set of predefined poses computing best translations for sample points using least squares. In (Yan and Pollefeys, 2006), feature trajectories are used to reconstruct a kinematic chain, allowing the system to infer the structure of the observed object. A different particle-constraint structure is defined in (Masson et al., 2005), where a flexible grid is used to track 3D objects. An elastic graph which nodes contain visual features computed using Gabor wavelet transforms is used in (Wiskott et al., 1999) to recognize human faces in frontal and profile views. In (Grasp and Taylor, 2000), kinematic chains are used to recover information about the configuration of an articulated object, such as a human figure, from point correspondences in a single image (see Figure

5.2).

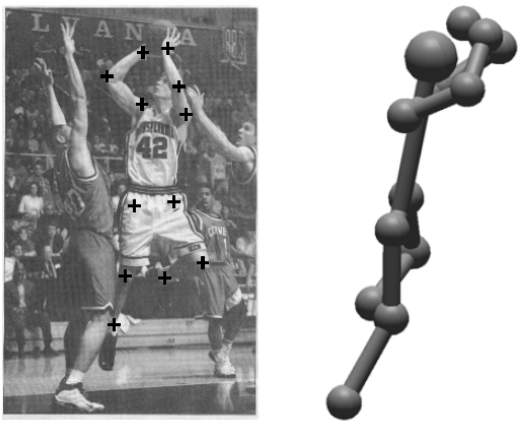


Figure 5.2: The image on the left represents an image containing a figure to be recovered, where joint locations have been located by hand. The image on the left shows the recovered 3D model from a novel vantage point (Grasp and Taylor, 2000).

Links between particles are not always explicit. In (Leibe et al., 2008) planar patches are extracted from multiple views of a given object and grouped into clusters with similar features. As each patch stores its possible relative positions in relation to the known centroid of the object, clusters are able to vote for different probable object locations. A point in space-scale with enough votes becomes a detection hypothesis, linking all the clusters that voted for it in the same structure. This method is explicitly designed for object detection, so no kinematics are considered.

Active Appearance Models (AAM) (Cootes et al., 1998) use a point distribution model to combine shape and gray-level variation in a single statistical appearance model. A training set of labeled images is required, where landmark points are marked on each example at key positions to outline the main features. For each labeled key point, its mean position and variance is computed from all available training images. Thus, a correct labeling is extremely important. Key points must be place correctly in each training image, always representing the same point in the modeled structure. Once the statistical shape information is gathered, a mean shape can be computed, from which new shapes can be generated costlessly (Cootes et al., 1995). Because each key point also gathers

appearance information, AAM become highly flexible deformable models that are able to represent any object within bounds of the training set. Figure 5.3 shows an example.

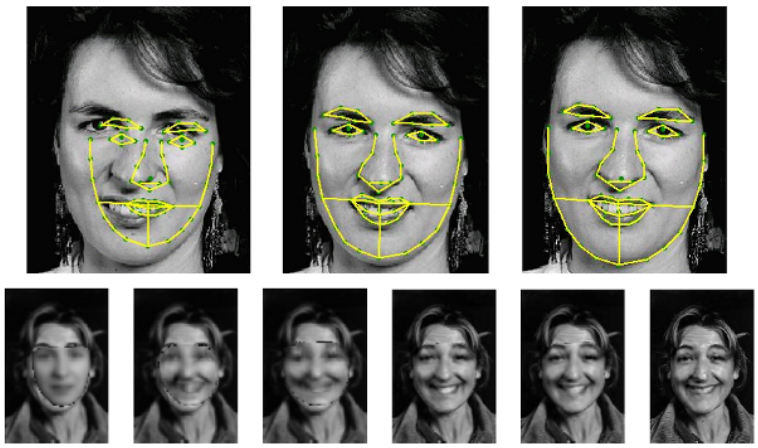


Figure 5.3: The three images on the top row show how an AAM shape model adapts to the underlying image iteratively. The bottom row shows how the appearance model converges from an average appearance.

5.3 Articulated Rigid Body Dynamics

Particle dynamics can be used to emulate complex systems accurately, like the structure of a certain virus (Freddolino et al., 2006) or the folding properties of proteins (Hinrichs and Pande, 2007). But they are also used to achieve believability, stability and speed of execution in animations (Capell et al., 2002) and interactive software (Murray et al., 2004). The proposed application of these particle systems to computer vision tasks requires high performance, but precise accuracy and realism are not primary concerns. The graph-like structure used to represent a visual entity will be controlled by a Verlet integration scheme and a simple constraint solver, as defined by Jakobsen in (Jakobsen, 2001).

5.3.1 Sentient Ragdoll Definition

A Sentient Ragdoll $\mathbb{A} = \{P, C\}$ is defined as an articulated rigid body whose evolution is driven by image analysis related processes. As such, it can be understood as a ragdoll that is able to perceive information extracted from images and video sequences and react according to a certain goal.

A Sentient Ragdoll is composed by a set of particles P and a set of constraints C between particle pairs.

Each particle $p, p \in P$ is defined by the following 4-tuple:
$p =< \vec{x}_p, \vec{x}^*_p, m_p, \vec{a}_p, V_p >$, being:

1. $\vec{x}_p$ a particle's current position in the image space.
2. $\vec{x}^*_p$ a particle's position in the image space at previous time step.
3. $\vec{a}_p$ a particle's acceleration.
4. m_p is the particle's mass.
5. V_p is the particle's Visual process, and it comprehends all image features and methods needed for achieving the visual task at hand. It will be explained in detail later.

Constraints $c, c \in C$ are defined by a 3-tuple: $c =< p, q, d >, p, q \in P$, meaning that particles p and q should rest at a distance d of each other. The present approach only considers distance constraints, although adding any other type (i.e. angular or scale constraints) is straightforward.

Ragdoll systems are simulated following the next steps:

- Accumulate Forces: Each particle acceleration is updated with current existing forces (gravity, wind...), if any.
- Kinematics: Each particle updates its velocity and position using a Verlet scheme.

- External constraints: Each particle position suffers a projection as a result of existing external constraints, if any (e.g. world limits, user interaction, collisions with other objects...)
- Constraint solver: Internal constraints (links between particles) are satisfied using relaxation.

The following sections explain each step in detail.

5.3.2 Kinematics

A first approach to particle dynamics simulation is the Euler integration method. Each particle motion is usually defined by two variables, its position x and velocity v, which are updated in each time step using the following equations:

$$x' = x + v \cdot \triangle\ t$$
$$v' = v + a \cdot \triangle\ t$$

where $\triangle\ t$ is the time step and a is the acceleration computed using Newton's second law $F = m \cdot a$, being F the accumulated forces acting on the particle and m its mass.

The physics system proposed in (Jakobsen, 2001) uses a velocity-less Verlet integration scheme (Verlet, 1967). Given a particle $p, p \in P$, particle velocities are implicitly computed from the current x_p and previous x_p^* positions, so the new value is given by:

$$x'_p = 2 \cdot x_p - x_p^* + a_p \cdot \triangle\ t^2 \tag{5.1}$$

which behaves similarly to Euler integration as $x_p - x_p^*$ is an approximation of the current velocity of a particle. In such a system, particle velocities are updated by forces (and thus accelerations), or simply moving particles to a given position (projection). Although it is not very accurate, as energy might dissipate, it is fast and stable, because velocity inaccuracies are not accumulated.

5.3.3 Constraint Solver

A common approach to defining an articulated rigid body consists of a system of interconnected springs and particles which dynamics are computed deriving Hook's law. However, if the number of particles and constraints grows, it is not always trivial to solve the corresponding system of differential equations.

In the adopted physics system, links are considered infinite stiffness springs, i.e. sticks, and constraints are solved using projection. Misplaced particles violating constraints are simply projected to a position where the considered constraint is satisfied. If two particles are too close they are pushed apart, and vice versa. One advantage of the Verlet scheme is that changes in velocity as particles are relocated to satisfy constraints are handled automatically. Because velocity is introduced implicitly through subtraction of consecutive positions, stability is preserved.

If particle weights (masses) are considered, the heavier the particle the weaker the suffered correction. If, for some reason, a certain particle should not move, being assigned an infinite mass will make it immovable. A zero mass, on the other hand, will force a particle to follow the rest of the system without affecting its dynamics (see Fig 5.4).

Given a distance constraint $c =< p, q, d >$ between two particles p, q at a distance d, with particle masses m_p and m_q, the new projected position x' for each particle is given by:

$$\begin{aligned} \vec{\delta} &= \vec{x}_q - \vec{x}_p \\ \epsilon &= \frac{|\vec{\delta}| - d}{1.0 + |\vec{\delta}| \cdot (m_p^{-1} + m_q^{-1})} \\ \vec{x'}_p &= \vec{x}_p \vec{m}_p^{-1} \cdot \delta \cdot \epsilon \\ \vec{x'}_q &= \vec{x}_q \vec{m}_q^{-1} \cdot \delta \cdot \epsilon \end{aligned} \tag{5.2}$$

When a constraint is satisfied its two particles are reallocated, which may

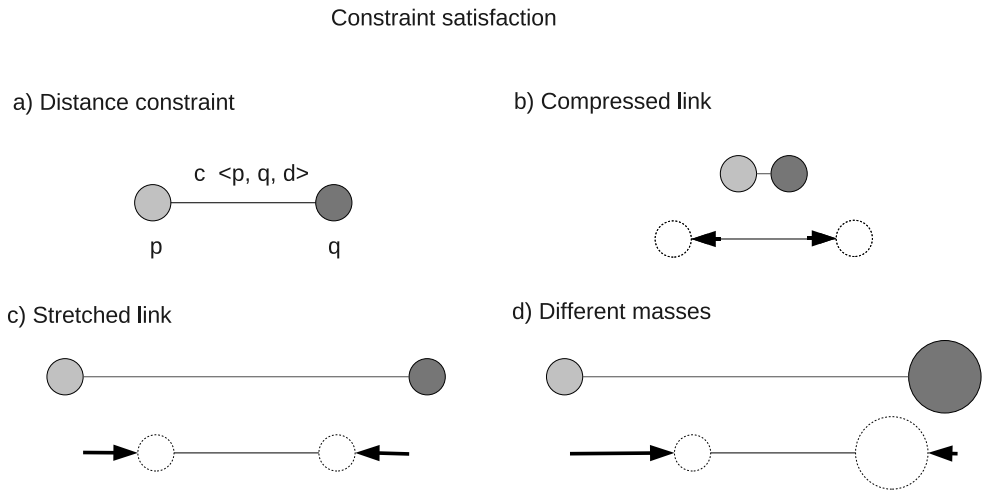
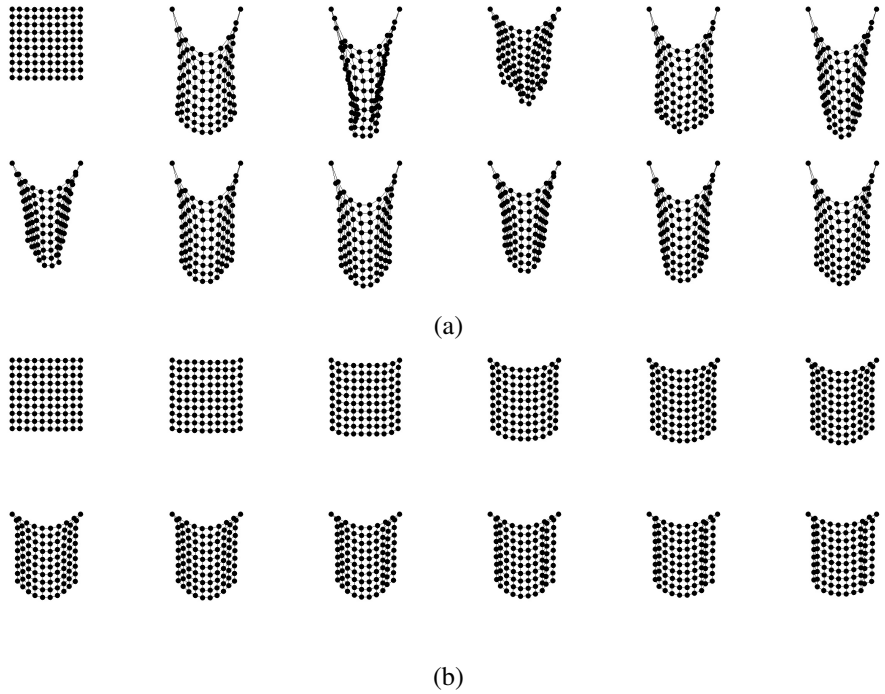


Figure 5.4: Local constraints can be consecutively satisfied in an iterative process. A distance constraint c is defined between two particles p and q (a). If a link between two particles with the same mass is stressed, they are equally projected to their ideal position (b, c). Heavier particles suffer a smaller correction (d). This process is repeated for every constraint iteratively, a fixed number of iterations.

result in a different constraint being violated. Using relaxation (Press et al., 2002), constraints can be consecutively satisfied in an iterative process. For a given number of iterations, each constraint is solved using expression 5.2. While constraints may not be satisfied in a single pass, the Verlet integration scheme allows the system to converge to the correct state over consecutive time steps, correcting positions incrementally.

This incremental correction allows relaxation to be stopped prematurely. Actually, the number of iterations establishes the elasticity of the whole system. Figure 5.5 shows a particle mesh which internal constraints are satisfied using one and fifty iterations between consecutive frames. A single iteration creates a flexible structure, while an increasing number of relaxation iterations produce a stiffer body.



(a)

(b)

Figure 5.5: A particle mesh that represents a piece of cloth is defined in an environment where gravity is present. Two extremes of the cloth are *nailed* to the background (their mass is infinite), but the rest of them evolve following the Verlet integration scheme, converging to the correct state over consecutive time steps. A higher number of iterations in the relaxation process produce stiffer bodies that also need a lower number of time steps to converge.

5.3.4 External constraints

Links between particles are considered internal constraints, and they define the structure of an articulated rigid body. But any particle may be affected by forces related to the environment where the structure is defined. In computer games and simulations, these forces may be produced by a collision with an static object, a link with another object, gravity, wind or through user interaction, to name a few examples.

Some of these forces may be introduced applying Newton's second law ($F = m \cdot a$), updating accelerations in the Verlet scheme. Gravity or Wind, for example, are easily defined by forces. But, in some cases, it may be more convenient just projecting the affected particle to the desired position (for

example, to prevent a particle from going through a wall), exactly as the constraint solver does when correcting stressed links.

Processes not related to the structure of the articulated rigid body which affect the position of one or many particles are considered external constraints. Affected particles are simply projected to the designed location or assigned a certain acceleration, and internal constraints drag the rest of the body accordingly.

It is precisely by external constraints, defined by visual processes, how an articulated rigid body will interact with images.

5.4 Application to Computer Vision

The current particle system was originally designed to be applied to computer games, where virtual objects react believably to interactions with the user and other virtual objects. As this virtual worlds usually mimic our own reality, forces acting on objects include gravity, wind and, given the violent nature of some games, projectile impacts and explosions.

In our approach these structures will traverse the projected space created by a video sequence, evolving from frame to frame according to a certain visual purpose. Therefore, dynamics will be ruled by the underlying visual process, conveniently updating particle masses (weights) and accelerations.

In Section 5.3.1 particles in a Sentient Ragdoll $\mathbb{A}$ were defined using a 4-tuple: $p =< \vec{x}_p, \vec{x}^*_p, m_p, \vec{a}_p, V_p >, p \in P$, where V_p represented the particle's visual process. In this context, V_p may create an acceleration or project the particle to the most suitable point according to a certain visual task.

The layout of particles and constraints created between them also depend on the purpose of the visual system. Particles can be placed on the visual space randomly, regularly (e.g. in a grid) or using salient point detectors, while constraints can be defined linking particles randomly, using the kth closest neighbors, using a single particle as a hub to which the rest of particles are linked to or building a Delaunay triangulation. The best configuration only depends on

the task at hand, even allowing dynamic structures that redefine their links and weights at every time step.

5.4.1 Tracking

Tracking systems generally store a representation of the tracked object, defining those features that will allow locating the object as it evolves through time, space and appearance. Different and increasingly more complex representations have been considered in literature: image patches, histograms of gray values, color or gradient orientations, texture descriptors, collections of local features... Not many approaches, however, consider that the appearance of a visual object may vary through time. While visual changes are common in a general tracking scheme, most solutions are designed to ignore this problem or a naive updating mechanism is used, constantly updating the descriptor. This leads to the issue of drifting and the loss of the tracked object.

However, it is also true that updating the tracked pattern may introduce false representations. Updated patterns may include pieces of background or strange objects, which usually leads to losing the tracked object. Updates must occur scarcely, only when they are strictly necessary. However, not even a sensible context-based updating mechanism like the adopted one, explained later, assures perfect updates.

Tracking solutions have commonly used a centralized logic approach. Objects are considered single entities which position is defined by the outcome of a feature matching process. If part-based descriptions are used, probability maps are created considering each part's matching weight, and a mode-seeking process is used to locate the most probable position of the object (Adam et al., 2006). In both cases the tracking process continues from a single inferred location. If this position is wrongly estimated, the tracked object may be lost.

A different approach may consider tracking as the outcome of a team of self-reliant trackers, allowing self-organization through local interactions, as proposed in the previous chapter. Different mode-seeking methods may be used to locate

the object, but each tracker is still an independent process. Information from the group can be used to correct individual behavior, so the tracking task becomes more robust to isolated mismatches. The present solution belongs to this latter group.

Modelling the tracked object using an articulated rigid body constitutes a robust approach to part-based tracking. Each particle tracks a piece of the considered object, so the system is able to cope with local deformations. Due to the elastic nature of articulated bodies, particles which lose their target can be relocated when constraints are satisfied, so tracking becomes more robust to occasional errors. This elasticity also allows the shape of the structure to be recovered, even after strong deformations (see Figure 5.6 and seen in Figure 5.7).

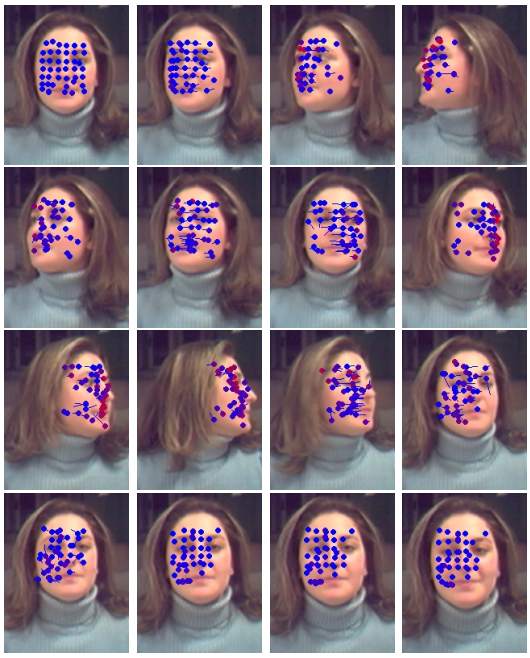


Figure 5.6: Most free-roaming particles manage to track their target properly thanks to their view bank, but some of them drift away from their ideal position.

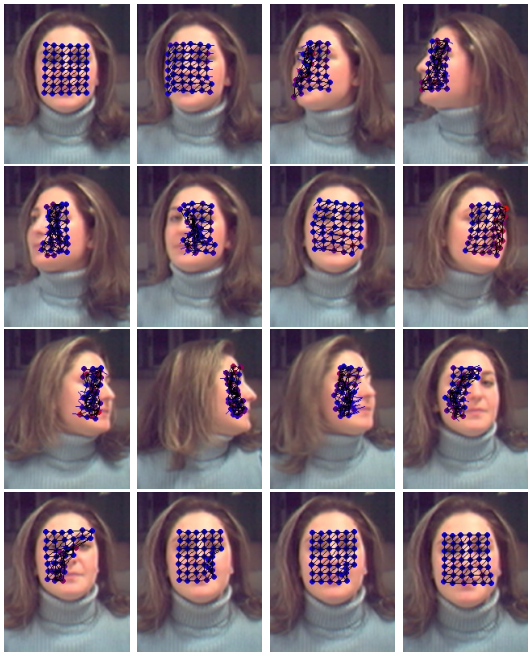


Figure 5.7: The sentient ragdoll tracking solution manages to recover its original shape after two out of plane rotations, successfully dealing with partial occlusions while updating tracked patterns.

5.4.2 Pattern matching and updating

In this context, particles may use any plausible tracking method found in literature. However, they do not have to share the same method: different approaches can be used depending, for example, on the location of the particle within the structure. Each particle's tracking method should return a single point where the particle should be projected to, defining an External constraint for the Sentient Ragdoll.

For demonstration purposes, each particle will use a relatively simple kernel tracking method with context-based updating, similar to the one proposed in (Guerra et al., 2005). This method is explained in detail in the following sections for self-containing purposes, but also to comment some improvements.

Pattern matching

Tracked targets will be stored by a particle's visual process, V_p, as color planar templates. Tracking will consist on finding their most probable position within a search window around the last known location, using a similarity function which optimum is found through exhaustive search.

Targets are found sliding templates through search windows. Resemblance between overlapped images is measured using a sum of squared differences (SSD), although any other measured could be used (Hagedoorn, 2000). SSD creates a number of local minima at best matching positions, as shown in Figure 5.8. Two local search window extrema are considered: the first one, the global minimum, dictates the new position where the particle must be placed in order to track its target; the second one, the second best local minimum, is used in the updating policy as a measurement of the presence of similar objects in the vicinity, as seen later.

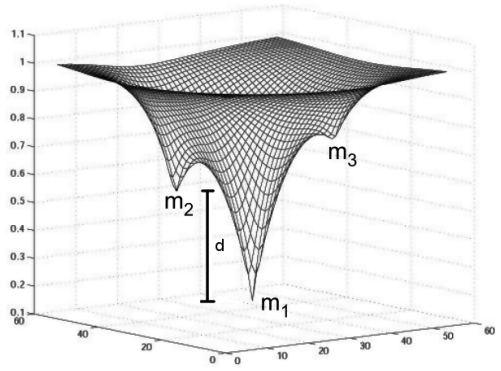


Figure 5.8: Surface created sliding a template through a search window. Surface values correspond to the similitude measure between the template and the overlapped region. The figure shows a global minimum at m_1, which correspond to the best matching position, and two local minima at m_2 and m_3, which represent context similitude. Distance d between m_1 and m_2 will be used by the template updating policy (Guerra et al., 2005).

Patch sizes may vary in order to represent objects at different scales. However, when patches are stored in V_p as templates, they are resampled to a normalized size, although the original patch size is also kept. When a template

is used for matching, a search region is defined in the current image around a particle's last known position. Its size is proportional to the size of the original patch represented by the template. Using the same proportion factor, a search window is created from the template size. During tracking, the search region is resized to this normalized search window, where the template convolution takes place. Figure 5.9 depicts this resizing process with two different-sized patches. This mechanism allows searching at different scales in constant time, which was not considered in the original work (Guerra et al., 2005).

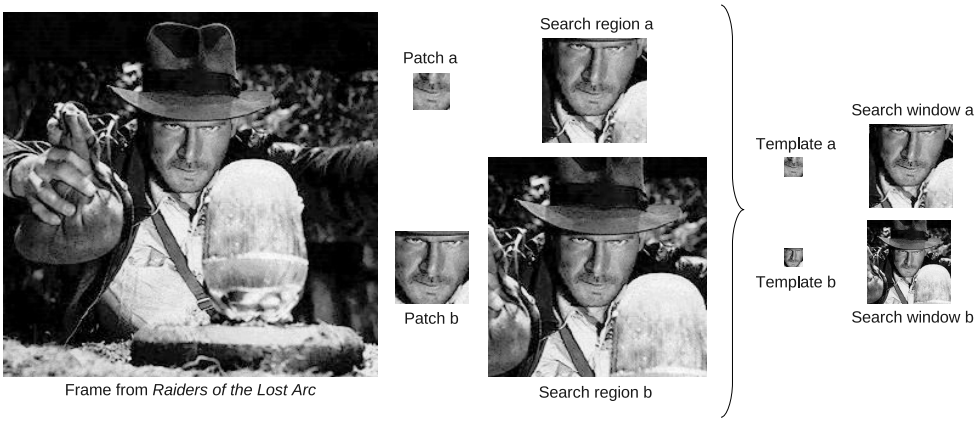


Figure 5.9: Image patches and search regions are resized to a fixed template and search window size. Thus, searches are performed in constant time, no matter the original patch scale.

Normalized template size and proportion factor for the search window should be chosen carefully. If templates are bigger than patches, they will contain redundant information from upsampled patches. If they are too small, many details from the original patch may be lost after downsampling. Smaller templates are, however, computationally cheaper.

Search region and search window sizes affect the ability of particles to follow fast moving objects. Smaller sizes are only suitable for tracking slow moving object if the image capture device frequency is not high enough, while bigger sizes allow particles to keep up with faster targets. For most tested cases, a template size of $11x11\ pixels$ and a search window three times bigger (the same proportion as shown in Figure 5.9) yields good results.

Updating using context

Visual changes in the tracked object should be expected as it traverses the video sequence. Many approaches, like (Adam et al., 2006), propose tracking solutions that simply ignore the updating issue, even when it seems obvious that keeping a constant pattern is not a solution in a changing environment (unless this constant pattern includes a nourished set of appearances of a given object).

However, a wrong updating policy will derive in the drifting issue: templates may be altered until they no longer represent a view from the tracked object. An updating policy may consider updating templates constantly (every frame), periodically (after certain number of elapsed frames) or using a threshold to detect changes. See (Guerra et al., 2005) for an detailed analysis of updating mechanisms.

Constant updates may accumulate small changes in the stored pattern, due to the discrete nature of digital images. If the pattern is periodically updated, irrelevant updates may occur, or even worse, critical updates may be missed. Using a constant threshold to update only when a certain amount of change is detected introduces the problem of choosing the right value. A low threshold is similar to constant updating, while a high threshold may miss critical changes.

Updating according to context seems a reasonable solution. A ratio can be computed between the extrema and the average value in the search window. If this ratio is high enough, meaning that the current view differs, updating takes place. But again, how to choose if this ratio is high enough? The current solution uses contextual information found in the search window in order to define a dynamic threshold, as proposed by (Guerra et al., 2005).

As stated before, the second minimum in the search window is also considered. This extremum is a measurement of the presence of patches that are similar to the one being tracked, and it is used to decide when a template should update. The context-based updating mechanism consists of the following two steps:

1. A threshold τ is initially computed as *half the distance between the two best*

minima in the search window, $\tau = (m_2 - m_1) \cdot 0.5$

2. In each frame, after template convolution takes place, the two best local minima m_1 and m_2 in the search window are found. A template will be updated on two occasions: 1) the global minimum m_1 is above the current threshold τ, meaning that the object appearance has changed significantly or 2) the second minimum m_2 is below the current threshold τ, meaning that there exists an object in the search window which is too similar to the one being tracked. As the patch is updated with the overlapped image, τ is also recomputed as $\tau = (m_2 - m_1) \cdot 0.5$

The updating mechanism used in our approach allows each particle to keep track of changes in the visual appearance of the tracked object. However, the current system is strictly pre-categorical, so there still exists the possibility of including views that do not belong to the tracked object. This may happen if, for example, a view belongs to the boundary of the object and contains background information. If the background is static and the appearance of the object changes (i.e. the object rotates or suffers motion blur), the tracking and updating mechanism will find a better match in the unchanged background scene.

This way, particles following parts located in the boundary of the tracked object may change their attention to pieces of background. While in most situations these particles will be forced to follow the group due to internal constraints, as seen later, their influence may be too strong under certain circumstances, sticking themselves to a patch in the background and dragging the rest of the structure with them, losing the object entirely. This usually happens if the tracked object does not contain good features to track or it if is too small. The use of a view bank alleviates this problem.

5.4.3 Managing views

As previously mentioned, V_p, a particle's visual process, contains not a single template but a bank of color planar templates, which represents a short-time visual memory of the most probable appearances of the object being tracked. While

tracking, each template in the view bank of a particle is matched against the search window. The one with the best matching cost is considered the current view for that particle.

As an object traverses the distal space, its appearance will usually change smoothly. This is represented in the proximal space as a set of views that will be revisited periodically. Storing these representations in a bank of views allows the system to reuse previously seen appearances of the proximal object, reducing the need for updating and thus drifting. These views also represent the visual knowledge that a precategorical vision system can build from the observed object autonomously.

Due to the finite nature of computer memory and processing time, this visual memory must be limited. For tracking purposes, only a small number of significative views will be stored, creating a short-term memory of recently seen and distinctive appearances. Fortunately, the use of previously explained context-based threshold also fulfills this objective. Because views are updated only if they have changed significantly or if a neighboring object is too similar, the singularity of each view in the visual bank is guaranteed up to a certain degree, as seen in Figure 5.10.

When a view bank is full and a new update takes place, an old view must be discarded. Choosing which one depends on when it was used for the last time (obsolescence) and how much it has been used (persistence). The most useful views are those with high persistence and low obsolescence, that is, views that have been recently used or views that have been used for longer periods. If each view stores these two values, a measure of $utility$ can be defined using the following expression:

$$utility = persistence/(1.0 + obsolescence) \tag{5.3}$$

When working with video sequences, it makes sense measuring time in elapsed $frames$. This way, each view will track $persistence$ as tp, or the number of frames that it has been used, and $obsolescence$ as to, or the last frame when

this view was used. Then, being now the current frame, $utility$ is defined as:

$$utility = tp/(1.0 + now - to) \tag{5.4}$$

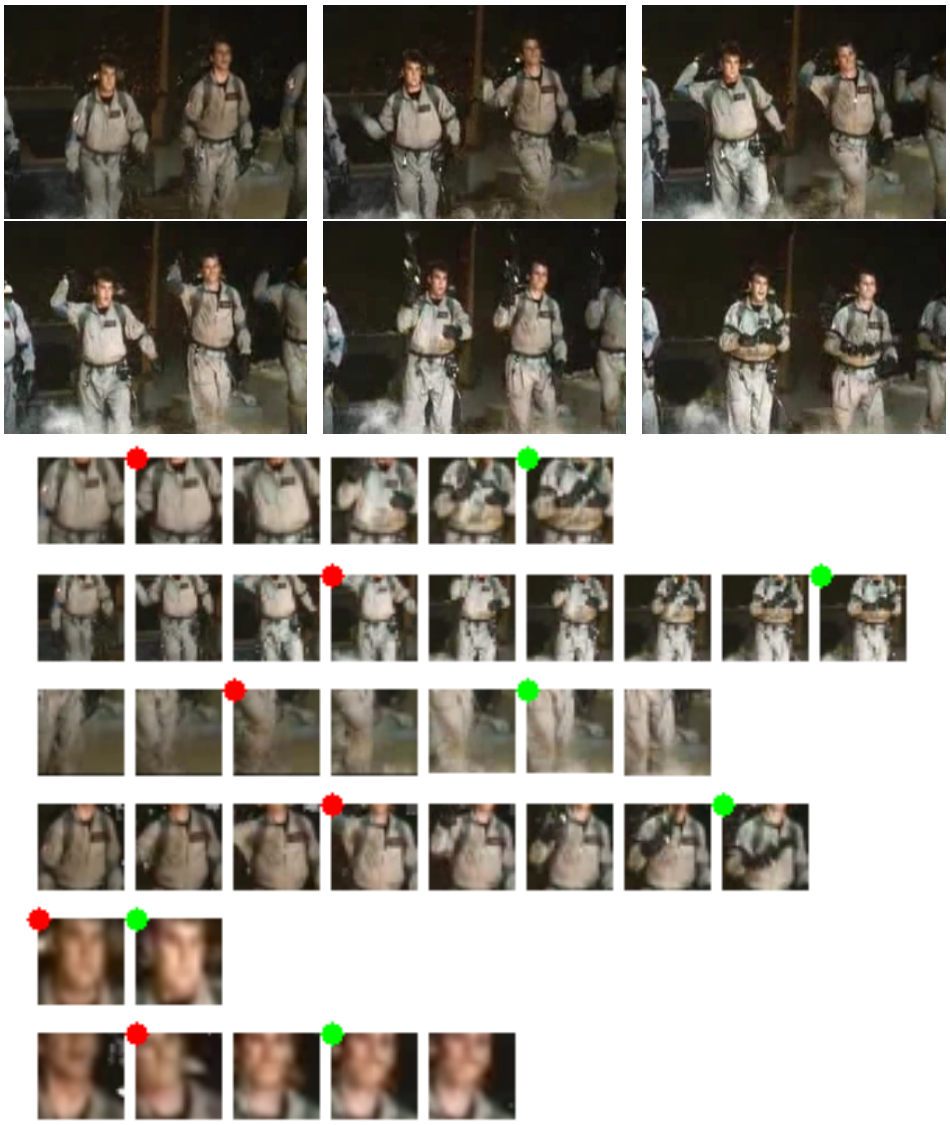
Once the $utility$ of each view is computed, the one with the lowest value is discarded. This way, the bank of views will always contain those patches that have been used recently and for longer times.

Figure 5.10 shows how six different particles update they view banks while tracking their targets in a 42-frames sequence. For demonstration purposes view banks capacities are not restricted, so they contain all considered updates. Note that, while the second particle needs up to nine updates, the fifth one only updates twice during the sequence. Currently used views are marked with a green circle (the red one marks the worst view, the one with highest matching score, which is usually remarkably different from the best view). Note that the current view for each particle is not always the last view in the bank (the bank grows from left to right), and that the *worst view* in a bank (marked with red circles in Figure 5.10) does not have to coincide with the discarded one. This decision depends strictly on the utility value of each view.

Canonic views

View banks in (Guerra et al., 2005) represent a short-term visual memory of recently seen template appearances, ideally allowing a system to overcome appearance changes. Tracking solutions that do not update their templates may lose their target if its appearance changes. But, on the other hand, these solutions are certain that only information that belongs exclusively to the tracked object is being used. In (Guerra et al., 2005), even its sensible updating policy, explained in previous sections, may not avoid capturing an image patch that does not belong to the tracked object. The tracker may stick to a piece of background and lose its target due to a variety of reasons: occlusions, rotations, drastic appearance changes...

The present work adopts a mixed solution. View banks will store a limited



(a)

Figure 5.10: Six sample frames from a 42-frames sequence from Ivan Reitman's *Ghostbusters* are shown. Below them, each row of pictures represents the state of the view bank of a tracking particle at the last frame of the sequence. Because updates depend on context, updating frequency is different for each particle. Green circles mark the currently used view for each particle, while red circles show the worst matching view.

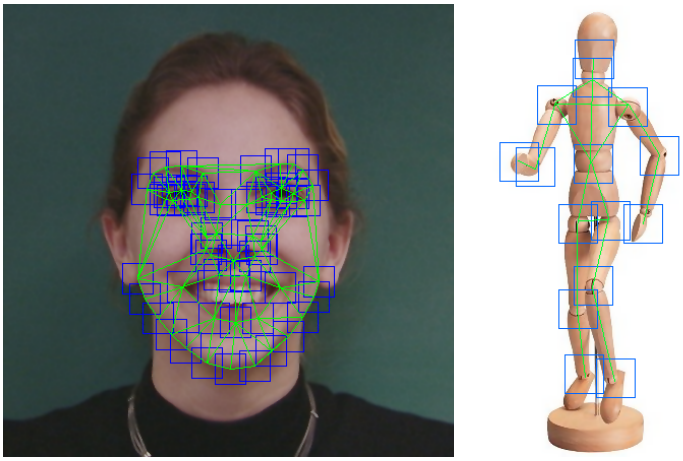
number of templates, but one of them will be an unchangeable view, called canonic, from those created on the initial tracking frame. This first frame should

show a representative pose of the tracked object (i.e. a frontal view of a face). Views captured from this initial image are supposed to reappear at some point. They represent a reliable model to go back while the rest of views are updated.

Finding an updating policy that guarantees that a recently updated view belongs to the object is not trivial. The tracker may have lost its target, updating its templates with pieces of background or patches from strange objects. If adverse conditions are met, all object patches in a view bank may be replaced with strange views, and the tracker will not be able to recall how its target looked like. A canonic view may allow the tracker to recover a certain object appearance. However, under changing light conditions, canonic views may become useless.

5.4.4 Particle layout

Although each particle follows its target independently as described in the previous section, they are all part of a bigger structure. Sentient Ragdolls are shaped creating constraints between particle pairs, so specific structures can be built mimicking the shape of the target object, as shown in Figure 5.11.



(a) (b)

Figure 5.11: Articulated rigid bodies created with the shape of a known object. a) Face structure created from annotated face points as defined by (Cootes et al., 1998). b) Simple human-like skeleton model.

With enough hand-marked samples, such structures could also contain averaged templates for each possible appearance of each particle in their respective view banks, similarly to elastic bunch graphs in (Wiskott et al., 1999).

However, the present approach deals with precathecoric tracking, so no specific shape nor appearance will be defined a priori. Instead, two mechanisms are considered to place particles on the target object autonomously: the first one places particles regularly on the vertices of a grid; the second one places particles at salient point locations, with and without scale.

Links between particles can be defined automatic and consistently using the edges created by the Delaunay triangulation of allocated particles. The Delaunay triangulation of a set of points lying on a plane creates a planar graph that subdivides the plane into triangles, maximizing their minimum angle (Delaunay, 1934) and satisfying, among others, the following properties:

- The exterior face of the triangulation is the convex hull of the point set.
- Each vertex has on average six surrounding triangles.
- The circumcircle of any triangle in the Delaunay triangulation does not contain any other vertex in its interior.

These properties assure well-shaped meshes that are suitable to create structures which are robust against deformations.

The following layouts are considered:

- Simple grid: particles are placed regularly over the object to be tracked, defining a grid. Each particle is linked to its four neighbors in the grid, if they exist.
- Delaunay grid: particles are also placed on a grid, but constraints are created using a Delaunay triangulation. The structure is similar to the Simple grid, but more robust.
- Rigid grid: particles are also placed on a grid, but each particle is linked to its eight neighbors.

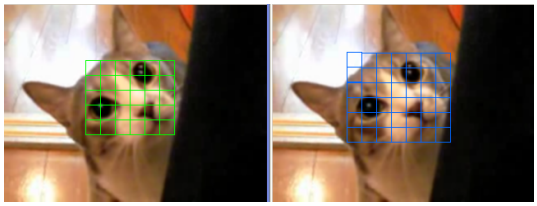


Figure 5.12: Simple grid with non-overlapping 19x19 patches.

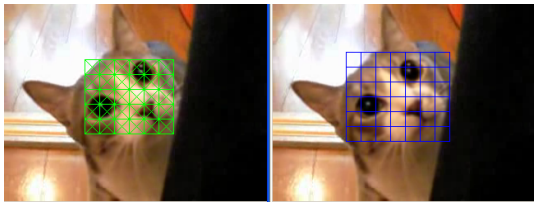


Figure 5.13: Rigid grid with non-overlapping 19x19 patches.

- KLT points: particles are placed on salient points defined by KLT features (which are, ideally, *good features to track* (Shi and Tomasi, 1994)). Constraints are also created from the resulting Delaunay triangulation.

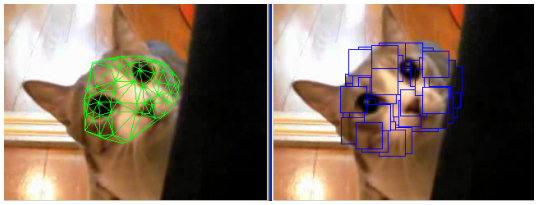


Figure 5.14: Delaunay triangulation from KLT points, using 19x19 patches.

- Scale salient points: salient points are computed using the SURF scale point detector (Bay et al., 2006). Thus, particles will follow parts of the object at different scales, like those shown in Figure 5.15. Once again, particles are also linked using the edges created by a Delaunay triangulation.

Grid structures are much less rigid than Delaunay-based meshes, so they allow stronger deformations and thus they grant particles more freedom to achieve their goals. However, because of their regularity, grid structures are prone to fold onto themselves until they look like a single grid cell (see Figure 5.16).

Folding is almost inevitable if the grid structure is ruled by distance

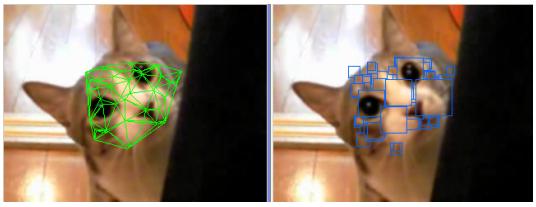


Figure 5.15: Delaunay triangulation from SURF scale salient points.

constraints, although it could be alleviated by using angular constraints instead, and it is tightly related to the updating mechanism. A sentient ragdoll which particles do not update their view bank will not show this folding tendency. Therefore, canonic views are also useful to prevent this effect.

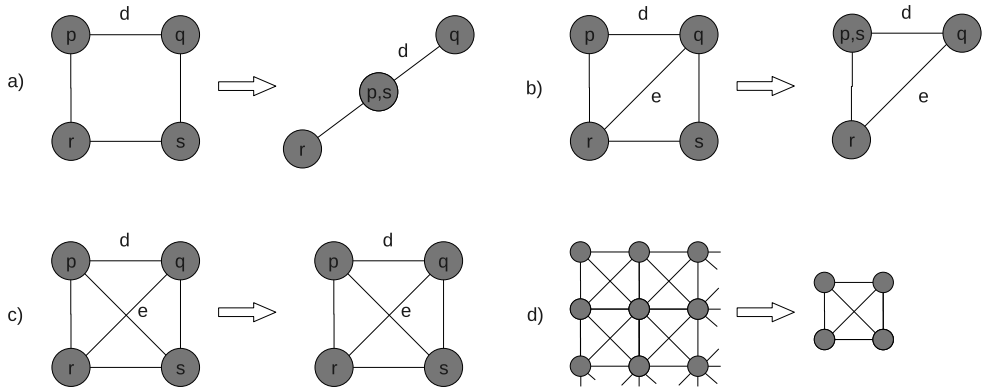


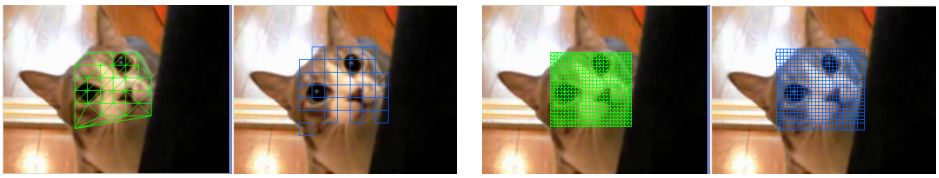
Figure 5.16: Basic grid cells like a) and b) allow certain configurations that do not violate constraints, but where the original structure is lost. Using two diagonal links, like in c), the basic cell may not fold onto itself. However, the main problem with grid-like structures when distance constraints are used is that they may still fold until they look like a basic cell, as shown in d).

While KLT and center-surround salient points contain interesting features by definition, particles in a grid may be created on uninformative image regions. Homogeneus and plain features are difficult to track: a particle following an homogeneous patch in a region with similar features (i.e. a particle following a white patch in a white wall) will not be able to find a distinctive extremum to follow. These patches can be rejected when creating a grid layout and even during tracking, if their color variance is lower than a threshold (see Figure 5.17 a). However, homogeneous regions may also correspond to blob-like structures,

so it is not always safe removing them.

The size of particle patches created from scale salient point detectors like SURF is defined by the scale at which each point is detected, which is optimum by definition. If this information is not available, particle patch sizes should scale proportionally to the size of the tracked object.

Grid layouts also require defining grid densities, i.e. distances between neighbor particles. Low distances produce dense grids where particle patches may overlap (see Figure 5.17 b), while large distances produce sparse grids where particles may not properly cover the tracked object, if they are not big enough.



(a) Delaunay grid with sparse, non-overlapping, 19x19 patches.

(b) Dense Delaunay grid with overlapping 11x11 patches.

Figure 5.17: Other possible grid configurations.

5.4.5 Tracking quality and particle weights

Particles may vary their mass as the articulated body evolves. As shown in Figure 5.4, heavier particles will pull from lighter particles. In this tracking context, particles correctly following their target should ideally pull lost particles, correcting their position. Thus, particle masses should be related to the quality of the tracking process, being assigned heavier masses if they are properly tracking their target.

Measuring tracking quality is not trivial for a system that updates its templates. If no updates are considered, a tracker may simply measure differences between its template and the matched image. But if templates are updated, no updating policy guarantees that a recently updated view belongs to the object. The tracker may have lost its target at some point, updating its template with a piece

of background or a patch from an occluding object. A recently updated template may have a good matching cost, even it the matched view does not belong to the tracked object, sticking the tracker to an unwanted image patch.

Nevertheless, the tracking activity, as explained in Section 5.4.2, produces some measures for each view in a view bank that can be used to evaluate tracking quality: the two best extremum values in the search window (see Figure 5.8) and a view's utility. The first extremum is a direct measure of how a view matches the underlying image. Utility of a view informs about a the obsolescence and persistence of a particle. Different weighting policies derived from these values are proposed and discussed:

- Matching quality: Particles are assigned a weight obtained from the first minimum in the search window. This way, particles which current view finds a good match in the search window are given heavier masses. However, as previously stated, good matches do not mean higher tracking quality, because matched views may correspond to a wrongly updated template. Only matching a canonic view can minimally assure tracking, as it certainly belongs to the original object.

- Utility: Reliability of a view is given by its utility. A view with high persistence and low obsolescence is ideally a useful view. However, a lost particle may also build up high utility values as it sticks to its wrong target.

- Matching and utility combined: Tracking can be evaluated combining matching cost and utility. Ideally, a good tracking activity should produce good matching values with high utility views. A perfect matching with a low utility value can be produced by a recently updated view or and obsolete view. In both cases that view is not reliable, and the matching cost is uninformative. Particles could be assigned higher weights if their matching cost is good and their utility high.

- Worst matching analysis: The view with the worst matching value in a view bank is the one that differs most from the matched position, and thus from the currently used view (see how worst views, marked with a red circle in

Figure 5.10, are different from current views, marked with green circles). This worst matching value will be similar to the best matching value if both views look similar. A high worst matching value is an evidence of the bank containing drastically different views.

However, it is difficult to tell apart whether the particle is properly tracking its target, but its view bank contains a strange view, or if the particle has wrongly updated its current template and it is tracking a strange view.

A safe solution consists on assigning heavier weights to particles which view banks keep their appearance heterogeneous, that is, particles which worst matching value is low.

- Homogeneus weighting: If no tracking quality measure seems to be reliable, maybe particles should all be assigned the same mass and let body dynamics adjust misplaced particles.

5.4.6 Ragdoll elasticity and shape recovery

The number of iterations in the constraint solver step limits the elasticity of the whole articulated rigid body and influences tracking.

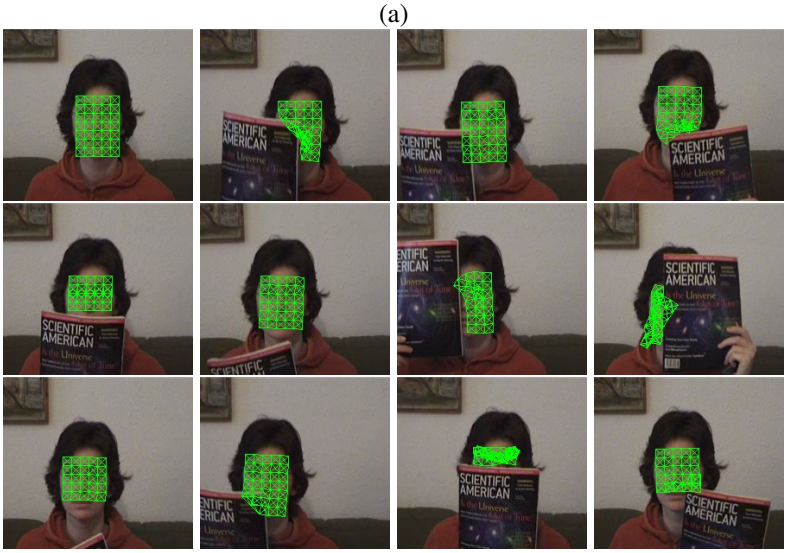
Elastic structures let particles follow their targets with a higher degree of independence. Particles are allowed to explore their search space and move around more freely, deforming the ragdoll as seen in Figure 5.18 a.

A higher number of iterations produce stiffer structures. Particles may still explore their search space, but they are not allowed to wander far away from their expected relative position within the structure. Misplaced particles are quickly corrected, limiting deformations.

In both cases, the original shape can be recovered after being deformed thanks to those particles that stuck to visible parts of the tracked object. These particles serve as an anchorage that allows internal constraints to push lost particles to their correct position, as seen in Figure 5.18 b.

The ability of sentient ragdolls to recover their shape is tightly related to

(a)



(b)

Figure 5.18: Rigidity limits freedom of movements. With just two iterations in the constraint solver step, resulting structures are very elastic (a). With ten iterations, the structure is much more rigid (b). Grid structures like the one shown in this example may still fold onto themselves, no matter their rigidity. In both cases, the structure is able to recover its shape after strong deformations if enough particles are still tracking their targets properly.

mesh configuration and template updating. Weak configurations, like grids, when combined with complete view bank updating, may lead to situations where the structure is folded and all particles store the same view. If particles forget the

appearance of their original target and the structure is not robust enough, the ragdoll will be unable to recover its shape. On the other hand, if particles store a canonic view or the structure is robust enough (e.g. a Delaunay triangulation with enough particles is used to create constraints), deformed ragdolls may recover their shape either matching canonic views or pushing particles when constraints are satisfied.

5.5 Discussion

The proposed kinematic system using Verlet integration and distance constraints has been successfully applied to tracking in a variety of situations. The combination of an elastic structure and a bank of views for each tracking particle allows the system to recover its shape, correcting misplaced particles.

Partial occlusions are also handled by our approach. It may happen due to the interference of a second object between the tracked one and the visual sensor or due to rotations of the tracked object (as seen in Figure 5.6). The effect is similar in both cases: some particles will lose their target and, most probably, the tracked template will be wrongly updated. However, as long as part of the object remains visible, some other particles may still track their targets properly. As the occlusion effect ends, the structure suffers a re-adaptation process as particles recover older views and constraints push them towards their correct position (Figure 5.18).

This is where the advantage of a multi-tracker approach with regard to a single-tracker approach stands out. In a multi-tracker scheme like the proposed one there exist multiple cues that may be used to correct misplaced particles, being links between particles the main important cue for relocation. Moreover, the effect of links is spread throughout the structure, so every particle contributes to this correction mechanism.

However, other cues can be considered. For example, link stress can be measured. If most links shared by a given particle are stressed, it can be assumed that the particle is stuck to a piece of background while the rest of the group is

tracking the object (although it could be the other way round). This clue can be included in our approach proportionally re-weighting each particle using the summed stress of its links. Different approaches could consider removing stressed links and killing unbound particles.

Link stress can be also used to detect scale changes. If the tracked object shrinks or grows all ragdoll links will experience a similar stress level, stretching if the object increases its size or compressing if it decreases. In those cases, link sizes could be updated to adapt the whole structure to the new size of the tracked object (Figure 5.19 shows a preliminary test). If this clue is conveniently combined with space-scale tracking particles, ragdoll structures would be able to track objects across scales.

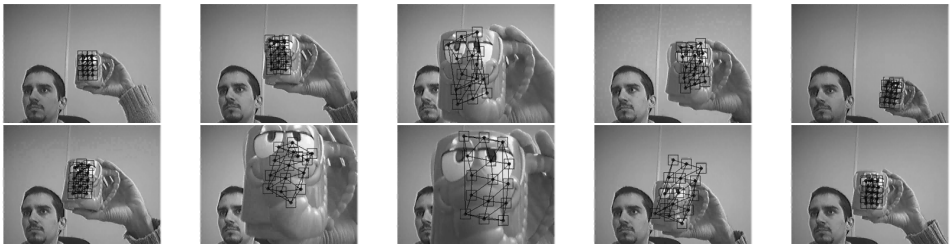


Figure 5.19: Changes in scale can be detected from link stress levels. Preliminary tests show that tracking objects across scale and space is feasible.

The updating behavior of neighboring particles could be considered when updating a given particle. If a rigid body is being tracked, it seems sensible that if most linked particles are not updating, the considered particle should not update either. However, this may not be the case for many objects, like faces, which parts may change independently (i.e. a blinking eye or a mouth while speaking), so it was not considered in our approach. A different kind of cues could make use of group behaviors in order to re-weight particles. For example, we may suppose that neighboring parts of a given object will evolve similarly through space. Thus, a particle which velocity is radically different from that of its neighbors is probably lost and its weight can be decreased.

6 Evaluation

THE two proposed particle-based tracking methods are empirically evaluated in this chapter. While many examples of the contribution of this work have been already shown in previous chapters, both tracking methods are now tested against five representative sequences.

In order to evaluate the goodness of proposed tracking methods, ground truth information is necessary. Five sequences where annotated frame by frame, locating the centroid of the target object and its scale, which is defined as the radius of the circle enclosing the object in each frame. However, these five sequences are just a representative subset of all the different tests performed during this work. They consider a variety of situations: rigid objects, deformable objects, out of plane rotations or fast movements, to name a few.

Both methods require a significative number of parameters to be set, leading to a wide variety of configurations and great versatility. For practical purposes some of these choices, such as search window sizes, search strategies, view bank capacity or the number of iterations of the constraint solver, will be fixed. For some others, such as feature maps, object descriptors, rule weights of particle layout, different combinations will be examined. The tracking swarms solution will be referred to for short as *Swarmtrack*.

6.1 Swarmtrack evaluation

Swarmtrack tracking activity $\varepsilon(\mathbb{S})$ is evaluated measuring, at each frame in each sequence, the Euclidean distance between the swarm's centroid $\vec{\mathbb{S}_c}$ and the location of the hand marked object's position $\vec{m_x}$, divided by its size m_s (See Equation 6.1). This division factor is required in order to obtain an error measure that is invariant to the scale of the tracked object, with $\varepsilon(\mathbb{S})$ values below one meaning that the centroid of the swarm remains within the image region occupied by the tracked object.

$$\varepsilon(\mathbb{S}) = \frac{\sqrt{(\vec{\mathbb{S}_c} - \vec{m_x})^2}}{m_s} \tag{6.1}$$

Two different experiments are conducted. The first one evaluates six scent-intensity feature combinations, using the static weights proposed for the steering behaviors of boids in Chapter 4. The second one considers variable random weights and five scent configurations, ignoring intensities.

The difference between *scents* and *intensities* should be highlighted. The former are image features that each particle stores as a descriptor of the object to be tracked. The latter are used to modulate the interest of a particle on a certain image region due to local measures like the presence of gradients or movement, which are completely unrelated to the tracked object. The use of intensities is optional and task-dependent.

6.1.1 First test configuration: scents, intensities and fixed weights

This first test considers color as the tracked scent and evaluates the effect of different image features used as scent intensities. Rule weights and search parameters are fixed to the following values: Hunt $w_1 = 1.0$, Cohesion $w_2 = 0.3$, Alignment $w_3 = 0.5$, Separation $w_4 = 0.01$, Momentum $w_0 = 0.1$ and search window size $n = 11$. Each swarm is composed by 150 particles approximately.

The following scent-intensity configurations are considered:

1. Scent: color - No intensity. Particles compute a movement vector towards the most appealing region in their vicinity, measured in terms of RGB color similarity between the color of a particle and image pixel values.

2. Scent: color - Intensity: gradient. Color scents are modulated using gradient magnitudes. Thus, particles are attracted towards pixels with the proper color placed on high gradient regions.

3. Scent: color - Intensity: inverted gradient. Similar to the previous configuration, but each location is weighted with the inverse of the gradient magnitude at that point. Particles are attracted towards pixels with the proper color placed on plain regions that show no boundaries.

4. Scent: color - Intensity: KLT points. Scents measures are halved at pixels that do not lie in the vicinity of a Kanade-Lucas-Tomasi (KLT) salient point.

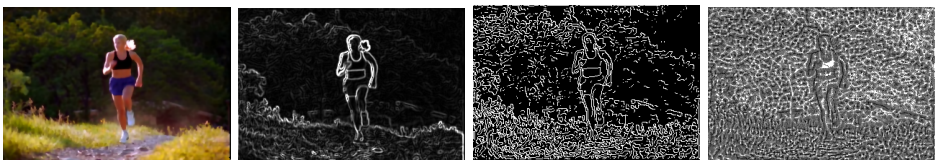
5. Scent: color - Intensity: inverted gradient, movement and KLT points. Movement is combined with most previous measures. Pixels are more attractive if they have a certain color, if they are placed on plain regions, if they lie next to a salient point and increase their weight if they contain movement.

6. Scent: color - Intensity: inverted gradient, movement, KLT points but no flocking activity: each particle looks independently for interesting pixels using the previous feature combination, but no flocking activity is included. That way, particles are able to roam freely across images, completely ignoring the rest of the swarm.

6.1.2 Second test configuration: scents and random weights

The second test evaluates different target descriptor combinations. No scent intensities are considered, and thus tracking depends solely on the target features stored by each particle. Figure 6.1 shows the four used feature maps. Rule weights

are updated at each time step using smooth cosine-interpolated random values, promoting heterogeneous behaviors.



(a) (b) (c) (d)

Figure 6.1: Scents used by tracking particles: a) RGB color, b) Sobel gradients, c) Thresholded zero crossings in the Laplacian, d) Simplified LBP codes.

The following combinations are considered:

1. Scent: color - No intensity. Particles compute a movement vector towards the most similar region in their vicinity, measured in terms of RGB color similarity between the color of a particle and image pixel values.

2. Scent: color and LBP - No intensity. Particles also store the simplified LBP code of the pixel where they were created, which contains texture information. Thus, they also look for regions in the image with texture descriptors similar to those they store.

3. Scent: color and gradient magnitudes - No intensity. Particles register the presence of boundaries on the region where they were created. Particles will be attracted towards target colors placed on boundaries or plain regions depending on their spawning point.

4. Scent: color and Laplacian - No intensity. Pixels placed at zero crossings in the Laplacian of an image are used as edge features, if the gradient magnitude at those points is higher than 16. Gradient magnitudes below that value are negligible in most contexts. Zero crossings are interesting binary features that are quite robust against light and scale changes.

5. Scent: color, LBP and gradient - No intensity. Texture and gradients are combined with color. The use of three scent features for tracking results in more restrictive searches.

6.2 Sentient ragdoll evaluation

A sentient ragdoll's tracking activity $\varepsilon(\mathbb{A})$ is evaluated measuring at each frame the distance between hand-marked objects and the ragdoll structure (see Equation 6.2). This distance considers both the Euclidean distance between the location of the hand marked object's centroid $\vec{m_x}$ and the ragdoll's centroid $\vec{\mathbb{A}}_x$, and the difference between the size of the hand marked object m_s and the ragdoll's size $\mathbb{A}_s$.

The centroid of the ragdoll is the average position of all particles in $\mathbb{A}$ and its size is the distance between its centroid and the furthest particle in the structure. This distance is divided by the size of the hand-marked object, so once again the error is invariant to the scale of the tracked object. If the ragdoll stays on the tracked object and both sizes are similar, $\varepsilon(\mathbb{A})$ values will be close to 1.0.

$$\varepsilon(\mathbb{A}) = \frac{\sqrt{(\vec{\mathbb{A}_c} - \vec{m_x})^2 + (\mathbb{A}_s - m_s)^2}}{m_s} \tag{6.2}$$

The amount of variables in the ragdoll solution is overwhelming. Both structural and behavioral parameters must be set to configure a sentient ragdoll: particle layout, which defines the ragdoll shape; structure elasticity; particle, template and search window sizes; particle weighting policy; view bank configurations... In order to focus on the more important aspects of this method, most parameters were fixed to values that have been found to be appropriate for most sequences:

- Template and search window: a fixed size of $11x11$ pixels was used for all templates. Search windows were defined as three times that size. Smaller search windows favor object loss, while bigger sizes increase computational costs.
- View banks: a maximum of three views were considered for each view bank. While only two views seem to yield similar results when a canonic view is included, only the use of a third view (or more in larger banks)

represents a short-term visual memory, properly registering recently used patterns.

- Articulated body elasticity: the number of iterations in the constraint solver step is limited to five. It allows a certain elasticity but also prevents strong deformations.
- Particle weights: particles masses are defined as the value of the first minimum of the worst view in the view bank. As explained before, it is a measure of how different that view is from the currently used one, and thus it represents the homogeneity of a view bank.
- Particle sizes: ragdolls are created on the initial frame of a video sequence defining a bounding box enclosing the target. Particles are defined as squares with sides 20% of the smallest bounding box side long. When scale salient points are used, their scale is used instead.
- Particle density: both in the grid and KLT layout, particles are not allowed to overlap more than 50% of their size. In grid layouts, grids are created using cells which size is exactly half a particle's side.

Once those parameters are established, twelve different configurations are evaluated. Four particle layouts are used: simple grid (four neighbors), robust grid (eight neighbors), KLT, and SURF salient points. Particles created using KLT and SURF salient points are linked using the edges of Delaunay triangulation.

Finally, each layout is combined with three template updating policies. View banks may be updated using contextual information, either completely replacing views when needed or keeping a canonic view, but no updating at all is also considered.

6.3 Use of boxplots

Boxplots are graphs that are built using five descriptive measures computed from the available data set: median, first and third quartile, minimum and maximum

values. Boxplots provide indication of data symmetry and skewness and also suspected outliers.

Boxplots are useful for comparing data sets side-by-side on the same graph, and they become especially handy for comparing tracking methods applied to video sequences that have different durations. Given how tracking quality is measured, explained in Sections 6.1 and 6.2, boxplots values below 1.0 for the tracking swarm solution mean that the swarm is inside the tracked object, while values around 1.0 for sentient ragdolls represent that the size and position of the ragdoll and the object being tracked coincide.

In a boxplot, the box itself contains the middle 50% of the data. The upper edge (hinge) of the box indicates the third quartile of the data set, and the lower hinge indicates the first quartile. The line in the box indicates the median value of the data. If it is not equidistant from the hinges, then the data is skewed. The ends of the vertical lines, or "whiskers", indicate the minimum and maximum data values, unless outliers are present. In that case the whiskers extend to a maximum of 1.5 times the inter-quartile range. Finally, the points outside the ends of the whiskers are outliers or suspected outliers.

6.4 Sequences analysis

6.4.1 Box sequence

The box sequence shows a rigid object being moved in front of a low cost webcam that returns very noisy images, which are smoothed using a bilateral filter. This sequence features slight scale changes, out of plane rotations (although all box sides look similar), motion blur and there exist certain chromatic similarities between the box and background objects. It contains 387 frames with a resolution of 320x200 pixels.

Swarmtrack

Figure 6.2 shows some frames of the box being tracked by the swarm:

Figure 6.2: Box sequence, featuring a rigid object. Swarm particles face two problems: color similarity between target object and background objects and sudden velocity changes.

Using color as the only tracked feature in this sequence is reliable as long as no gradient-related features are used to modulate scents. Using gradients or KLT points may attract particles towards background regions where edges are present, much more if the object suffers motion blur. When motion blur affects the tracked object its inner boundaries are softened, which contributes to particles leading towards static background objects with similar chromatic features.

If gradient regions are avoided, using inverted gradient magnitudes as intensity features, the swarm is able to stay on the tracked object. In this case, motion blur does not have any influence on tracking. Results obtained when particles are not flocking at all are relatively good, but just because the box gathers stray particles as it moves around. Figure 6.3 shows results for this first test.

If gradient magnitudes are not used as scent intensities, but they are part of the set of features that particles look for, tracking quality is improved. Apart from some frames where the swarm sticks to the lower part of the box, which creates some outliers in most boxplots (see Figure 6.4), using more feature maps to describe preys allows the swarm to stay on the tracked object during the whole sequence.

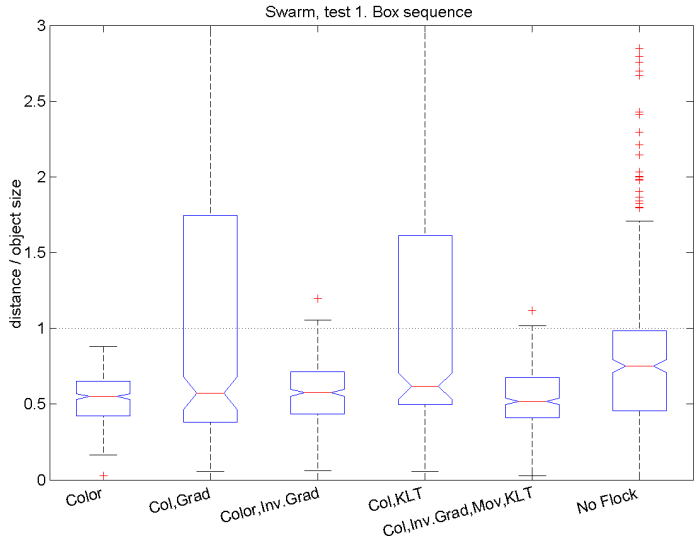


Figure 6.3: Box tracking results using Swarmtrack, first test. The swarm loses its target when it moves near the face if the used feature combination relies heavily on gradient magnitudes due to motion blur.

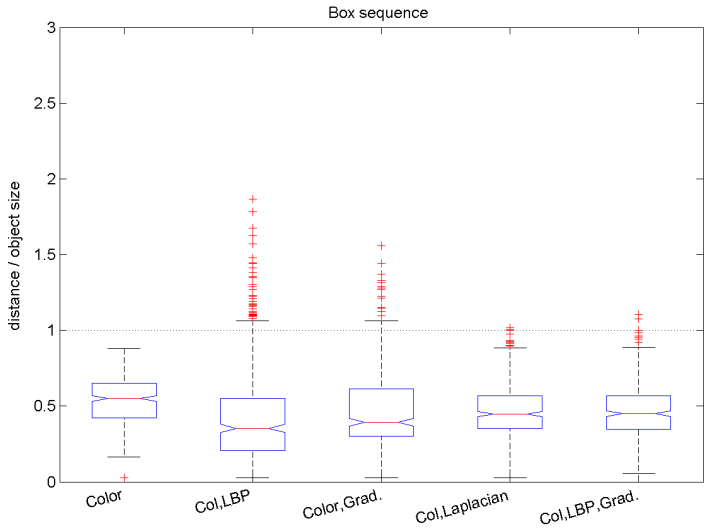


Figure 6.4: Box tracking results using Swarmtrack, second test. Richer prey descriptors produce better tracking results.

Sentient ragdoll

Figure 6.5 shows some frames of the box being tracked by the sentient ragdoll.

Figure 6.5: Box sequence, featuring a rigid object. The articulated body has to adapt to out of plane rotations, blurry object appearances and minor scale changes. The KLT particles layout is shown.

All configurations allow the sentient ragdoll to follow the box. However, if the updating policy replaces all templates in a view bank with novel views, grid layouts fold onto themselves. When original views are replaced and forgotten the structure is not able to recover its original shape. While folded structures still follow their target, they no longer fulfill their original purpose, as most particles are tracking the same view. This effect is stronger on the simple grid (SGRID in 6.6), because its structure is less rigid.

Structures created from a Delaunay triangulation of KLT or SURF points are much more rigid. Contrary to grid-like structures, placing particles irregularly (and establishing links between them created from the Delaunay triangulation) prevents folding.

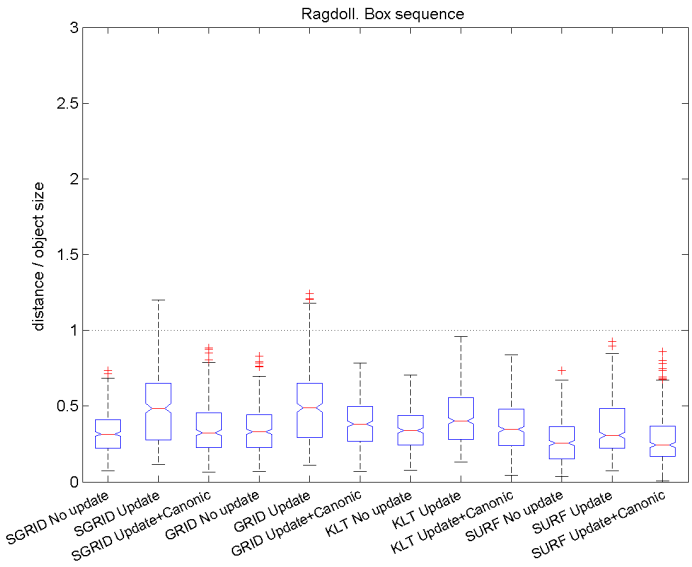


Figure 6.6: Box tracking results using the sentient ragdoll. Most configurations are able to track the object successfully. However, grid layouts with complete bank updating end up in folded structures.

6.4.2 Hand sequence

This sequence is similar to the previous one, but the tracked object is a moving and gesturing hand. Its skin color is shared with two non-target objects, the face and the arm, so the two main difficulties in this sequence are color similarity with background objects and the continuously changing shape of the object. This sequence contains 579 frames with a resolution of 320x200 pixels.

Swarmtrack

Figure 6.7 shows some frames of the hand being tracked by the swarm:

Figure 6.7: Hand sequence, featuring a highly deformable object. Hand's colors are similar to face's color, which may attract particles when the hand moves near the face.

At some point in this sequence the tracked hand moves in front of the face. While the swarm could easily stick to any of both objects, and indeed some particles stay on the face, most of them are correctly placed on the hand and attract those that stay behind. Particles have a certain momentum that pushes the swarm in the direction of previous movements, granting the swarm a weak predictive capacity. A similar situation occurs in the box sequence. However, in that case, the tracked object stops for a while losing momentum. When the box starts moving away from the face, particles are not able to leave the region.

No scent intensity combination seems to be significantly better than other, as shown in Figure 6.8. Once again, the object gathers stray particles as it moves around when no flocking is used.

Using combined scent features produce even better results, with tracking distance error medians below 0.5. However, when Laplacian edges are used or LBP and gradient magnitudes are combined, particles seem to prefer tracking the arm instead of the hand in the latter frames of the sequence, creating the outliers shown in Figure 6.9.

This effect occurs because the region where the swarm was initially created, the palm of the hand, did not contain strong gradients. As the hand varies its gesture, gradients and new textures appear when fingers cover the palm. At some point, particles find that the arm region, where no gradients are present, is more similar to what they should be tracking and switch their object of attention.

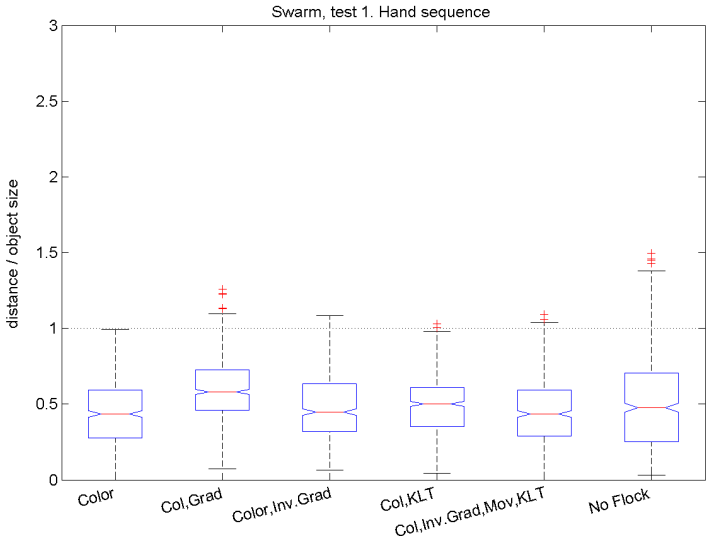


Figure 6.8: Hand tracking results using Swarmtrack, first test. The swarm is able to overcome the presence of a similar object in the background thanks to steering behaviors and gained momentum.

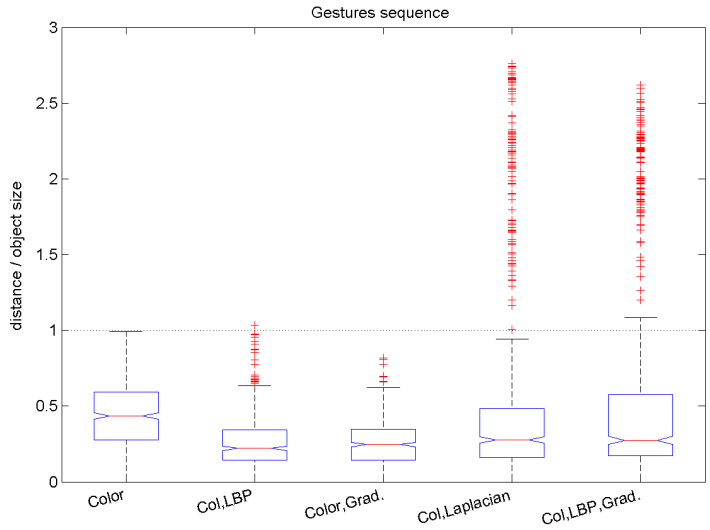


Figure 6.9: Hand tracking results using Swarmtrack, second test. In the latter frames of the sequence, particles seem to prefer the arm to the hand, producing many outliers.

Sentient ragdoll

Fig 6.10 shows some frames of the hand being tracked by the articulated visual body.

Figure 6.10: Hand sequence, featuring a highly deformable object. Hand's palm does not show many good features to track. The articulated body may thus easily fold onto itself.

This sequence is an example of how particles lose their target if they are allowed to replace all their view bank when updating. The three most rigid configurations (8-neighbor grid, KLT and SURF) stick to the face when the hand moves in front of it, as all particle views drift away from the original hand appearance. The 4-neighbor grid, however, manages to follow the target, which may indicate that flexible structures are more able to adapt to changes than rigid structures.

Sentient ragdolls with KLT and SURF layouts track the hand even when this sequence also shows a weakness of many salient point detectors: the region where the articulated body is defined, the palm of the hand, does not contain many salient features. Thus, created ragdolls do not contain more than half a dozen particles, which leads to unstable tracking results (shaky centroid and body orientations).

The best result is given by the rigid grid layout configuration using canonic views. However, as shown in 6.10, the ragdoll is not able to keep its shape and almost folds onto itself. Once again, the lack of features within the palm is the reason for this effect.

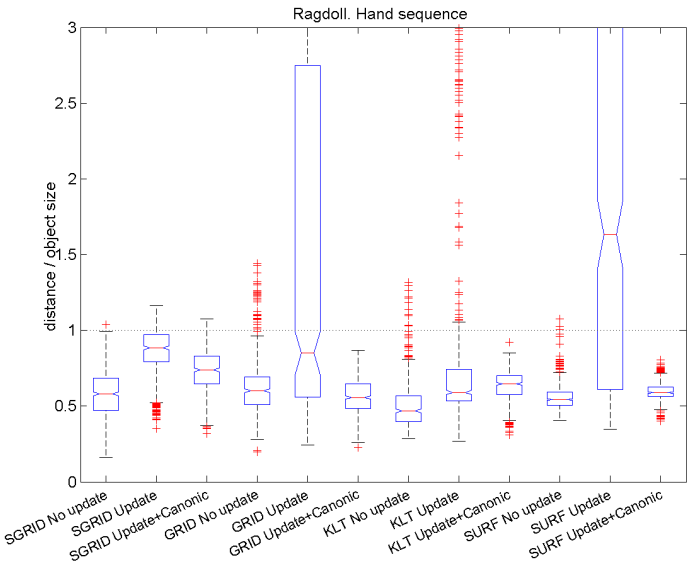


Figure 6.11: Hand tracking results using the Sentient ragdoll. Rigid configurations (Grid, KLT and Surf) using complete view bank updating stick to the face when the hand moves in front of it. Grids with no updating or those using canonic views yield better results.

6.4.3 Crossing pedestrian sequence

This sequence features a pedestrian crossing a street, seen from a far away camera. The pedestrian is relatively small in relation to the image size, its colors are similar to those of the background and images suffer strong JPEG compression artifacts (which, thanks to the bilateral filter applied to all sequences, are softened). At some point, the pedestrian is slightly occluded by some hanging cables. This sequence contains 188 frames with a resolution of 320x200 pixels.

Swarmtrack

Figure 6.12 shows some frames of the pedestrian being tracked by the swarm:

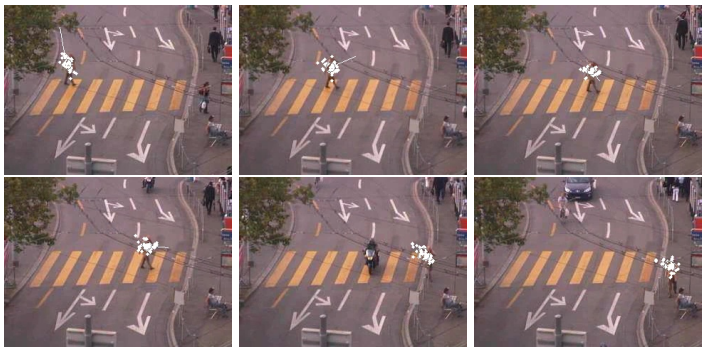


Figure 6.12: Crossing pedestrian sequence, featuring a small object which color features are similar to the background. While particles may feel attracted to background objects at some point, flocking behavior keeps them together.

Flocking is crucial in this sequence, leading particles to keep together moving around the target and following it until the end of the sequence. All scent-intensity combinations allow the swarm to follow its target successfully, except when no flocking is used. The colors of the tracked object are distinctive enough, but they do not prevent particles from moving away looking for similar targets.

Adding more scent features also yields good tracking results, as shown in Figure 6.14. The target contains many strong gradients and salient points, so feature combinations including gradient and KLT points behave well.

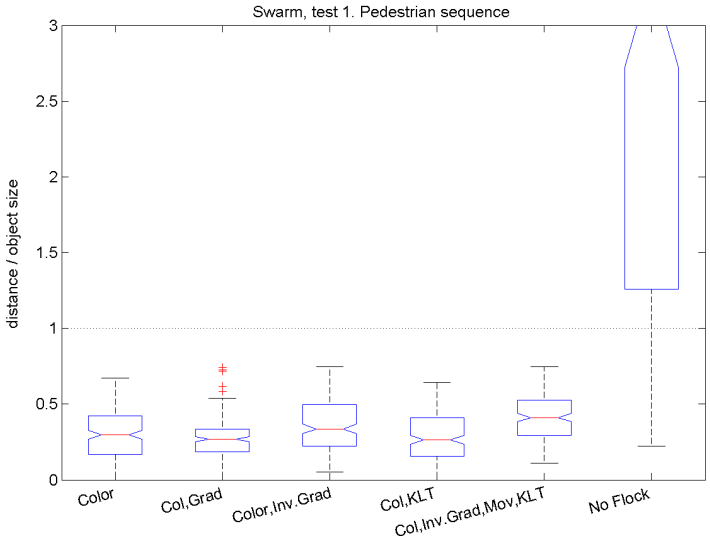


Figure 6.13: Crossing pedestrian tracking results with Swarmtrak, first test. Because the tracked object's colors are relatively different from the background and the object contains many strong gradients and KLT points, all methods are able to track it successfully. When no flocking behavior is used, however, particles stick to background patches and the object is lost.

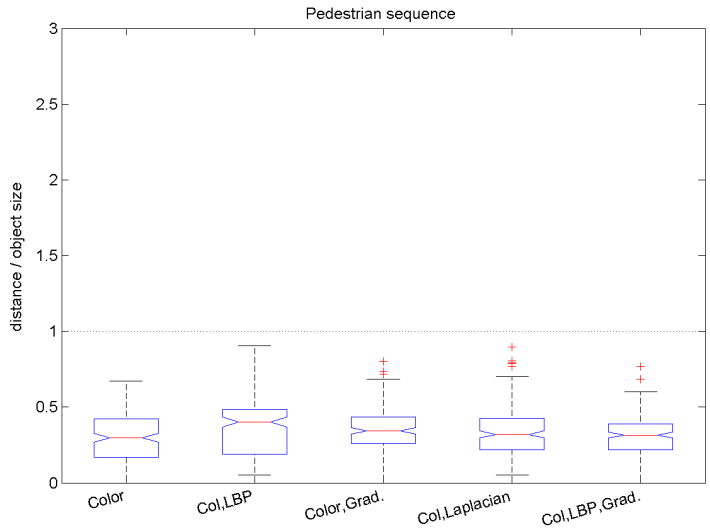


Figure 6.14: Crossing pedestrian tracking results with Swarmtrack, second test. Results are also good when more features are tracked.

Sentient ragdoll

Figure 6.15 shows the pedestrian being tracked by the sentient ragdoll.

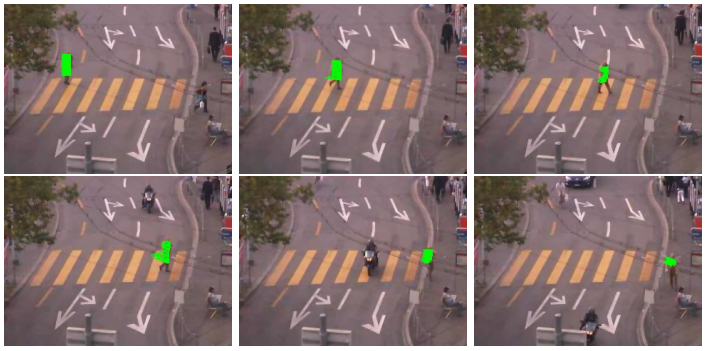


Figure 6.15: Crossing pedestrian sequence. The target contains many salient features that should be easy to track.

This sequence is an example of why a pattern matching based tracker, even using context-based updating, may drift away from the original appearance of its target. Once again, the target is lost if particles are allowed to replace all their view bank. As the pedestrian crosses the street, its shape changes slightly (his arms and legs move), but the plain grey background is almost constant. Templates from the pedestrian's outer parts contain background. When these templates are matched against the search window, those parts with background (that is, those parts that do not change significantly) produce better matching results. When templates are updated, they slowly include more and more background until they finally substitute the original patch with background views. These particles build up mass, because all their views look similar, so they finally pull from those particles that may be correctly following the right target.

Canonic views prevent original appearances to be forgotten and avoid templates from drifting towards plain background views while particles are still allowed to adapt to appearance changes. Configurations using the SURF detector fail because not enough points are found in smaller scales. Most salient points are found in scales that also include background, and thus particles end up losing their target, as shown in Figure 6.16.

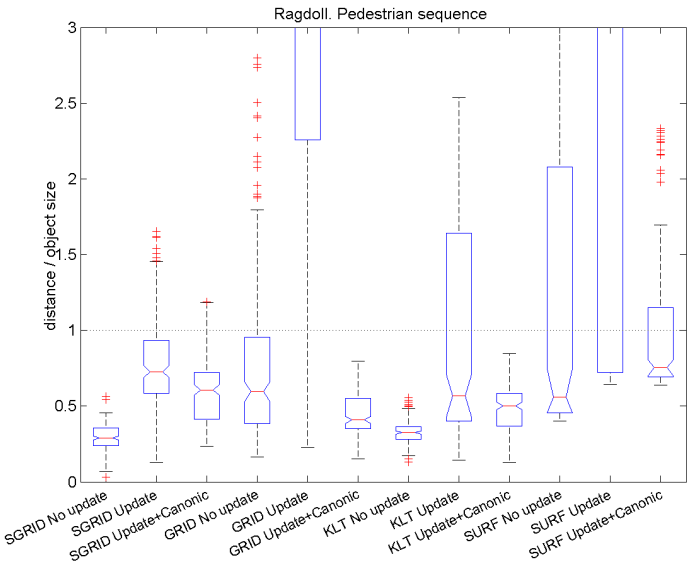


Figure 6.16: Crossing pedestrian tracking results with the sentient ragdoll. Configurations with complete view bank updating, once again, lose the target. SURF configurations fail because the object is too small to find enough scale salient points.

6.4.4 Skier sequence

A fast moving skier descends a slope in a zigzag movement. Although his yellow cardigan stands out from the white background, his fast movements may become a complication for methods based on local searches. This sequence is the shortest one, with only 66 frames with a resolution of 320x200 pixels.

Swarmtrack

Figure 6.17 shows the skier being tracked by the swarm.

Figure 6.17: Skier sequence, featuring a fast moving object. Although target's colors are very distinctive, its fast zig-zag movements are difficult to track.

The swarm has no problems at all following its target even though its fast speed. The distinctive colors of the target, strong gradients and high number of salient points allow particles to follow their prey pixels successfully.

For fast moving objects like the one shown in Figure 6.17, most tracking approaches based on local searches need a search window big enough to keep up with object speeds. However, in the present approach, as far as enough particles find their target they may attract the rest of the swarm. This effectively increases the search space, even when the search space of each single particle is not big enough to cope with high speeds.

Using more restrictive scents combinations also results in successfully tracking, as shown in Figure 6.19

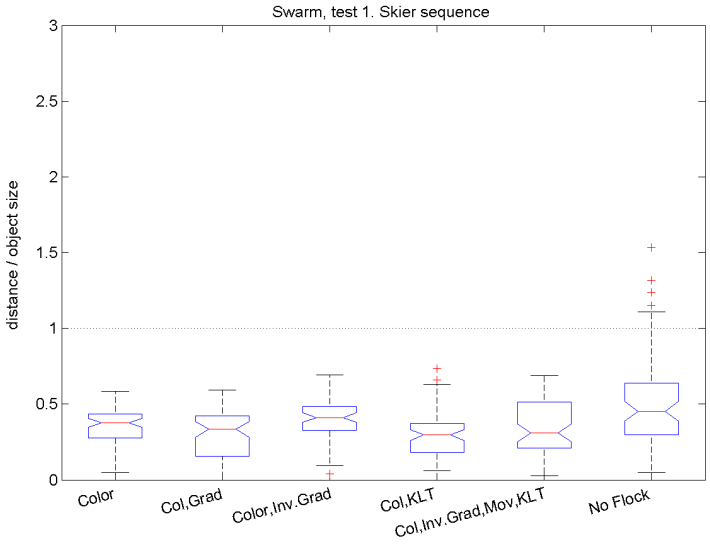


Figure 6.18: Skier sequence tracking results with Swarmtrack, first test. All swarm configurations track the object successfully.

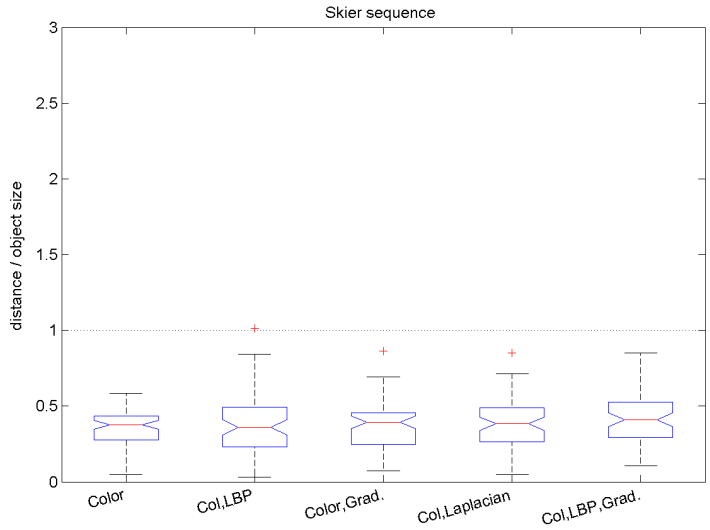


Figure 6.19: Skier sequence tracking results with Swarmtrack, second test. All swarm configurations track the object successfully.

Sentient ragdoll

Figure 6.20 shows the skier being tracked by an articulated visual body.

Figure 6.20: Skier sequence. Fast moves make it difficult for most particle layouts to track the object properly.

Most configurations seem to chase the target, although it is difficult because of its fast speed. However, both grid configurations with no updating mechanisms completely fail at tracking, losing the target shortly after the first frame. Grids contain many patches created in plain yellow regions, which are difficult to track.

KLT and SURF configurations with no updating, on the other hand, are able to track the object because they focus on salient features. SURF yields best results thanks to particles performing tracking at different scales concurrently, which allows the ragdoll to cope with speed.

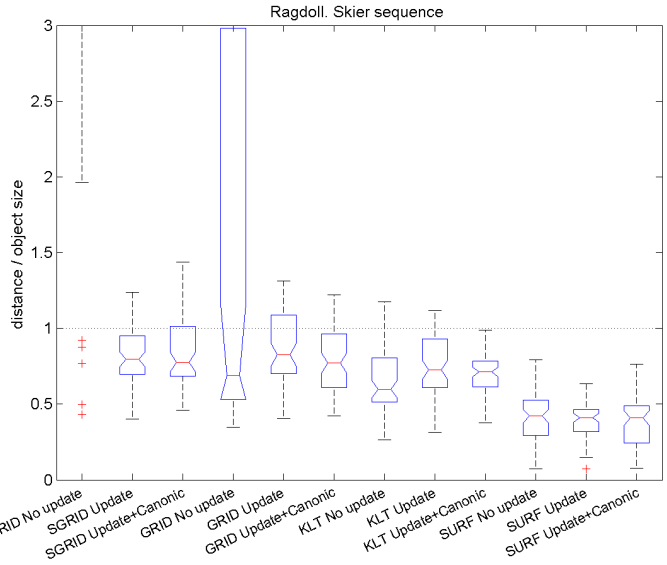


Figure 6.21: Skier sequence tracking results with the sentient ragdoll. Both grid configurations with no updating mechanisms are not able to adapt themselves to the target's movement. Best results are achieved with SURF configurations, because searches are performed at different scales.

6.4.5 Girl sequence

This sequence features a girl playing with her newly acquired webcam. She is testing the camera software's ability to track faces and keep subjects in plane, so she moves around and makes faces. Due to her movements, her face suffers scale changes, out of plane rotations and even goes out of view. The tracked object in this sequence is quite big in relation to image size. This sequence contains 873 frames, sized 320x200 pixels.

Swarmtrack

Figure 6.22 shows some frames of the girl's face being followed by the swarm.

Figure 6.22: Girl sequence, featuring a large object. The swarm wanders within the tracked object.

All scent-intensity combinations allow particles to track their target successfully. However, given the big size of the tracked object and its homogeneous chromatic features, particles may wander within its image space (e.g. at some point the swarm is located on the forehead and some time later the swarm is placed on a cheek).

When more restrictive features are used to describe preys, particles are able to stick to facial landmarks reducing the wandering effect.

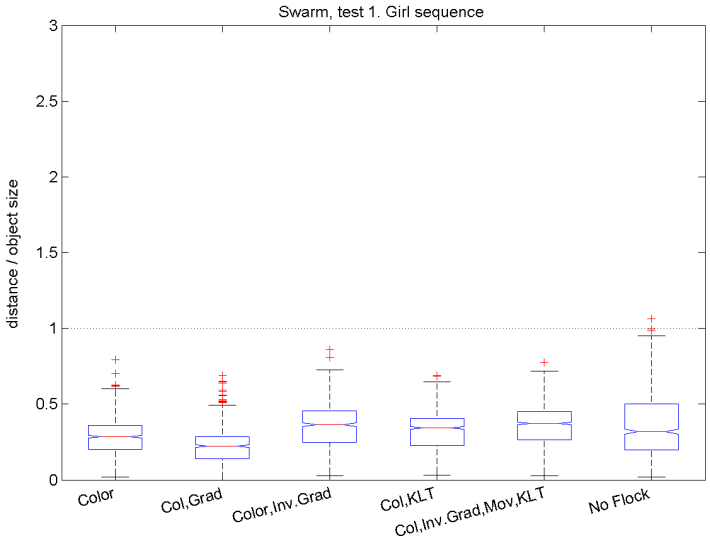


Figure 6.23: Girl sequence tracking results for Swarmtrack, first test. Even with no flocking behavior most particles are able to stay within the object, because the hair clearly limits its region.

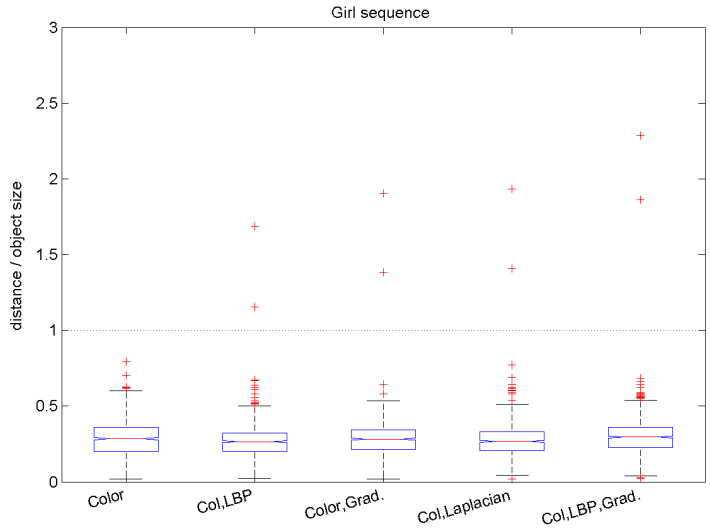


Figure 6.24: Girl sequence tracking results for Swarmtrack, second test. Using more features for tracking, the swarm is able to stick to facial landmarks and reduce the wandering effect.

Sentient ragdoll

Figure 6.25 shows the girl's face being followed by an articulated visual body.

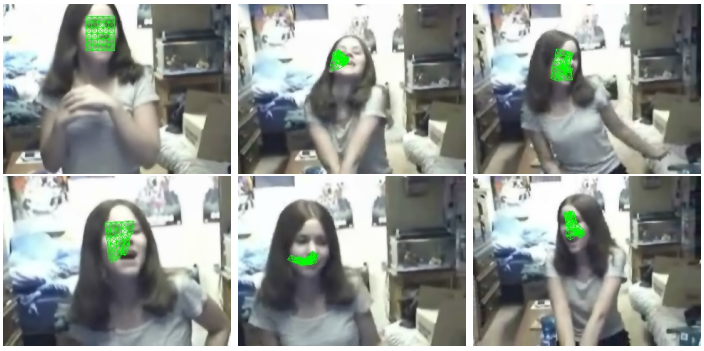


Figure 6.25: Girl sequence. The tracked object contains many good features to track, but some out of plane rotations, scale changes and complete occlusions (when the object leaves the scene for some frames) represent a challenging situation.

Complete view bank updating, once again, leads to object loss. Facial views are slowly replaced with dark hair patches because their plain features are easier to match. In the second half of the sequence, all original templates have been completely substituted. Configurations using canonic views overcome this problem because particles are able to recall the appearance of their original target.

The ability of sentient ragdolls to recover their original shape stands out when particles do not update at all or, to a lesser extent, when canonic views are used. When lighting conditions are constant and the appearance of targets does not change permanently, objects can be tracked without updating. If view banks contain a proper representation of their original target (either canonic or properly updated), the structure is able to quickly recover from strong deformations, as seen in Figure 6.25. The use of canonic views allows a certain degree of adaptation to changes, but also slows down shape recovery.

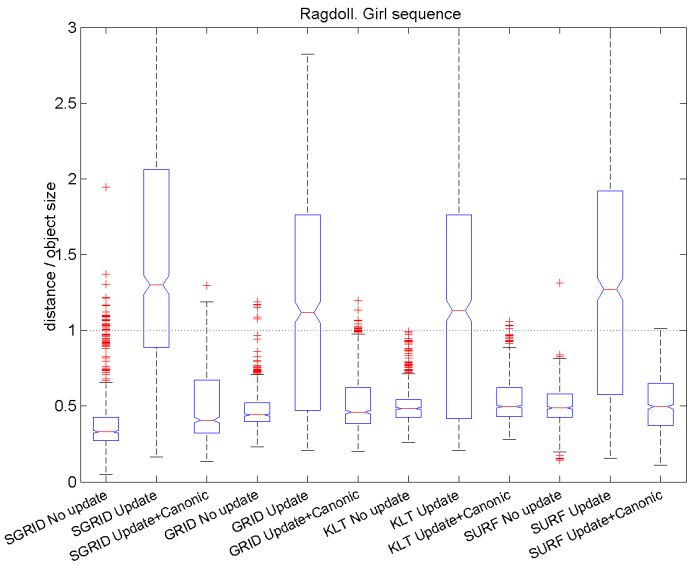


Figure 6.26: Girl sequence tracking results using the sentient ragdoll. Once again, all configurations using complete view bank updating fail. Fast object movements produce templates drifting towards hair regions.

6.5 Other methods

The best configuration of each method for each sequence are tested against two tracking methods: Opentracking and Camshift.

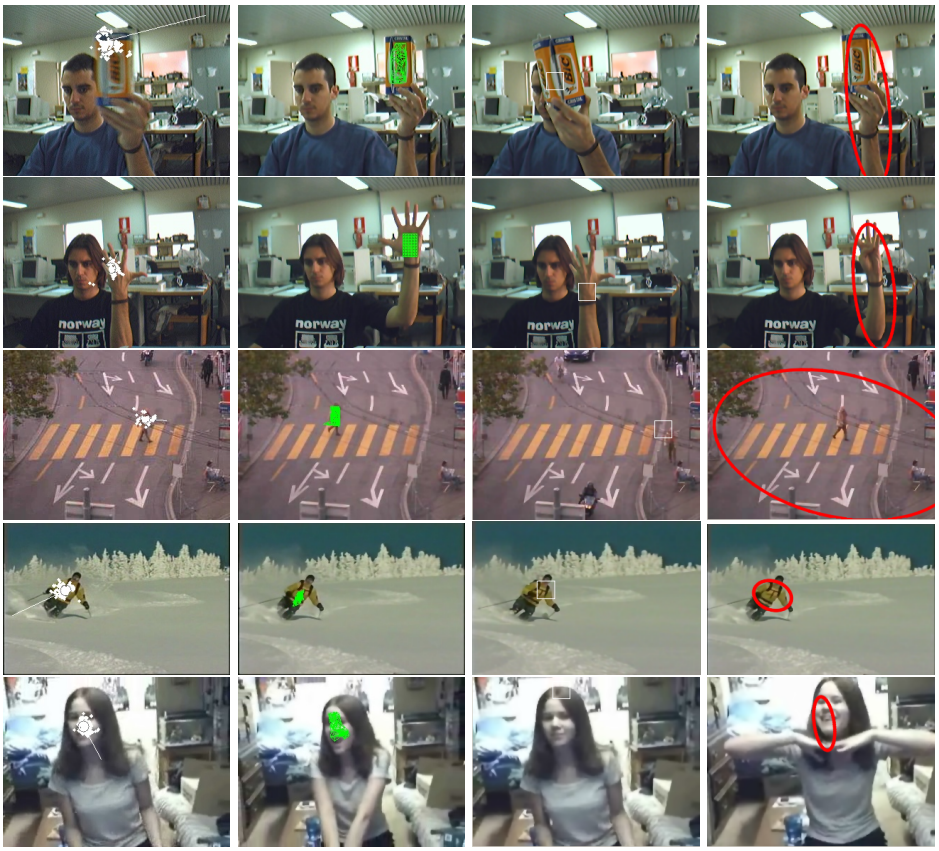


Figure 6.27: Frame examples from each method. From left to right: Swarmtrack, Sentient ragdoll, Opentracking and Camshift.

OpenTracking (Guerra, 2002) (Guerra et al., 2005) is the pattern matching tracking method that each particle uses independently in the sentient ragdoll solution. Therefore, it is similar to a one-particle ragdoll tracker with just one view in the visual memory. While relying on simple local searches for template matching, it proposes a sensible context-based template updating policy.

Camshift (Bradski, 1998) (Allen et al., 2004) is a modification of Meanshift, a robust nonparametric technique for climbing density gradients to find the mode

(peak) of probability distributions. In Camshift, the mean shift algorithm is modified to deal with dynamically changing color probability distributions derived from video sequences.

These two methods have been chosen because they are algorithmically similar to proposed solutions. Swarm particles, while looking for prey pixels, follow a hunting behavior that leads them towards regions showing a higher density of desired features. Therefore, they perform a peak search which is similar to how the mean shift algorithm seeks distribution modes. Ragdoll particles are based on OpenTracking, so comparing the efficiency of a single-particle tracker with that of the articulated visual body was necessary in order to highlight the advantages of population-based method.

Figures 6.28, 6.29, 6.30, 6.31 and 6.32 show the boxplots of the best configuration of each swarm against OpenTracking and Camshift for the same sequence.

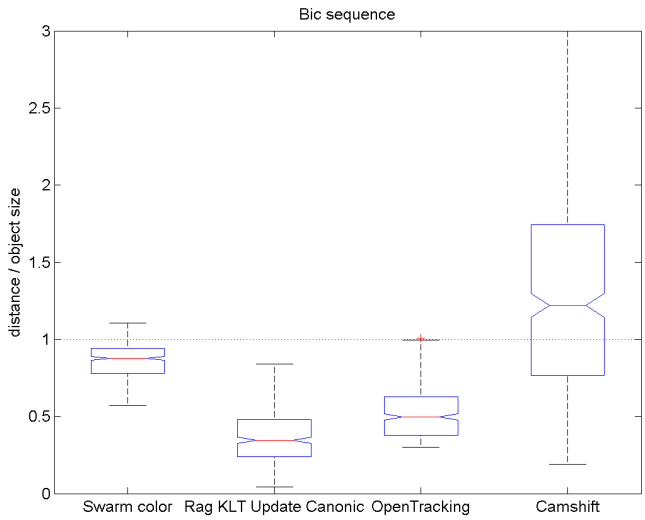


Figure 6.28: Swarmtrack and Sentient ragdoll compared to OpenTracking (single pattern matching using context-based updating) and Camshift with the Box sequence.

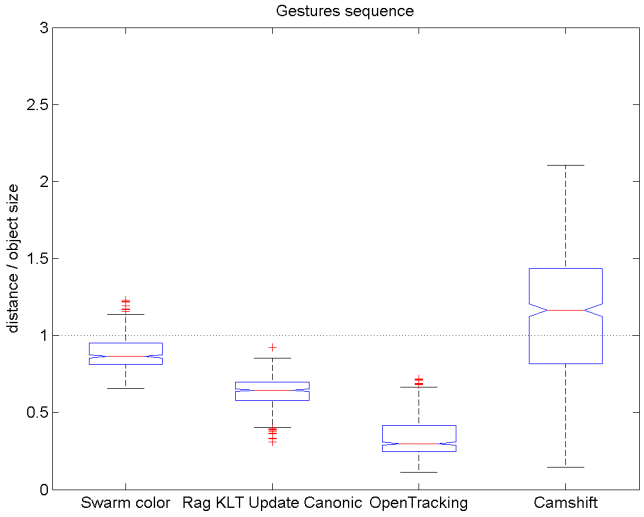


Figure 6.29: Swarmtrack and Sentient ragdoll compared to OpenTracking (single pattern matching using context-based updating) and Camshift with the Hand sequence.

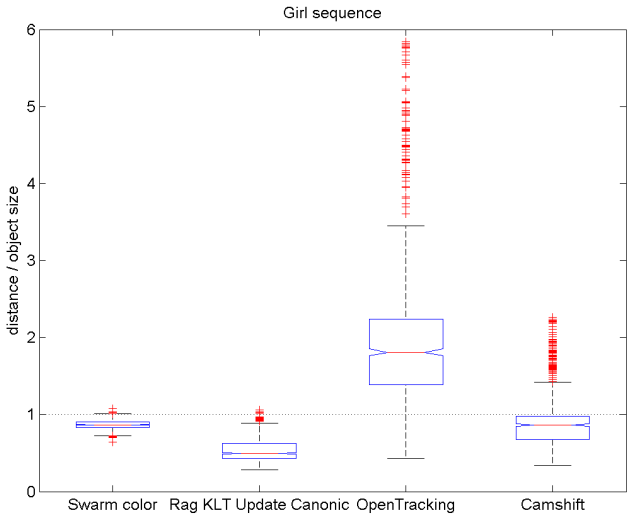


Figure 6.30: Swarmtrack and Sentient ragdoll compared to OpenTracking (single pattern matching using context-based updating) and Camshift with the Girl sequence.

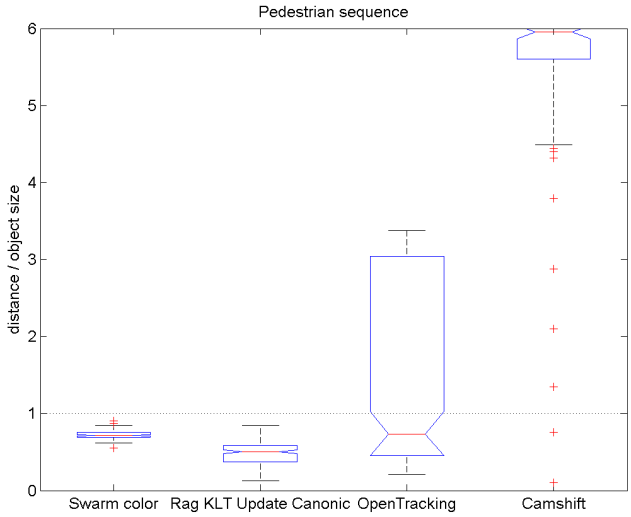


Figure 6.31: Swarmtrack and Sentient ragdoll compared to OpenTracking (single pattern matching using context-based updating) and Camshift with the Pedestrian sequence.

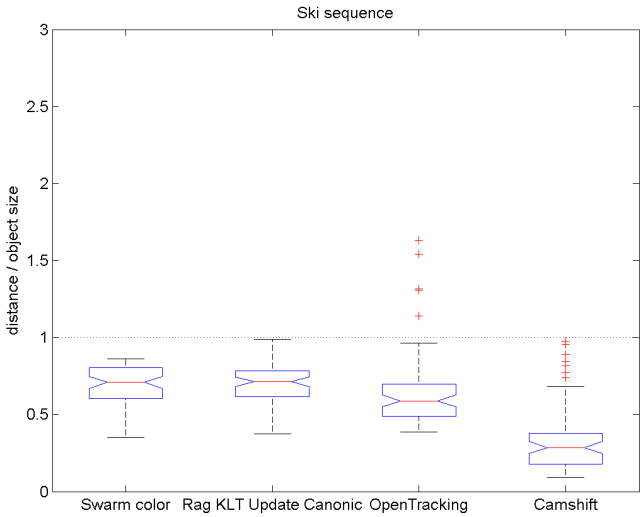


Figure 6.32: Swarmtrack and Sentient ragdoll compared to OpenTracking (single pattern matching using context-based updating) and Camshift with the Ski sequence.

OpenTracking suffers the same problems that have been previously exposed for the ragdoll solution: updating may lead to object loss, as it happens in the Pedestrian sequence. In most occasions, the initial location of the tracker is critical: it must be placed on a clearly distinguishable region. Otherwise, the tracker will easily drift away, as it happens in the Girl sequence when the tracker is not placed on the bridge of her nose (where it captures part of her eyes and eyebrows). However, no updating at all is even worse, with the tracker not being able to follow its target for more than a few frames. The proposed ragdoll solution inherits these problems, but only partially. It is much less sensitive to the initial placement of the tracker because, while some particles may be wrongly placed, many others may still capture useful patterns.

Camshift is a powerful tracking method that calculates the color (e.g. hue component) probability distribution of the tracked object within the search window, and it is able to adapt to scale changes. As a method based on color, it suffers from certain weaknesses that are unavoidable if no structural information is used. If the tracked object shares chromatic features with background objects, Camshift (and Swarmtrak, or any other method based on color), may fail. In the box sequence, the tracked region grows from its initial size until it encompasses both the box and the whole arm. Something similar happens in the hand and pedestrian sequence.

6.6 Discussion

The essence of Swarm Intelligence, that is, the use of distributed and self-reliant entities which collaborate unknowingly to achieve a complex task, has been successfully applied to the problem of object tracking in video sequences. While object tracking in its most general form has not been solved yet and both proposed approaches have their own weaknesses, new perspectives have developed for research.

Swartrack, the solution based on boids, is especially suitable for tracking highly deformable objects. Due to the lack of structural constraints, the swarm is able to adapt to any underlying shape and track it as long as its appearance is distinctive enough. However, objects will be probably lost if their share chromatic features with nearby image regions because current particles are guided almost exclusively by color. Moreover, they lack a feature updating mechanism, so any appearance change will not be properly registered.

Nevertheless, multiple image features can be used to strengthen the method. While some features like gradients and textures work better as object descriptors, other features like movement and figure/ground segmentation can be used to weight search regions. The combination of multiple features is crucial to overcome situations where certain image measures may be unreliable. Gradient based features, for example, are not useful if motion blur is present (which greatly depends on the quality of the image capture device), because boundaries become blurry when objects move. The negative effect of motion blur could be cancelled if other features like color and optic flow may take lead.

While the size of the search window of each particle is relatively small, the combined effect of all search windows and steering behaviors increase the swarm's search space. Thanks to this property, tracking swarms are able to overcome partial occlusions and keep up with fast moving objects as lost particles are guided by the rest of the swarm.

Among all scent-intensity combinations, tracking swarms using color and dynamic rule weights seem to perform best. While object tracking based on

chromatic features with no template updating is not reliable at all, it is still useful under controlled environments. Dynamic rule weights produce swarms that can be used in broader contexts, contrary to static weights which values must be found empirically.

Sentient ragdolls applied to tracking offer a blend between the free roaming particles of the pure swarm approach and rigid template tracking solutions. While sentient ragdoll particles perform their tracking activity independently, they must comply with the internal constraints that shape the structure. This awareness of its own conformation allow sentient ragdolls to push stray particles to their correct relative position within the structure, effectively recovering their original shape after strong deformations.

Elastic structures seem to perform better than rigid configurations, because particles enjoy greater freedom of movement. However, the number of iterations of the constraint solver should be chosen carefully in order to obtain an adequate degree of elasticity. It should allow particles to perform their tracking activity with ease, but it should also allow the structure to recover its shape after deformations. Five iterations seem enough to achieve this goal.

Particle layouts based on space-scale salient points are suitable for tracking objects at different scales concurrently. While some particles focus on small details, other particles perform a more general tracking activity looking for the object as a whole or locating big pieces of it. Tracking is strengthen with this approach.

The most influential parameter affecting both the tracking activity and a sentient ragdoll's shape recovery ability is the template updating policy. The currently adopted scheme based on context is suitable for short-term tracking activities. While it is much more robust than other template updating techniques, it does not prevent particle from replacing all views with strange templates. Sentient ragdolls with irregular layouts may overcome this problem, but not indefinitely. In a general setting, as soon as adverse circumstances meet, the tracked object will be lost.

Is updating really necessary? If an object is being tracked through the sum

of its parts, using solutions based on swarms, articulated rigid bodies or something similar, is it necessary to update each part in order to track the object properly?

If a face is being tracked and the tracker is created initially on a frontal view, when is updating required? If local features change (e.g. blinking eyes or a mouth while talking), there are still parts like eyebrows or noses that are more or less constant. If the face performs an out of plane rotation, half the face is still visible, and those particles tracking that part can do it properly.

If light conditions change, is updating required? Would not it be more adequate using a light invariant region descriptor? But what if the head completely turns around? Should it still be tracked? Isn't it a completely new object? It is, semantically, part of the same object, a head, but not a face anymore.

The used template updating solution can deal with changes but, obviously, only if tracking is successful. If the tracked pattern changes its appearance drastically, the currently used pattern matching method may not be able to locate it. The best matching position may not be right, and thus the updating mechanism will also fail, adding a strange view to the view bank.

This problem seems unavoidable with currently used color based descriptors. An example of this situation could be a pedestrian who walks into a dark shadow. While the transition may be smooth, his color features change drastically. Tracking may or may not, depending on surrounding context, locate the new target's position. If it does it properly, the updating mechanism may incorporate new templates to the view bank, registering how the pedestrian's appearance darkens. Otherwise, templates will drift away from the current object's appearance, as seen in Figure 6.33.

However, such an evident appearance change is not necessary for the updating mechanism to fail. If the tracked object varies while moving, due to internal deformations or motion blur, tracking may drift towards more stable regions. That way, the tracker may wander within the tracked object or, at some point, reach its boundaries and completely abandon it, as shown in Figure 6.34

Updating must be tackled carefully in a tracking system, if the tracker is

Figure 6.33: Drifting template. Due to strong appearance changes, the updating mechanism adopts strange views and loses the target.

Figure 6.34: Drifting template. Tracking drifts towards low variation regions.

not supported by a detection system or any other confirmation method that avoids templates to drift away from reliable views. For most systems, it seems preferable not to update at all and rely on invariant descriptors, in order to have reliable measures about whether the target object is still being tracked or not.

7 Conclusion and Future Work

7.1 Conclusion

In this thesis, the topic of Swarm Intelligence applied to computer vision, and more specifically to real-time object tracking in video streams, has been addressed. The main part of this work has focused on proposing and building two novel swarm-based solutions that are able to track visual objects in video sequences in a wide variety of real-world scenes. The first model considers a pure swarm structure which members are guided by psycho-social factors in a prey-predator metaphor, while the second model proposes a swarm which individuals are subject to structural constraints that shape an elastic structure. Quantitative experimental results confirm the suitability of both methods for many different working conditions.

It should be emphasized that the ultimate purpose of the present work consists on the general application of swarming techniques to computer vision tasks, that is, the solution of computer vision problems using multiple individual and relatively simple agents working independently, collaborating locally to achieve a task that is much greater than individual capacities.

The work described in this document can be summarized in the following conclusions:

- Swarm Intelligence and population-based solutions are inspired by biological models. Approaches based on social insect behavior have proved to be especially suitable for distributed control and optimization algorithms in a wide range of applications. While some approaches consider detailed theoretical models of the creature, group or activities they simulate (e.g. pheromone lay and evaporation rates), many others rely on a powerful metaphor (e.g. PSO and Genetic algorithms) to achieve successful systems.

- Approaching a problem through metaphors usually results in lateral thinking, and may allow finding creative solutions which would have not been considered otherwise. Through metaphors, problems can be remodeled into similar but unrelated situations. Once the original problem has been temporally substituted, attention can be focused on solving the meta-problem. Finally, generated ideas are back-mapped and adapted to the original task, if possible. This procedures are extensively accepted in arts and design, where creativity is crucial (Casakin, 2007).

- Scarce references are found in literature where Swarm Intelligence is applied to find solutions to computer vision problems in real time. The application of Swarm Intelligence to computer vision tasks creates new and stimulating approaches to visual problems, opening a relatively unexplored field. As swarm members focus on their own jobs, carefully designed to be relatively simple and self-reliant, solutions to complex problems emerge exploiting synergy and/or stigmergy. Tracking emerges from social and physical interactions between individuals.

- Most object tracking solutions found in literature rely on a single optimum to locate the new position of the tracked object, rejecting alternative locations. Thus, tracking is usually resumed at each time step from a single previous position. Approaches based on Swarm Intelligence may exploit the distributed nature of swarm members to track object parts, different object representations or multiple objects simultaneously.

7.2 Main contributions

In summary, the work described in this thesis made the following contributions:

1. An introduction to population and swarm-based methods, including genetic algorithms, ant colony optimization and particle swarm optimization (PSO). Differences between the proposed tracking swarm and PSO were discussed. While both methods are noticeably similar, the main advantage of our model lies in the possibility of considering heterogeneous populations. Our swarm members may adopt different tasks, working procedures and even perceptual abilities, but still collaborate towards a single common goal.

2. A review of proximal object descriptors and tracking methods. Object tracking literature is plagued with a large amount of solutions, and most of them are designed for specific situations. Although this work's contributions do not solve the general tracking problem, they offer a suitable alternative to classic methods in a wide variety of scenarios.

3. A novel swarm tracking solution derived from a predator-prey hunting metaphor and the psychosocial behaviors simulated by a computer model of coordinated animal motion, as defined for Craig Reynolds's Boids (Reynolds, 1987). Predator particles composing the swarm extract a set of features to be tracked from the image region that encloses the target object. Then, during the following video frames, each particle will track its own target independently, although subject to three steering behaviors: Cohesion, Alignment and Separation. The location and velocity of the tracked object emerges from the weighted swarm's centroid and the weighted average of all predator particles. The main features of this method include:

   (a) The complete independence of particles in relation to the adopted tracking method: each particle may use any tracking method found in literature, contrary to PSO solutions where all particles must share the same technique. This allows the creation of heterogeneous swarms which attention may spread through a wide variety of image features.

(b) Thanks to this independence, each particle's tracking activity can be computed in parallel, making the best of current multi-core architectures. This would allow the creation of heavily populated swarms, ideally increasing the robustness of group decision or mode-based processes.

(c) Steering behaviors do not impose structural constraints. While different rule weights may create organized groups that behave like a flock or school, or chaotic swarm-like dynamics, particles' relative positions are not fixed. They may roam freely in the image space, and thus Boids become suitable for tracking objects with any degree of non-rigidity.

(d) Tracking methods based on local searches may lose fast moving targets and occluded objects. While tracking swarms using local searches also suffer this condition, the search space is effectively increased to encompass the whole region covered by the swarm. That way, fast objects can be tracked even when small search windows are used, and complete and partial occlusions may be overcome if enough particles find their target and attract the rest of the group.

4. A novel tracking solution based on elastic structures that are composed by particles linked by infinite stiffness springs (also known as *ragdolls*). They can be considered swarms which particles suffer structural constraints and which, therefore, are forced to keep a fixed relative position in the group. The adopted solution is based on the work of Thomas Jakobsen (Jakobsen, 2001), who proposed a robust physics engine where particle dynamics are simulated using a Verlet integration method and constraints are solved in an iterative process. These structures are extensively used in the videogame industry to simulate articulated rigid body dynamics thanks to their relative simplicity and low computational cost.

   The proposed solution adds forces and projections that are governed by an underlying visual task, allowing the ragdoll structure to interact with images and video sequences. In this work, this scheme is applied to object tracking, so the sentient ragdoll is able to track objects in video sequences while

trying to maintain its shape. The adopted radgoll physics system has the following characteristics:

(a) Internal constraints (links between particles) define the shape of the structure. External constraints (forces and projections created by an underlying visual task) move and deform the structure in the space created by digital images. Deformed structures recover their original shape as internal constraints are satisfied.

(b) Each particle's visual task is self-reliant. Particles composing a sentient ragdoll try to fulfill their own objective independently, and thus the structure may contain particles with heterogeneous tasks (e.g. some particles may act as object detectors while other particles act as trackers). Once again, given their individuality, particles' visual tasks are highly parallelizable.

(c) The Verlet integration method avoids particles from building up velocity, so no errors are accumulated. That way, particles that are projected to a new location simply adjust their velocity accordingly. This also allows the system to converge to the correct state over consecutive time steps, correcting positions incrementally. It is especially useful in video sequences, where the structure evolves from frame to frame.

(d) In the proposed tracking system, particles store their target's description as a collection of color planar patches. Tracking becomes a pattern matching process where templates are updated using contextual information. However, even a sensible template updating mechanism is prone to errors in a pre-categorical system, where it is difficult to obtain a measure of the tracking quality.

(e) The shape-preserving ability of ragdolls allows the structure to push particles to their right relative position within the structure no matter how much it is deformed, recovering its original shape. Increasing degrees of rigidity can be obtained simply increasing the number of iterations in the constraint solver process.

5. An in-depth study of decentralized tracking systems, using the two previously proposed contributions: a swarm of free roaming tracking particles subject to steering behaviors and a swarm of linked tracking particles that manages to preserve its shape (i.e. a sentient ragdoll). While both systems are able to track their targets in a wide variety of situations, it must be concluded that pre-categorical object trackers may lose their targets due to a variety of unavoidable circumstances: sudden velocity changes, complete occlusions, appearance changes or background object similarities, to name a few.

   While decentralized tracking systems offer many advantages in these situations, it is not trivial to tell apart when a pre-categorical tracking system is still correctly following its target. Therefore, no such system can be reliable under uncontrolled working conditions for long time periods. Fortunately, both proposed swarming methods are general enough to be used in many different settings.

7.3 Future Work

There are still many lines of work that deserve attention:

1. Alternative pathway exploration: in their tracking process, ragdoll particles examine their local neighborhood looking for the location of their target, and then they move to this position. This location is currently unique, as particles choose the optimum point in the search window according to the template matching score. However, many local optima can be found, although most of them are currently ignored. These local optima represent objects in the search window which appearance is similar to the tracked object's appearance, something which is exploited by the template updating mechanism. But all these local optima could also be considered alternative locations of the tracked object. These optima could be considered in order to explore alternative target locations. New tracking particles could be spawned at these points and, in order to keep a limited number of

individuals, particles could be killed under certain conditions (e.g. age, health or local population density). That way, particle colonies would evolve in a life-death process that would ideally keep them over the tracked object.

2. Use of automatic discriminative features extraction: instead of using a predefined set of features to describe the tracked object, tracking particles could decide at each time step which features differentiate the tracked object from the surrounding context. This ability would, ideally, allow particles to adapt to changing conditions.

3. Consider new constraint types in the ragdoll model. Currently, only distance constraints are considered, but new types like angular and soft constraints could be easily included (i.e. constraints that limit angles between particle pairs and constrains that are only allowed to solve a portion of the suffered deviation at each time step).

4. Compound swarms. Dozens of independent tracking particles interact to create a single tracking entity. Could dozens of these entities interact to create a superorganism? Moreover, such a superorganism could be composed of heterogeneous entities, that is, entities with different goals, procedures and rules. For example, some entities could be devoted to feature or object detection while others could be specialized in tracking.

   The two proposed swarming solutions can be combined to create compound swarms in four different ways: swarm of swarms, ragdoll of swarms, ragdoll of ragdolls and swarm of ragdolls. While the first three options offer complex structures with increasing degrees of rigidity and deserve special attention, preliminary results with the fourth one indicate certain structural instability.

5. New applications for the sentient ragdolls. These structures could also be used to analyze gestures and poses of articulated or deformable objects like faces, hands and human bodies. Predefined ragdoll structures that mimic the studied object shape (e.g. a human-like skeleton or constraints defined to shape a hand or a face) combined with acquired knowledge

of the considered object seems a promising line of work. Like Natural Motion's Dynamic Motion Synthesis and similar techniques (Shapiro et al., 2007) (Arikan et al., 2003) (Ngo and Marks, 1993), which imbue ragdolls with artificial intelligence to give them self-awareness and self-preservation behaviors, sentient ragdolls could be trained to model the dynamics and appearance of the object they represent.

6. In general, the application of Swarm Intelligence to computer vision problems should be studied with more detail, considering its use in segmentation, detection, identification, image stabilization, homeostatic processes and many other visual tasks.

Part II

Inteligencia de Enjambres en Visión por Computador. Aplicación al Seguimiento de Objetos

8 Inteligencia de Enjambres en Visión por Ordenador

8.1 Objetivos

El objetivo de la presente tesis consiste en la estudiar la aplicación de la Inteligencia de Enjambres a la Visión por Ordenador, concretamente al seguimiento precategórico de objetos en secuencias de video. Los objetivos de la tesis incluyen los siguientes puntos:

1. Presentar una breve introducción de la Inteligencia de Enjambres y a la Visión Artificial.

2. Revisar el estado del arte del seguimiento de objetos en secuencias de video. Se analizarán las definiciones de objetos en el espacio bidimensional que constituyen las imágenes digitales, incluyendo sus diversas representaciones, descriptores y distintas técnicas de detección. Se revisaran técnicas de seguimiento de objetos según su representación, incluyendo el seguimiento de puntos, de núcleos (kernel), de siluetas,

contornos y modelos flexibles.

3. Presentar una introducción a la vida artificial y a los modelos de enjambres artificiales aplicados a la resolución de problemas numéricos, cubriendo técnicas como los algoritmos genéticos, la optimización por colonia de hormigas y la optimización por enjambres de partículas. Revisar el estado del arte de la aplicación de técnicas de enjambre a la visión por ordenador.

4. Diseñar e implementar dos sistemas basados en enjambres capaces de seguir objetos en secuencias de video, abordando cada sistema de manera distinta pero complementaria: enjambres de partículas libres y enjambres de partículas sujetas a restricciones estructurales.

5. Evaluar los sistemas implementados sobre secuencias de video significativas que representen los problemas habituales a los que se enfrenta cualquier sistema de seguimiento de objetos (cambios de iluminación, ocultamientos parciales y globales del objeto seguido, cambios de escala, etc.). Comparar su ejecución con los datos aportados por la anotación manual de la localización y tamaño del objeto a seguir en cada secuencia.

6. Presentar las conclusiones obtenidas durante el trabajo de tesis y las líneas de trabajo futuro.

8.2 Planteamiento

La conocida como Inteligencia de Enjambres aparece bajo diversas formas tanto en la cultura popular (libros, películas y videojuegos) como en entornos de investigación y desarrollo. El concepto, surgido a partir del comportamiento de animales eusociales como las termitas y las hormigas, propone formas de inteligencia colectiva emergente a partir de la interactuación entre entidades relativamente simples. Como grupo, consiguen abordar y resolver problemas a los que, como individuos, difícilmente encontrarían solución. La sinergia es el concepto que resume este tipo de actividades, que pueden ser explicadas como ”El todo es mayor que la suma de sus partes”.

A pesar de que las soluciones numéricas basadas en Inteligencia de Enjambres fueron propuestas hace más de dos décadas, es ahora cuando empiezan a ser aplicadas en campos diversos. Puesto que los individuos que forman un enjambre realizan su actividad de manera independiente, los enjambres artificiales son altamente paralelizables, lo cual resulta muy adecuado para ser implementado en los actuales ordenadores multinúcleo. Resulta igualmente atractiva la relativa simplicidad de cada miembro de un enjambre, ya que su diseño puede resultar más abordable que el de otras metodologías monolíticas y complejas.

El uso de estas técnicas en visión por ordenador parece limitado a su aplicación en problemas de optimización, con algunas excepciones puntuales. Dado que este campo está abierto a la experimentación, gracias en parte a la inexistencia de un modelo único de visión artificial, la adopción de técnicas basadas en enjambres puede dar lugar a soluciones originales a problemas

conocidos.

Este trabajo aborda el problema del seguimiento de objetos en secuencias de video desde la perspectiva ofrecida por la Inteligencia de Enjambres. Siguiendo como máxima la simplicidad de cada individuo y su independencia frente al resto de componentes del grupo, independencia relativa al procesamiento de la información pero no a su actividad 'social', se proponen dos soluciones basadas en enjambres. Se mostrará cómo el comportamiento que emerge de la interacción entre individuos va más allá de las capacidades individuales.

8.3 Metodología

8.3.1 Seguimiento de Objetos

El seguimiento de objetos consiste en la localización continua y autónoma de entidades visuales según evolucionan en una secuencia de vídeo, lo cual es un problema fundamental en la Visión por Ordenador. Si bien el seguimiento en entornos controlados o conocidos puede ser considerado un problema resuelto, una solución general de seguimiento todavía continúa siendo una tarea desafiante. El seguimiento de objetos es por lo tanto un tema ampliamente investigado en la actualidad.

El seguimiento es principalmente una actividad de bajo nivel: los objetos pueden ser seguidos sin importar su categoría. Los bebés son capaces de seguir objetos incluso antes de aprender qué son o para qué sirven, utilizando para ello pistas de bajo nivel como el color, la textura y el movimiento. Una vez un objeto es reconocido, la acción de seguimiento puede aprovechar información disponible como el tipo de movimiento que puede realizar el objeto o los distintos aspectos que puede presentar según el punto de vista.

En este trabajo se despreciará la carga semántica de los objetos. Al resultar una actividad visual instintiva resulta especialmente interesante replicarla para poder ser utilizada en cualquier contexto, sin necesidad de recabar información previa del objeto a seguir. No se considerarán clases conocidas y por lo tanto no podrá utilizarse información conocida de antemano, adoptando un enfoque precategórico. Diversas técnicas permiten localizar objetos de interés en imágenes

sin conocer nada sobre ellos. Consisten principalmente en la detección de puntos o regiones que, por sus características, son claros candidatos a contener información relevante sobre la imagen. Estas técnicas incluyen la detección de puntos y regiones de interés, el análisis de la saliencia visual, la substracción de fondos una vez modelados y la segmentación de regiones a partir de características de bajo nivel, entre otras.

Así pues, para permitir que un ordenador pueda tratar de seguir objetos visuales en las secuencias de vídeo obtenidas a través de dispositivos de entrada de imágenes (sean reproducciones de videos o cámaras que alimenten al sistema de un flujo de imágenes en tiempo real) es necesario definir qué es un objeto en una imagen.

No existe una única definición de qué es un objeto en el espacio proximal (esto es, el espacio en el que guarda una proyección del mundo real, o espacio distal, creada por un sensor visual). Si acaso, podría ser ”cualquier cosa que resulte de interés para la tarea a realizar”. Se utilizarán diferentes representaciones dependiendo en gran medida del grado de rigidez del objeto de interés y de si éste se define como un único elemento o como la suma de múltiples partes. Si bien los objetos rígidos requieren descripciones simples, la mayor parte de los objetos en el mundo real muestran movimiento no rígido.

El movimiento no rígido es clasificado generalmente en tres grupos: el movimiento de partes rígidas, en el que las partes de un objeto se mueven independientemente unas de otras (categorizado como movimientos articulados), el movimiento coherente y movimiento fluido. En las simulaciones por ordenador, el movimiento coherente suele ser estimado a través de cuerpos rígidos articulados

discretos compuestos por múltiples partes de tamaño infinitesimal.

El saber cómo evoluciona un objeto en una secuencia de video, si éste es rígido o elástico, dictamina cómo puede ser descrito. La representación más simple de un objeto visual en un espacio bidimensional es un único punto, que puede coincidir con el centroide del objeto. Formas geométricas simples (como rectángulos o elipses) pueden ser usadas para definir regiones de la imagen alrededor de puntos específicos, que pueden abarcar todo el objeto o representar partes distintivas a diferentes escalas. Pueden considerarse transformaciones afines para adaptar estas formas a rotaciones, cizalla, traslaciones o escalados. Una colección de puntos y/o regiones puede ser combinada para definir objetos compuestos. Para tratar con objetos deformables, muchos trabajos consideran sus siluetas, contornos y esqueletos dada su versatilidad para describir formas. La Figura 8.1 muestra algunos ejemplos de estas representaciones.

Una vez establecido el tipo de movimiento que sufre el cuerpo a analizar y como será definido, el objeto se describe por su apariencia, utilizando características extraídas de las imágenes como la información cromática en distintos espacios de color, las magnitudes y orientaciones del gradiente, descriptores de textura o movimiento; y por su relación con otros objetos, creando estructuras complejas. La definición del objeto dictamina qué técnicas serán usadas para trabajar con él.

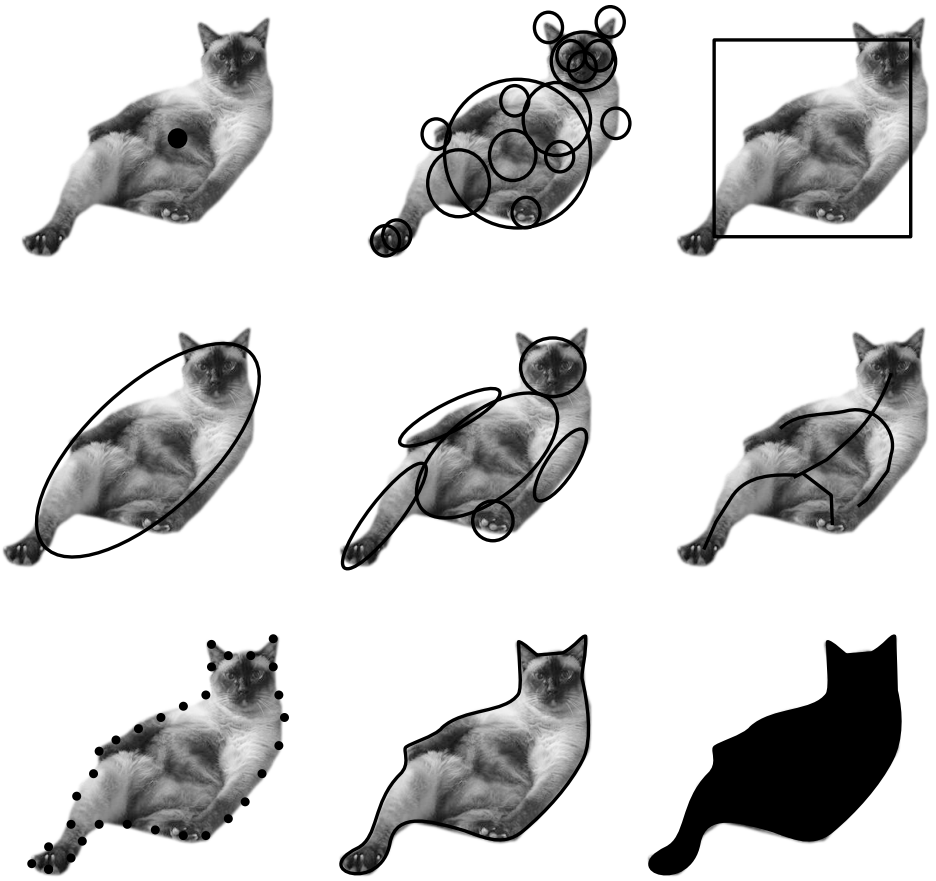


Figure 8.1: Representación de objetos bidimensionales en el espacio proximal.

8.3.2 Enjambres Artificiales

La habilidad de las criaturas que forman enjambres para lograr realizar tareas complejas sin importar la simplicidad de cada individuo ha inspirado numerosas soluciones numéricas basadas en enjambres en campos muy diversos. La Inteligencia de Enjambres, expresión introducida en 1989 en el contexto de los sistemas celulares robóticos, es considerada un tipo de inteligencia basada en el comportamiento colectivo de sistemas descentralizados y auto-organizados. La Inteligencia de Enjambres puede ser representada por el concepto de Sinergia,

entendida como ”la interacción o cooperación de dos o más organizaciones, substancias u otros agentes para producir un efecto combinado mayor que la suma de los efectos por separado”.

Sin embargo, antes de que la habilidad de las colonias de insectos para resolver problemas de optimización fuera modelada y aplicada a problemas numéricos, diversos modelos ya mostraban habilidades de auto-organización, evolución y comportamientos emergentes.

A finales de 1940 el propio John von Neumann postuló la posibilidad de crear vida artificial utilizando ordenadores y desde entonces numerosos sistemas imitan de una forma u otra, a distintos niveles, comportamientos similares a los mostrados por criaturas y ecosistemas biológicos. La Vida Artificial, como se denomina este campo de estudio, trata de recrear en los ordenadores sistemas orgánicos que abarcan desde complejos ecosistemas donde interactúan diversos organismos a la simulación de la genética de criaturas digitales. Uno de sus objetivos es lograr sistemas que, al menos en apariencia, muestren comportamientos similares a los modelos biológicos en los que se inspiran.

Basándose en las propuestas de John von Neumann sobre la Vida Artificial, John Conway creó uno de los primeros autómatas celulares en el que la interacción entre entidades simples según unas reglas de comportamiento muy sencillas daba lugar a estructuras complejas y difíles de predecir (ver Figura 8.2).

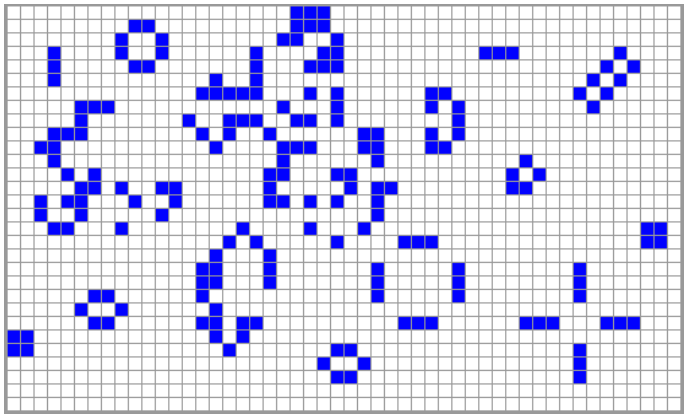
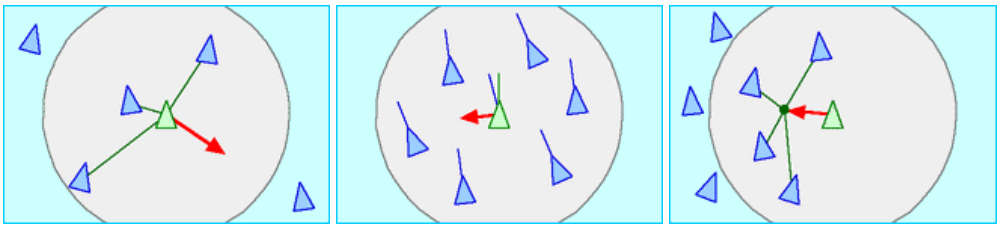


Figure 8.2: El Juego de la Vida, de Conway.

Boids

No siempre es necesario construir complejos sistemas para lograr imitar comportamientos realistas. En 1986, Craig Reynolds propuso un modelo computacional de movimiento coordinado en grupo independiente del sistema de locomoción de los individuos. Llamó a sus criaturas Boids, y sus habilidades básicas de agrupamiento estaban definidas por tres simples reglas denominadas Cohesión, Alineamiento y Separación. Estas reglas básicas fueron posteriormente apoyadas por experimentos en peces reales. Gracias a ellas, cada boid era capaz de maniobrar según las posiciones y velocidades de sus compañeros de grupo, así como de reaccionar a obstáculos del entorno.

En el modelo de Boids, las reglas de movimiento se definen mediante vectores (ver Figura 8.3). La regla de cohesión define un vector de un boid hacia el centroide de los compañeros cercanos de bandada, lo que mantiene al grupo unido. La regla de Alineamiento define un vector que es el promedio de las velocidades de los compañeros cercanos, lo que permite al grupo moverse en una dirección



(a) (b) (c)

Figure 8.3: Los Boids de Reynolds siguen tres comportamientos direccionales básicos, lo que les permite realizar movimientos en grupo muy realistas.

común. La regla de separación es simplemente un vector desde el centroide de compañeros muy cercanos hacia el boid considerado, lo que evita choques entre miembros del grupo.

Para conseguir el movimiento coordinado del grupo, cada boid no tiene más que calcular y sumar estos tres vectores (y cualquier otro que quiera añadirse como regla de movimiento, como la habilidad para esquivar obstáculos) y utilizar la resultante como velocidad de movimiento.

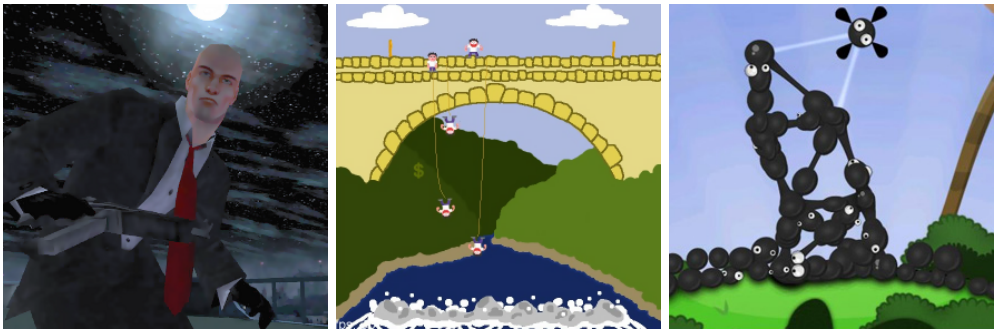
Los boids de Reynolds son tremendamente versátiles: pululan en su entorno virtual y se muestran como un enjambre, una bandada, un banco de peces o un rebaño dependiendo de los pesos asignados a las reglas de movimiento. Por ejemplo, asignar un peso bajo a la regla de alineamiento permite que cada boid siga su propia velocidad, dando el grupo el aspecto de un enjambre. Asignar un peso mayor a esta misma regla hace que los boids traten de imitar las velocidades de sus vecinos, por lo que el grupo se comporta como una bandada de pájaros. Su uso está ampliamente extendido en los medios visuales (cine y videojuegos) cuando se requieren dinámicas de grupo coordinado.

Ragdolls

En algunas ocasiones se requiere emular sistemas articulados complejos, como la estructura de un virus o las propiedades de plegado de una proteína, para lo cual se utiliza la dinámica de partículas sujetas a restricciones. La simulación de la dinámica de partículas sujetas a restricciones está estrechamente relacionada con la industria del videojuego, donde muchas veces se requiere simular la física de cuerpos rígidos. Thomas Jakobsen estableció, con el sistema creado para *Hitman: Codename 47*, de IO Interactive, el núcleo de un motor de física para videojuegos centrado en la estabilidad y velocidad de ejecución (ver Figura 8.4).

La versatilidad de estos sistemas ha permitido su aplicación en muy diversos contextos. Tan sólo en el campo del videojuego encontramos ejemplos diversos: muchos elementos en *Hitman: Codename 47*, de IO Interactive, desde la corbata del personaje principal a los cadáveres de sus víctimas, utilizan el sistema de física Ragdoll adoptado en el presente trabajo (ver Figura 8.4a). Fue desarrollado por Jakobsen específicamente para el juego. En *Bungee Manager* (prototipo creado por el autor de este trabajo), las cuerdas de puenting son simuladas mediante Ragdolls (ver Figura 8.4b). En *World of Goo*, de 2DBoy, el usuario puede crear complejas estructuras colocando criaturas capaces de ligarse entre sí, luchando contra la gravedad para alcanzar diversos objetivos. Es un claro ejemplo de puzzle basado en la física que requiere de un motor fiable para su ejecución (ver Figura 8.4c).

Definiendo un cuerpo articulado como un conjunto de partículas y enlaces en una estructura similar a un grafo, Jakobsen propuso un método



(a) a (b) b (c) c

Figure 8.4: Ragdolls en videojuegos

sorprendentemente simple para simular la dinámica de cuerpos articulados complejos basándose en un esquema de integración Verlet, solucionando las restricciones de forma iterativa. Este método, conocido como 'muñecos de trapo' (ragdolls), es capaz de manejar la simulación de cientos de cuerpos rígidos interactuando en tiempo real, desde ropa y pelo a plantas y cuerpos humanos, por lo que numerosos motores de física para dos y tres dimensiones ya incorporan soluciones semejantes.

Los ragdolls pueden ser considerados, en el contexto del presente trabajo, como enjambres cuyas partículas están sujetas a restricciones estructurales. A diferencia de un modelo de Boids, en el que las partículas se mueven libremente, en los modelos Ragdoll las partículas siempre guardan la misma posición relativa dentro de la estructura. Por lo tanto, son sistemas capaces de recordar y mantener una forma concreta, como pueda ser un esqueleto humano o la forma de una mano, y recuperarla tras sufrir deformaciones.

8.3.3 Computación bioinspirada

La computación bioinspirada se apoya en los campos de la biología, la informática y las matemáticas, y está íntimamente relacionada con la inteligencia artificial (IA), abarcando el conexionismo, los comportamientos sociales y la emergencia. A diferencia de la IA clásica, donde los problemas son resueltos desde la perspectiva del programador, quien depende de sus conocimientos y habilidades para diseñar algoritmos adecuados, la computación bioinspirada aplica soluciones que pueden encontrarse en la naturaleza a problemas similares al tratado utilizando el principio de analogía.

Computación evolutiva

La propia evolución como sistema es capaz de encontrar soluciones óptimas adaptadas y específicas. Un importante campo de la computación bioinspirada lo constituye la computación evolutiva, en la que la programación y algoritmos genéticos utilizan los mismos principios que la evolución para buscar soluciones a problemas concretos: la selección, la reproducción y la mutación. En la Figura 8.5 se muestra una de las criaturas digitales creadas por Karl Sims mediante evolución. En el entorno propuesto por Sims, las criaturas deben evolucionar con el único objetivo de avanzar todo lo posible. Los distintos actuadores de sus cuerpos crean mecanismos de locomoción que, tras sucesivas generaciones de criaturas de las que solo las mejores sobreviven, se adaptan para optimizar el tipo movimiento en el medio en el que viven.

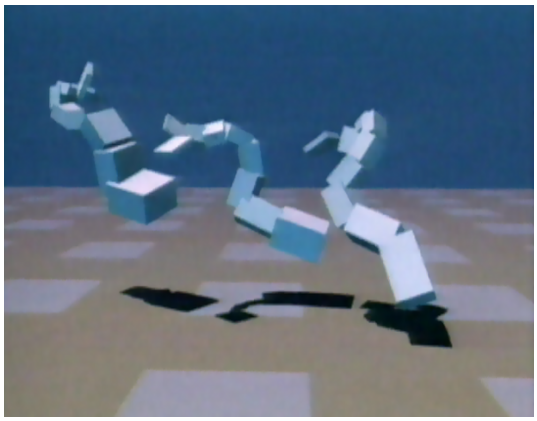


Figure 8.5: Criatura digital surgida de la adaptación al medio virtual en el que existe.

Optimización por colonia de hormigas

En el caso de la Inteligencia de Enjambres, nos encontramos con que es precisamente optimizar lo que los animales eusociales realizan contante y eficientemente: desde la asignación de tareas a encontrar el camino mínimo entre dos puntos, los insectos se basan en sus enormes poblaciones y control descentralizado para encontrar soluciones óptimas. La Optimización por Colonia de Hormigas fue propuesta por Marco Dorigo en 1992, y todavía hoy se estudia y mejora constantemente. Puede ser considerada una de las primeras aplicaciones de la Inteligencia de Enjambres a la resolución de problemas.

En su trabajo, Dorigo creaba hormigas virtuales capaces de recorrer grafos, impregnándolos de feromonas en el proceso, proponiendo una técnica probabilística de resolución de problemas que se reducía a la búsqueda de caminos mínimos en estos grafos. En la actualidad, el uso del intercambio de información entre agentes utilizando el entorno (una forma de sinergia denominada estigmergia) es suficiente para que un algoritmo pertenezca a la clase

de los algoritmos de colonias de hormigas.

Optimización por enjambres de partículas

En 1995 James Kennedy y Russell C. Eberhart propusieron otro algoritmo estocástico basado en poblaciones para la resolución de problemas: la optimización por enjambres de partículas (PSO). Este algoritmo se inspiraba en el comportamiento social en grupo de animales, así como en el trabajo de Heppner y el ya mencionado Reynolds (Boids). PSO fue diseñado explícitamente para encontrar soluciones a problemas de optimización utilizando principios psico-sociales.

Estos enjambres se modelan como un grupo de partículas con una posición, que representa una solución válida al problema, una velocidad y cierta memoria. La dinámica social permite que cada individuo recorra el espacio de soluciones comunicando sus resultados al resto del enjambre, y gracias a esta colaboración el grupo termina por encontrar una solución adecuada (aunque nada impide el que el óptimo encontrado sea local o que la solución óptima no sea encontrada, como ocurre con otras heurísticas).

La Figura 8.6 muestra la evolución de un enjambre en un entorno tridimensional en el que los individuos buscan el punto más alto. Un enjambre de doce partículas creadas aleatoriamente trata de encontrar el punto máximo de un paisaje tridimensional. Las líneas verdes muestran el movimiento actual, mientras que las amarillas muestran el movimiento en la iteración anterior. Nótese que, aunque este enjambre es capaz de superar los máximos locales, esta habilidad

depende de la implementación utilizada. Aun asi, los sistemas PSO son muy versátiles y su adaptación a cada problema es relativamente sencilla.

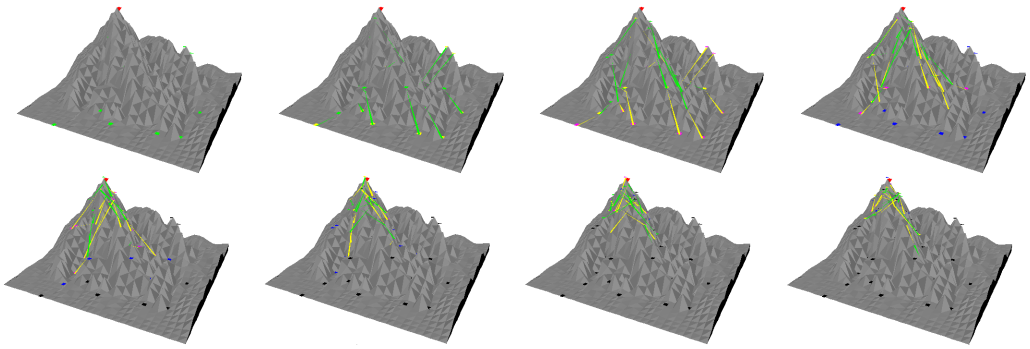


Figure 8.6: Particle Swarm Visualization

Comparados con los algoritmos genéticos y los sistemas de optimización por colonias de hormigas, los sistemas basados en la optimización por enjambres de partículas son mucho más sencillos de diseñar e implementar, lo que ha permitido que sigan siendo un campo de investigación muy activo. Aunque su primera aplicación práctica fue en el campo de las redes neuronales, PSO se utiliza actualmente en aplicaciones muy diversas, incluyendo las telecomunicaciones, la minería de datos, el diseño, el procesamiento de señales, entre otras.

Project Computing, 2004. http://www.projectcomputing.com/resources/psovis/index.html

8.4 Aportaciones Originales

Se resumen a continuación las dos propuestas originales contenidas en el presente trabajo:

8.4.1 Modelo de enjambre de partículas libres: Boids

La primera aportación original de este trabajo consiste en una solución basada en la Inteligencia de Enjambres en la que el problema del seguimiento de objetos en Visión por Ordenador se modela como una metáfora depredador-presa en el ecosistema creado por una secuencia de video. Inspirado por la actividad individual de los insectos sociales y los modelos de comportamiento distribuido, el seguimiento es entendido como una propiedad emergente de un enjambre de partículas depredadoras cuando éstas persiguen a sus píxeles-presa a través de los cuadros de la secuencia analizada.

Las partículas depredadoras buscan a sus presas guiadas por su aroma, entendido como una combinación de características de la imagen que cada partícula guarda como objetivo. El sistema no requiere ni una plantilla visual completa del objetivo a seguir ni un modelo de su dinámica. Siguiendo las mismas reglas de dinámica coordinada de grupo planteadas por Reynolds para sus Boids, las partículas vuelan en el espacio bidimensional creado por las imágenes digitales.

Comportamientos direccionales

Las siguientes expresiones definen cómo un Boid se mueve en un sistema con n reglas:

$$\begin{aligned} \vec{v_p}(t) &= \omega_0 \cdot \vec{v_p}(t-1) + \\ &\quad \omega_1 \cdot (\vec{rule}_1) + \\ &\quad \omega_2 \cdot (\vec{rule}_2) + \\ &\quad \omega_3 \cdot (\vec{rule}_3) + \\ &\quad \ldots \\ &\quad \omega_n \cdot (\vec{rule}_n) \end{aligned} \tag{8.1}$$

$$\vec{x_p}(t) = \vec{x_p}(t-1) + \vec{v_p}(t). \tag{8.2}$$

donde:

- $v_p(t),\ v_p(t-1)$ son la velocidad del boid en la iteración actual y previa.
- $x_p(t),\ x_p(t-1)$ son la posición del boid en la iteración actual y previa.
- $\vec{rule}_1..\vec{rule}_n$ son comportamientos direccionales, vectores calculados para satisfacer ciertas condiciones: cohesión, separación, alineamiento, seguimiento de un líder, movimiento aleatorio...
- $\omega_1..\omega_n$ son los pesos que controlan la influencia de cada regla.

A las reglas que definen los vectores de cohesión, alineamiento y separación de los Boids clásicos se añade una cuarta, la caza, que guía a cada partícula

hacia las regiones cercanas en las que las presas sean más parecidas a los gustos de cada depredador (ver Figura 8.7). Formalmente, dado un enjambre de partículas $\mathbb{S}$ que habita una secuencia de video de la cual se extraen un conjunto de mapas de características de imagen $\vec{F}$, cada partícula p es una 4-tupla: $p =< \vec{x}_p,\ \vec{v}_p,\ c_p,\ \vec{sl}_p >, p \in \mathbb{S}$ donde:

1. $\vec{x}_p$: Posición en el espacio de la imagen en el instante actual de la partícula p.

2. $\vec{v}_p$: Velocidad de la partícula.

3. c_p: Confort de la partícula, una medida de la calidad del seguimiento en el instante de tiempo actual, obtenida de la mejor presa en la vecindad de la partícula. Tendrá un valor entre 0.0 (peor) y 1.0 (mejor), y su cálculo depende del método de seguimiento empleado.

4. $\vec{sl}_p$: Lista de los aromas registrados por la partícula como característica de la presa a perseguir, obtenidos de $\vec{F}$ (intensidad, color, gradiente, textura, movimiento...)

   La Figura 8.7 muestra cómo una partícula (representada como un punto rojo) analiza su vecindad buscando presas que satisfagan sus gustos. El tamaño de las flechas en la imagen de la derecha representa el interés que cada región alrededor de la partícula despierta en ésta. La resultante de la regla de caza, representada como una flecha verde, apunta a la región que contiene el mayor número de presas interesantes.

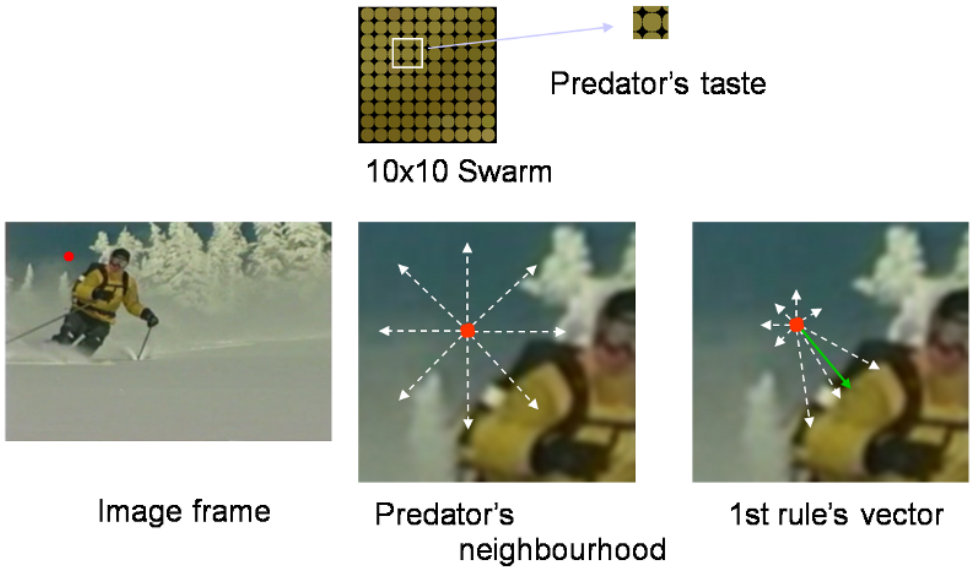


Figure 8.7: Regla 1: Caza.

Seguimiento

El modelo adoptado permite que cada partícula implemente su propio sistema de seguimiento, al contrario que en PSO donde todas las partículas deben ser homogéneas. Así, es posible la creación de enjambres cuya atención puede abarcar una amplia variedad de características de la imagen y distintos comportamientos dentro del mismo enjambre. Se consideran varias combinaciones de características de la imagen como aromas para las presas: color, intensidad del gradiente, bordes, puntos de interés, cantidad de movimiento y texturas.

Gracias a la independencia del proceso de caza (seguimiento), esta función es altamente paralelizable, lo que permite la creación de enjambres densos en equipos multinúcleo.

En la implementación del modelo se adoptó un esquema de búsqueda local en ventanas. La regla de caza de cada partícula busca en una vecindad cercana la zona que contenga las presas más atractivas.

$$\vec{V}_1^p = \frac{\sum_q (\vec{x}_q - \vec{x}_p) \cdot \beta_{qp}}{\sum_q \beta_{qp}} \forall q \in N \tag{8.3}$$

donde

- $\vec{x}_q$ es la posición de una presa q
- β_{qp} representa el interés que la presa q despierta en el depredador p, esto es, la semejanza ente la lista de características a seguir por la partícula p, $\vec{sl}_p$ y las características de un píxel concreto. Su cálculo dependerá de las funciones de distancia definidas entre las características usadas.

Puesto que los comportamientos direccionales no imponen restricciones estructurales, los miembros del grupo tienen total libertad para seguir a sus presas (siempre dentro de los parámetros del comportamiento social). El sistema propuesto es por lo tanto capaz de seguir objetos altamente deformables en secuencias de caracteríticas muy variadas, moviéndose sobre el objeto imitando la dinámica de un enjambre de insectos voladores.

Red de detección de características

En el cuadro inicial de una secuencia de video el enjambre es colocado sobre el grupo de píxeles presas a ser seguido. Cada partícula extrae y guarda un conjunto

de características de la región en la que es creado. Estas características de la presa son calculadas utilizando una Red de Detección de Características compuesta por una variedad de Detectores de Características FD_i que extraen y combinan rasgos de la imagen para crear una serie de mapas de características $\vec{F} = f_1, f_2..f_m$ de los cuales el enjambre extrae información de sus presas (ver Figura 8.8). Sin pérdida de generalidad, estos mapas son normalizados al rango $0..1$.

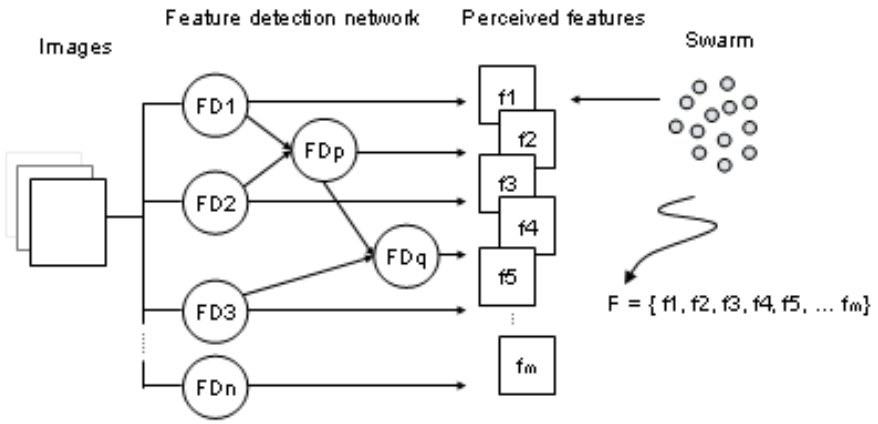
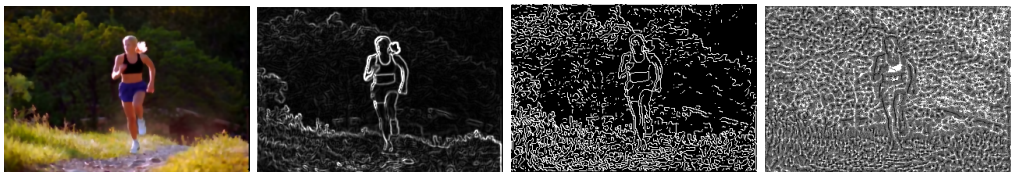


Figure 8.8: Las partículas del enjambre extraen información de los mapas de características, calculados a partir de las imágenes o creados por las propias partículas.

Estos rasgos caracterizan el *aroma* de una presa, que es una abstracción de la combinación de características transformadas que la partícula perseguirá. El valor de intensidad de un píxel podría ser considerado el *aroma* más simple, pero muchos otros rasgos pueden ser utilizados (ver Figura 8.9). Los *aromas* de las presas pueden ser además modulados, modificando sus intensidades mediante el uso de otros rasgos: magnitudes del gradiente, movimiento, puntos salientes o mapas de feromonas creados por las propias partículas. Así, los miembros del enjambre persiguen presas que despiden un cierto *aroma*, que se vuelve más atractivo al cumplir ciertas condiciones (por ejemplo, la presa se sitúa en una

región donde se ha detectado movimiento o donde los depósitos de feromonas tienen un nivel de concentración mayor).



(a) (b) (c) (d)

Figure 8.9: Aromas utilizados por las partículas de seguimiento: a) color RGB, b) gradientes Sobel, c) cruces por cero de la Laplaciana normalizados, d) códigos LBP simplificados.

Pesado de las reglas

Se consideran dos conjuntos de pesos para las reglas, obtenidos empíricamente:

- $w_1(Caza) = 1.0$
- $w_2(Cohesion) = 0.3$
- $w_3(Alineamiento) = 0.5$
- $w_4(Separacion) = 0.01$
- $w_0(Inercia) = 0.1$
- $n(Tamano\ de\ la\ vecindad\ (pixeles)) = 11$

Aunque los pesos propuestos funcionan adecuadamente en la mayor parte de las situaciones encontradas, puede argumentarse que establecer estos valores manualmente es una solución ad hoc. En lugar de tratar de encontrar el conjunto

óptimo de pesos para cada situación, se adopta un enfoque completamente estocástico. Cada partícula, en cada instante de tiempo, cambiará los pesos de sus reglas continuamente.

El uso de valores aleatorios procedentes de ruido blanco provocaría comportamientos erráticos en las partículas, lo que a su vez repercutiría en la calidad del seguimiento al no dar tiempo al enjambre a adaptarse. En su lugar, las partículas obtienen los nuevos pesos a partir de los valores interpolados entre dos puntos aleatorios, creando una función de ruido suave mediante una onda definida por partes de frecuencia y amplitud aleatoria. En la fila superior de la Figura 8.10 se muestra una señal de ruido blanco, demasiado dura para parecer natural en este contexto. En la fila inferior de la misma figura se muestra una señal aleatoria cuyos valores se obtienen de la interpolación por coseno entre puntos aleatorios (marcados con círculos azules) definidos a lo largo de la dimensión de la función.

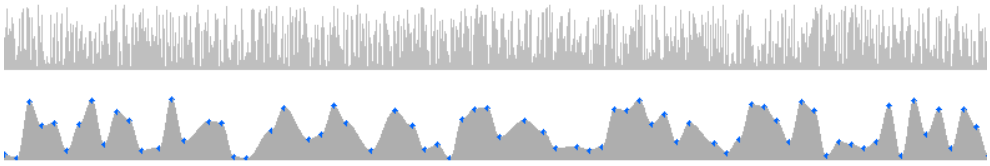


Figure 8.10: Ruido blanco y Onda de ruido.

Al considerar el uso de pesos aleatorios con este tipo de evolución para ponderar las reglas de movimiento se logra que las partículas experimenten cambios de tendencia en su comportamiento durante el seguimiento de manera suave (por ejemplo, ciertas partículas pueden 'decidir' alejarse del grupo momentáneamente, volviendo a agruparse momentos después). La Figura 8.11 muestra varios instantes de una secuencia en la que un enjambre persigue un

objeto de movimiento rápido.

Figure 8.11: Enjambre de partículas siguiendo a un objeto en una secuencia de vídeo.

Diferencias con la optimización por enjambres de partículas (PSO)

La presente solución difiere de los modelos PSO, aunque ambos métodos comparten las mismas raíces: el comportamiento social de grupos y los trabajos de Reynolds y Heppner. Sin embargo, mientras que PSO realiza búsquedas globales en el espacio de soluciones, el seguimiento basado en boids realiza búsquedas locales.

Las partículas en PSO representan soluciones que evolucionan en el espacio de soluciones, mientras que las partículas boid vuelan en el espacio bidimensional creado por imágenes digitales. Esta característica permite que las partículas boid persigan metas heterogéneas. Los comportamientos direccionales siguen aplicándose, así que aunque cada partícula tenga una meta diferente, siguen perteneciendo al mismo enjambre. Las partículas de PSO que pertenecen a un mismo enjambre deben ser homogéneas por definición, ya que su posición representa una solución al problema.

Las soluciones PSO requieren un cierto número de interacciones antes de encontrar una solución única. Por lo tanto, al ser aplicadas a secuencias de vídeo, las soluciones PSO deben completar las iteraciones antes de avanzar al siguiente cuadro. Más aún, aunque varias soluciones proponen adaptar PSO a entornos dinámicos, las partículas suelen ser forzadas a olvidar sus posiciones (soluciones) anteriores. El seguimiento basado en boids funciona de manera continua, con partículas que adaptan su posición de cuadro a cuadro.

El seguimiento emerge del estado de todas las partículas, localizando el objeto en el centroide del enjambre y estimando su velocidad de la velocidad

promedio ponderada de las partículas del enjambre. PSO confía en la bondad de una única partícula, la solución óptima, que puede no ser la mejor solución. Ambas soluciones difieren completamente en este aspecto: mientras que las partículas boids evolucionan independientemente, optimizando su confort individual, las partículas PSO deben converger a un único punto.

Otras consideraciones

Otros procesos pueden ser introducidos en el esquema de Boids propuesto. Entre ellos podrían considerarse los siguientes:

- Filtrado de imágenes: Puesto que las secuencias utilizadas fueron obtenidas con cámaras de poca calidad, o descargadas de servicios online en Internet, muchas mostraban ruido perceptible y artefactos debidos a la compresión. Se barajaron tres filtros distintos: un fuerte filtrado gaussiano, un filtro pequeño de mediana y un filtro bilateral con amplio rango de frecuencias. De los tres, el último parece el más adecuado de manera general, ya que consigue suavizar gran parte del ruido a la vez que preserva los bordes originales de la imagen.

  En la Figura 8.12 se muestra el efecto de tres tipos de filtros. La imagen de la izquierda (a) es filtrada utilizando un filtro gaussiano de apertura 11 (b), un filtro de mediana de apertura 5 (c) y un filtro Bilateral de apertura 5, rango 64 y frecuencia 3 (c). El filtro bilateral es preferible ya que conserva los bordes originales de la imagen.

- Ventanas de búsqueda ponderadas y medida de *cansancio* de las partículas:

(a) (b) (c) (d)

Figure 8.12: Filtrado de ruido.

Con el fin de evitar que las partículas experimenten saltos bruscos durante el seguimiento, dado que se presupone que el objeto seguido cumple el principio de continuidad, las ventanas de búsqueda podrían ser ponderadas para que los saltos largos sean más costosos que los cortos. De igual forma, las partículas podrían experimentar *cansancio*, sufriéndolo al realizar saltos largos.

- Mapas de feromonas: una característica interesante podría consistir en un mapa de feromonas similar al creado en los enfoques ACO. Durante el seguimiento, cada boid podría depositar en la trayectoria seguida una cantidad de feromonas proporcional a la medida de calidad del seguimiento (el confort de la partícula). Depósitos consecutivos incrementarán la acumulación de feromonas en esa zona, aunque con el tiempo acabará por evaporarse. Así, las partículas podrían incorporar una nueva regla para sentirse atraídas hacia las regiones con altas concentraciones de feromonas, manteniendo al grupo cohesionado (ver Figura 8.13).

- Búsquedas aleatorias: para permitir al enjambre la exploración de áreas más amplias en caso de perder su objetivo, pueden introducirse búsquedas

(a) (b) (c) (d)

Figure 8.13: Mapas de feromonas en una misma secuencia en distintos instantes de tiempo. Los valores están invertidos por motivos de visualización, por lo que los puntos oscuros representan niveles mayores de concentración de feromonas.

aleatorias como un nuevo comportamiento direccional. Un vector de movimiento aleatorio puede ser añadido a las reglas de comportamiento, con una magnitud inversamente proporcional al valor de confort de una partícula. De esta forma, las partículas que pierdan su objetivo podrán saltar a lugares lejanos para continuar con su búsqueda, en lugar de permanecer atrapadas en algún mínimo local.

Comportamiento seguidor

El comportamiento depredador y las interacciones cooperativas sociales llevan a las partículas hacia aquellas áreas de la imagen que son similares al lugar donde el enjambre fue creado, emergiendo un comportamiento de seguimiento para objetos rígidos y no rígidos donde el centroide del enjambre y su velocidad describen la trayectoria del objeto y su velocidad. El seguimiento es alcanzado mediante la optimización del confort individual: el enjambre resuelve el problema de optimización espacial mediante un algoritmo voraz, maximizando el confort de cada partícula y así minimizando la distancia entre el centroide del enjambre y el objeto seguido.

a) Una sola partícula puede engancharse a píxeles fondo fácilmente...

b) ...pero muchas partículas son capaces de permanecer sobre el objeto.

c) Los comportamientos direccionales mantienen a las partículas sobre el objeto perseguido.

Figure 8.14: Los comportamientos direccionales pueden corregir a las partículas perdidas durante el seguimiento.

Una sola partícula no sería capaz de seguir al objeto durante mucho tiempo. Sería fácilmente atraída hacia píxeles en el fondo cuyas características fuesen similares a las perseguidas por la partícula (ver Figura 8.14 a). Sin embargo, al observar varios docenas de partículas independientes tratando de perseguir a sus presas, resulta aparente que muchas de ellas si son capaces de permanecer sobre el objeto durante periodos de tiempo más largos (ver Figura 8.14 b). Gracias a los comportamientos direccionales, estas últimas son capaces de guiar a las partículas perdidas en la dirección correcta.

El enjambre es capaz de seguir a su objetivo en una gran variedad de condiciones de iluminación y fondos distinta. Debido a la ausencia de rigidez estructural en el patrón perseguido, los objetos pueden ser deformables y escalables. La Figura 8.15 muestra algunas secuencias donde el enjambre persigue sin problemas a sus objetivos.

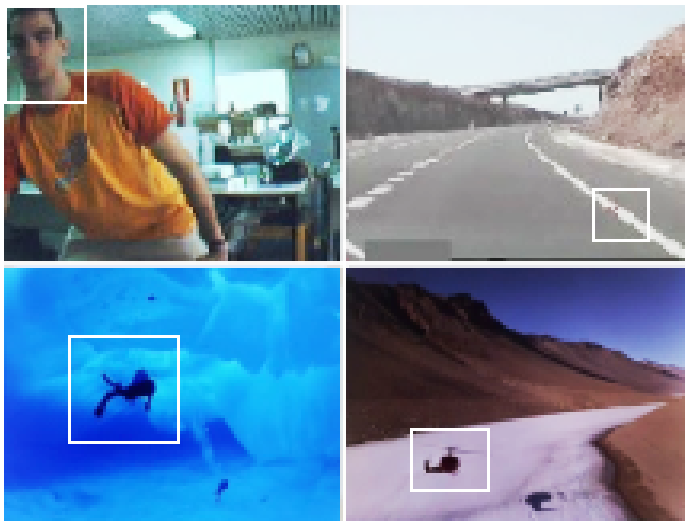


Figure 8.15: Objetos seguidos con éxito, marcados en un recuadro blanco.

Es importante señalar que, debido a la ausencia de restricciones estructurales, el enjambre flota libremente sobre los objetos perseguidos. Si un objeto tiene características homogéneas y es suficientemente grande, el centroide del enjambre vagabundeará por su interior. En el ejemplo de seguimiento de la línea de carretera en la Figura 8.15 el movimiento debe ser restringido verticalmente.

Dado que las partículas de los actuales enjambres no son capaces de actualizar los descriptores de las presas perseguidas, el objeto seguido puede ser perdido si su apariencia sufre cambios cromáticos drásticos (ver Figura 8.16(a,b)). Además, si en la vecindad del enjambre del objeto seguido existieran objetos de apariencia similar (como ocurre con los patos de la Figura 8.16 c), el enjambre podría fácilmente saltar de un objeto a otro, ya que no considera información contextual.

(a) (b) (c)

Figure 8.16: Problemas en el segumiento.

8.4.2 Modelo de enjambre de partículas sujetas a restricciones: Ragdolls

La segunda aportación original de este trabajo consiste en una estructura de cuerpos articulados elástica aplicada al seguimiento en tiempo real de objetos en secuencias de video. Esta estructura puede entenderse como un enjambre cuyos miembros están sujetos a restricciones de distancia entre pares, esto es, un conjunto de partículas y enlaces.

Cada partícula realiza independientemente el seguimiento de un patrón en una secuencia de video, pero forma parte de la estructura creada por el conjunto de partículas y restricciones. La estructura elástica resultante es similar a un grafo y a lo que se conoce como Ragdolls en la industria del videojuego. Los sistemas Ragdoll suelen constar de las siguientes etapas:

- Acumulación de fuerzas: la aceleración de cada partícula es actualizada por todas las fuerzas (gravedad, viento...), si existiesen.
- Cinemática: cada partícula actualiza su velocidad y posición usando el esquema Verlet.

- Restricciones externas: cada partícula sufre una proyección como resultado de la existencia de restricciones externas, si existiesen (por ejemplo, límites del mundo simulado, interacción del usuario, colisiones con otros objetos...)
- Resolución de restricciones: las restricciones internas (enlaces entre partículas) son satisfechas mediante relajación.

Se explican a continuación las distintas etapas en el orden adecuado para la mejor comprensión de esta solución.

Dinámica de partículas

Para lograr la simulación correcta del sistema de partículas que forma los cuerpos rígidos articulados se utiliza el motor propuesto por Jakobsen. En el sistema original, cada partícula debía responder ante restricciones externas (fuerzas y proyecciones) mientras trataba de no violar las restricciones internas (enlaces entre partículas). Para ello, Jakobsen propuso un sistema Verlet sin velocidad, esto es, calculando la velocidad de cada partícula a partir de su posición actual y la inmediatamente anterior:

$$x'_p = 2 \cdot x_p - x^*_p + a_p \cdot \triangle\, t^2 \tag{8.4}$$

donde x'_p es la nueva posición de la partícula p, x_p es su posición actual y x^*_p es su posición en el instante anterior.

Satisfacción de restricciones

Al contrario que los sistemas que utilizan las ecuaciones eulerianas de movimiento, el no usar explícitamente la velocidad otorga numerosas ventajas, ya que las partículas no acumulan errores de velocidad y pueden ser proyectadas en cualquier momento a cualquier posición. De esta manera, las restricciones internas pueden ser satisfechas precisamente proyectando las partículas a las posiciones adecuadas de manera iterativa. Considerando que cada partícula tiene su propia masa, en cada iteración se aplican las siguientes expresiones para satisfacer una restricción de distancia d entre dos partículas p y q:

$$\begin{aligned} \vec{\delta} &= \vec{x}_q - \vec{x}_p \\ \epsilon &= \frac{|\vec{\delta}| - d}{1.0 + |\vec{\delta}| \cdot (m_p^{-1} + m_q^{-1})} \\ \vec{x'}_p &= \vec{x}_p \vec{m}_p^{-1} \cdot \delta \cdot \epsilon \\ \vec{x'}_q &= \vec{x}_q \vec{m}_q^{-1} \cdot \delta \cdot \epsilon \end{aligned} \tag{8.5}$$

La Figura 8.17 muestra cómo las restricciones locales son satisfechas consecutivamente en un proceso iterativo. Una restricción de distancia c se define entre dos partículas p y q (Figura 8.17 a). Si un enlace entre dos partículas es estresado, las partículas son proyectadas a sus posiciones ideales (Figura 8.17 b, c). Las partículas más pesadas sufren correcciones menores (Figura 8.17 d). Este proceso se repite para cada restricción iterativamente, un número fijo de iteraciones.

Así pues, si dos partículas se encuentran a menor o mayor distancia que

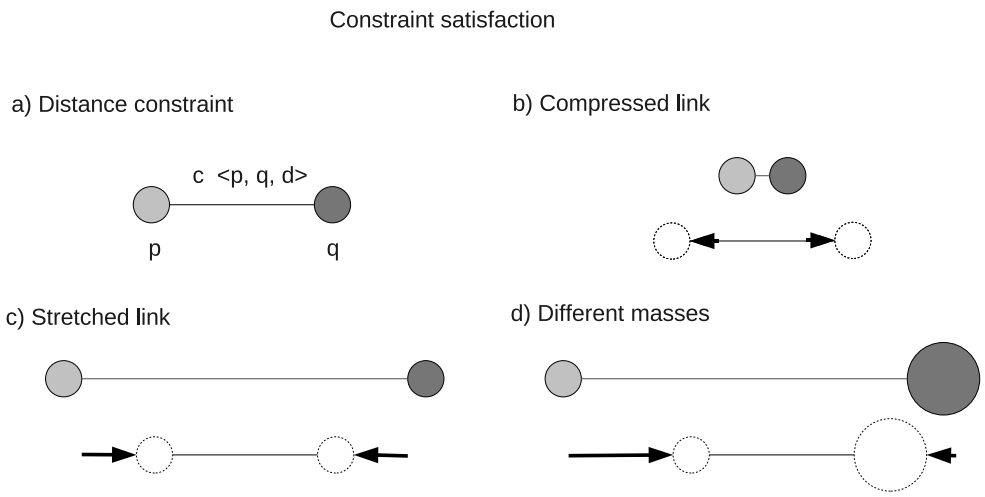


Figure 8.17: Satisfacción de restricciones.

la dictada por el enlace que las une, son proyectadas proporcionalmente según sus masas (ver Figura 8.17. Esta proyección podrá producir la violación de otras restricciones, que serán resueltas en la siguiente iteración. A mayor número de iteraciones, más elásticas se vuelven las estructuras.

La Figura 8.18 muestra el impacto de las iteraciones en la elasticidad en una malla de partículas que representa un trozo de tela, definida en un entorno donde se considera la fuerza de la gravedad. Dos extremos de la tela están clavados al fondo (ambas partículas tienen masa infinita, por lo que no pueden moverse), pero el resto evolucionan siguiendo el esquema de integración verlet, convergiendo al estado correcto tras varias iteraciones. Un mayor número de iteraciones en el proceso de relajación produce cuerpos más rígidos, que por otra parte requieren menos cuadros para alcanzar un estado de equilibrio.

(a) Algunas muestras de los cuadros 0 a 176 del resolvedor de una iteración (malla elástica)

(b) Algunas muestras de los cuadros 0 a 15 del resolvedor en cincuenta iteraciones (malla más rígida)

Figure 8.18: Elasticidad de un cuerpo rígido.

Fuerzas externas

Los enlaces entre partículas son considerados restricciones internas, y definen la forma del cuerpo articulados. Pero cualquiera de estas partículas puede ser afectada por fuerzas relacionadas con el entorno en el que esta inmersa la estructura. En los videojuegos y simulaciones, estas fuerzas pueden ser producidas por una colisión con otro cuerpo rígido, un enlace con un segundo objeto, la gravedad, viento o la interacción del usuario, por nombrar algunos ejemplos.

Algunas de estas fuerzas pueden ser introducidas mediante aceleraciones en el esquema Verlet. La gravedad o el viendo son ejemplos de fuerzas introducidas de esta forma. Pero en muchos otros casos resulta mucho más simple proyectar a las partículas afectadas a la posición deseada (por ejemplo, para evitar que una partícula penetre en un obstáculo), tal y como se hace durante la relajación para resolver las restricciones.

Así pues, los procesos no relacionados con la estructura del cuerpo articulado que afectan a la posición de una o varias partículas son considerados fuerzas externas. Las partículas afectadas son simplemente proyectadas a la

posición designada o reciben una cierta aceleración. Las restricciones internas se encargarán de que el resto del cuerpo articulado reaccione convenientemente.

Es precisamente mediante estas fuerzas externas, definiéndolas como el resultado de diversos procesos visuales, como los cuerpos articulados presentados pueden interactuar con imágenes.

Modelo de ragdoll perceptivo

Cada partícula p estará definida por la siguiente 4-tupla: $p =< \vec{x}_p, \vec{x}_p^*, m_p, \vec{a}_p, V_p >, p \in P$, donde:

- $\vec{x}_p$ es la posición de la partícula en el instante actual.
- $\vec{x}_p^*$ es la posición de la partícula en el instante anterior.
- m_p es la masa de la partícula.
- $\vec{a}_p$ es la aceleración de la partícula.
- V_p representa el proceso visual asignado a la partícula. En este contexto, V_p puede crear una aceleración o una proyección de la partícula según la tarea visual a realizar.

El presente trabajo añade al sistema propuesto por Jakobsen la capacidad de percibir y analizar imágenes digitales, creando fuerzas y proyecciones que afectan a las partículas en base a un proceso visual subyacente. De esta forma, cada partícula puede realizar actividades como la detección, seguimiento o reconocimiento de objetos independientemente, mientras la estructura a la que

pertenece la obliga a permanecer en una misma posición relativa al resto de partículas.

Esta propuesta se aplica al seguimiento de objetos en secuencias de video, logrando estructuras elásticas y deformables capaces de seguir objetos mientras tratan de mantener su forma original. La Figura 8.19 muestra cómo partículas libres terminan por perderse tras dos rotaciones fuera de plano del objeto perseguido, mientras que en la Figura 8.20 las mismas partículas, formando parte de un cuerpo rígido elástico, consiguen recuperar su forma original tras sufrir fuertes deformaciones.

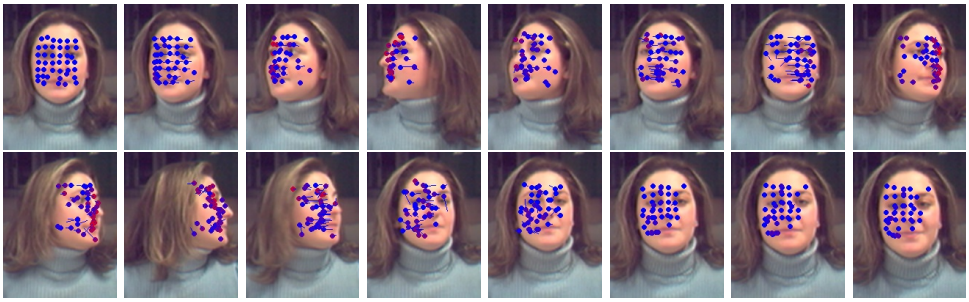


Figure 8.19: La mayor parte de las partículas libres son capaces de seguir su objetivo adecuadamente gracias al banco de vistas, pero muchas de ellas pierden sus objetivos por culpa de la deriva del patrón utilizado.

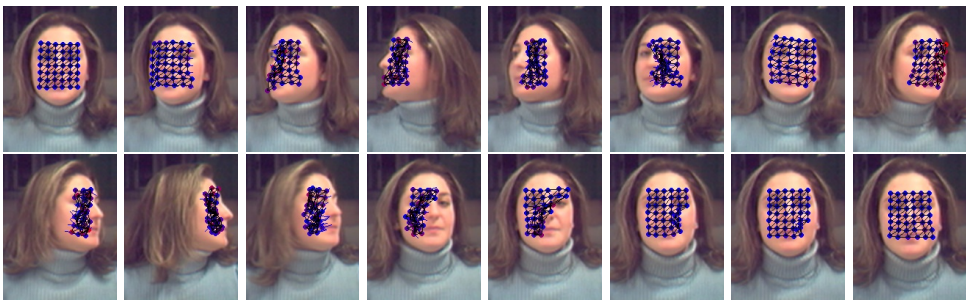


Figure 8.20: La solución de seguimiento basada en Ragdolls consigue recuperar su forma original tras dos rotaciones fuera de plano, superando con éxito ocultaciones parciales.

Búsqueda de patrones

Si bien cada partícula puede adoptar cualquier sistema de seguimiento disponible en la literatura (incluso distintos sistemas dentro del mismo enjambre), se optó por dotarlas de un mecanismo de búsqueda de patrones con actualización basada en contexto capaz de adaptarse a los cambios de apariencia del objeto seguido.

Cada partícula comparará su descriptor del patrón a seguir con cada posición de una ventana de búsqueda definida a su alrededor. La función de distancia utilizada (en este caso Suma de Diferencias Absolutas) definirá una serie de óptimos locales, que configuran el contexto de la partícula. Se tomará el óptimo global como la posición a la que la partícula debe moverse para continuar el seguimiento, mientras que el primer óptimo local será usado en la política de actualizado del patrón.

La Figura 8.21 muestra una superficie creada mediante la convolución de un patrón en una ventana de búsqueda. Los valores de la superficie corresponden a la medida de similitud entre el patrón y la imagen en cada punto de la ventana. En la figura se observa un mínimo global en m_1, que corresponde al punto que mejor casa con el patrón, y dos mínimos locales m_2 y m_3, que miden la similitud del contexto (objetos cercanos similares al buscado). La distancia d entre m_1 y m_2 se utiliza en la política de actualización.

Cada partícula puede guardar parches a distintas escalas, permitiendo que una estructura realice el seguimiento de distintas partes de un objeto a diferentes escalas. Para ello, todos los patrones de búsqueda y su correspondiente ventana se definen a un tamaño fijo. Cualquiera que sea el tamaño del parche utilizado

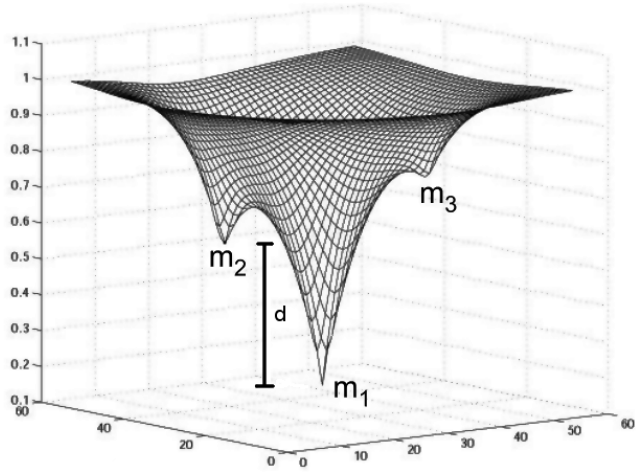


Figure 8.21: Mínimos en una ventana de búsqueda.

por una partícula durante el seguimiento, es redimensionado a este tamaño predeterminado, y lo mismo se hace con su región de búsqueda (manteniendo siempre la misma proporción). Esto permite que las búsquedas a cualquier escala se resuelvan con el mismo coste computacional sin importar el tamaño del parche original. El proceso se muestra en la Figura 8.22.

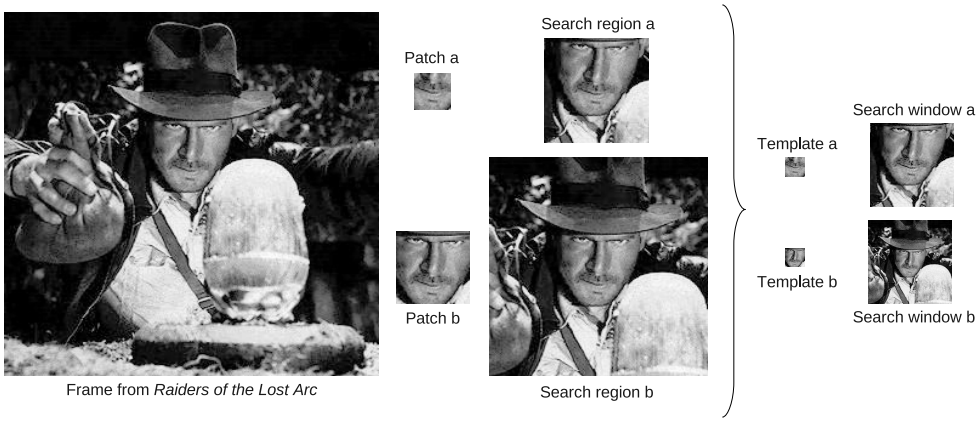


Figure 8.22: Seguimiento a diferentes escalas en tiempo constante.

Actualización del patrón de búsqueda

Esta actualización racional del patrón de búsqueda (y del banco de vistas) consigue que cada partícula guarde un conjunto de vistas representativas y no redundantes del objeto seguido. Para ello, es el contexto de la partícula el que dicta cuándo ésta debe actualizar su descriptor. Esta condición viene dada por los valores de los mínimos locales, vistos en la Figura 5.8. Concretamente, se actualizará en dos ocasiones:

1. El mínimo global m_1 está por encima del umbral de actualización τ, lo que significa que la apariencia del objeto ha variado significativamente, o
2. el segundo mínimo m_2 está por debajo del umbral de actualización τ, lo que significa que hay un objeto en la ventana de búsqueda que es demasiado similar al que estamos siguiendo.

Al realizar la actualización, se calcula el nuevo umbral como:

$$\tau = (m_2 - m_1) \cdot 0.5 \tag{8.6}$$

La actualización por contexto crea distintas frecuencias de actualización para cada partícula. En la Figura 8.23 se muestran todas las actualizaciones del patrón de búsqueda sufridas por seis partículas distintas. Se detallan seis cuadros de una secuencia de cuarenta y dos de *Cazafantasmas*, de Ivan Reitman. Bajo éstos, cada línea muestra las vistas capturadas por seis partículas diferentes. Los círculos verdes muestran la vista utilizada al terminar la secuencia, mientras que los rojos muestran la vista que peor casó con la ventana de búsqueda en ese momento.

Nótese que la mejor vista no es siempre la última. Cada partícula actualiza su patrón según su contexto.

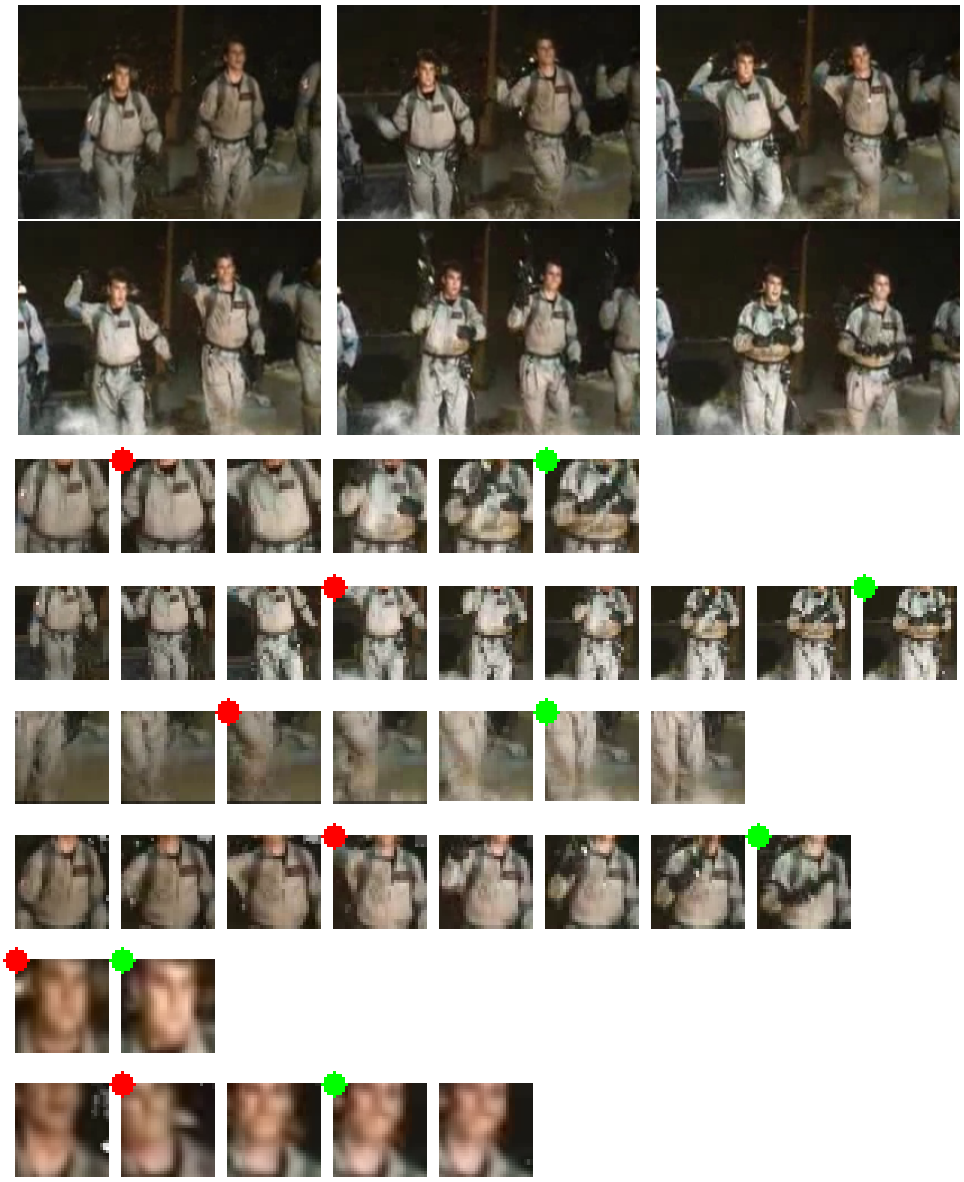


Figure 8.23: Actualización basada en contexto

Banco de vistas

Cada partícula guarda un conjunto de vistas de su objetivo, lo que representa una memoria visual a corto plazo donde se guarda la apariencia reciente del objeto perseguido. En cada iteración cada partícula realiza la búsqueda de cada una de las vistas contenidas en su memoria visual en la ventana de búsqueda, y tomará como vista actual la que obtenga el mejor valor devuelto por la función de comparación.

Debido a la capacidad finita de cómputo y almacenamiento, el tamaño del banco de vistas debe ser limitado (en el presente trabajo el límite son tres vistas por partícula). Es por lo tanto necesario decidir qué vista debe ser sustituida en caso de que el banco de vistas esté lleno y haya que actualizar. Para ello se utiliza una medida de la utilidad de cada vista, a partir de su persistencia (durante cuántos cuadros ha sido utilizada) y su obsolescencia (cuándo fue la última vez que fue utilizada). Asi, la utilidad queda definida como:

$$utilidad = persistencia/(1.0 + obsolescencia) \tag{8.7}$$

La vista de menor utilidad es sustituida por la nueva vista, manteniendo la memoria visual actualizada con las vistas recientes más utilizadas (de ahí que sea considerada una memoria a corto plazo). Con el fin de evitar que las partículas olviden completamente la apariencia original del objeto seguido debido a sucesivas actualizaciones del patrón, se considera el uso de una vista canónica. Esta vista no se actualiza nunca, representando una primitiva memoria a largo plazo.

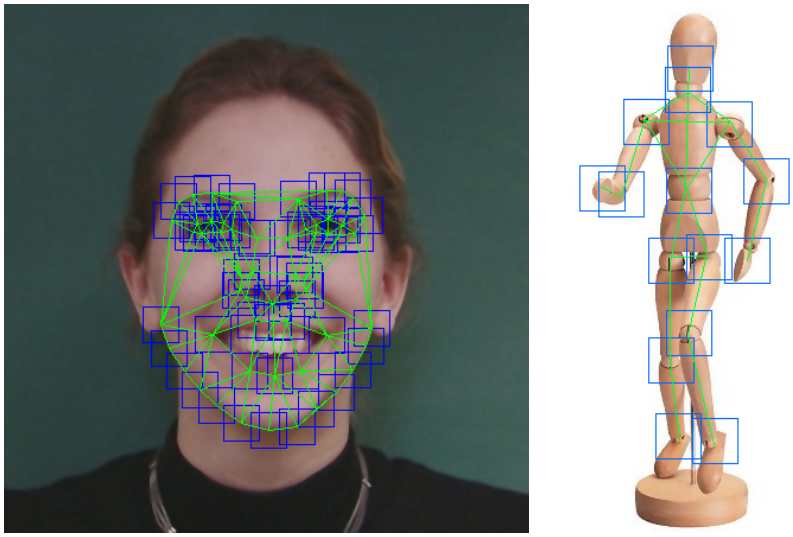
Si el seguidor es creado sobre una perspectiva habitual del objeto seguido

(por ejemplo, una cara frontal), es aceptable suponer que las vistas contendrán un patrón fiable de la apariencia de su parte correspondiente del objeto. Estas vistas canónicas son susceptibles de volver a aparecer en algún momento de la secuencia, sin importar las ocultaciones que puedan sufrir en un momento determinado.

Las vistas canónicas dejan de ser útiles cuando cambia la apariencia del objeto de forma permanente o durante un tiempo excesivamente largo. Si la alteración se debe a cambios de iluminación, el uso de descriptores invariantes puede aliviar el problema. Sin embargo, estas vistas dejan de ser útiles cuando un objeto muestra una perspectiva completamente nueva de si mismo (por ejemplo, una cabeza gira para mostrar la nuca en lugar de la cara o un coche que previamente veíamos de lado gira y se enfrenta a la cámara). Para abordar estos casos parece necesario el uso de conocimiento previo sobre el objeto, ya que el seguidor necesita conocer la nueva apariencia para reconocer en ésta al mismo objeto que venía siguiendo. Puesto que en este trabajo se considera exclusivamente el seguimiento precategórico, estos casos no son considerados.

Forma de las estructuras

La estructura que crean las partículas tendrá una forma u otra dependiendo de los enlaces entre partículas. Los cuerpos articulados pueden ser construidos con la forma de un objeto conocido, adoptando por ejemplo una estructura facial creada a partir de puntos anotados manualmente o la forma de un esqueleto humano simplificado (ver Figura 8.24). Esto permite incluir información a priori del objeto a estudiar, aunque debido al carácter precategórico del presente trabajo esta opción no ha sido considerada.



(a) (b)

Figure 8.24: Estructuras definidas por el usuario

.

El trabajo propuesto considera diversas estructuras genéricas. La más simple consiste en una rejilla en cuyos vértices se colocan las partículas y donde los enlaces pueden ser de cada partícula hacia los cuatro (ver Figura 8.25 a) u ocho vecinos directos. Otra configuración consiste en colocar las partículas en las localizaciones definidas por un detector de puntos salientes (con o sin escala) (ver Figuras 8.25 b y 8.25 c) y crear los enlaces entre ellas a partir de la triangularización de Delaunay que definen las partículas. Esta segunda configuración tiene la ventaja de que las partículas se crean en puntos que son, por definición, interesantes para un sistema de seguimiento. Además, son estructuras más robustas que no pueden plegarse sobre sí mismas.

Al utilizar un detector de puntos saliente en escala como DoG o SURF se obtiene no solo una buena localización espacial de cada partícula sino también su escala óptima en ese punto. Los ragdolls creados utilizando este tipo de detectores

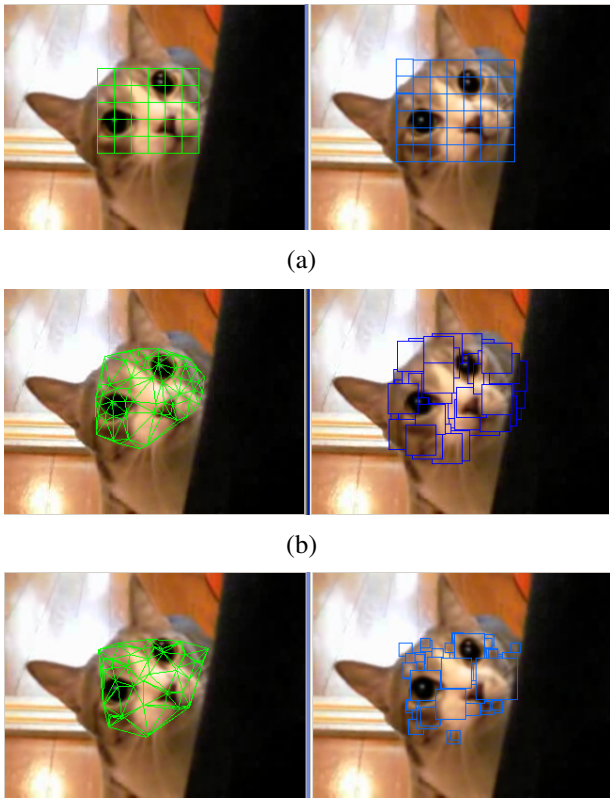


Figure 8.25: Tres configuraciones genéricas de la estructura: a) Regilla simple de parches no superpuestos de 19x19 píxeles. b)Triangulación de Delaunay de puntos KLT, usando parches de 19x19 píxeles. c) Triangulación de Delaunay de puntos salientes SURF con escala

están compuestos por partículas de diferentes tamaños (esto es, partículas cuyas vistas tienen un tamaño diferente del de otras partículas).

Gracias al sistema mostrado en la Figura 8.22 el seguimiento se realizará a distintas escalas al mismo tiempo. Esto facilita el seguimiento de objetos de movimiento rápido y robustece la calidad del seguimiento en términos generales, al permitir al seguidor seguir detalles de distinto tamaño. Por otra parte, las estructuras creadas a partir de la triangulación de puntos salientes en escala suelen ser mucho más rígidas, limitando las posibles deformaciones.

Calidad del seguimiento y pesos de las partículas

Se plantea la necesidad de evaluar la calidad del seguimiento. Esta medida de calidad define la masa de cada partícula, lo que a su vez influye en cómo ésta responde ante las fuerzas creadas por las restricciones internas. Las partículas de mayor masa tienen mayor capacidad de arrastre de la estructura (su voto es más importante). Se evalúan distintas medidas:

1. Calidad del seguimiento: las partículas son asignadas un peso obtenido del óptimo global en la ventana de búsqueda. Así, las partículas cuya vista actual encuentra buena correspondencia son asignadas pesos mayores. Sin embargo, que la función de similitud devuelva un buen valor no asegura la calidad del seguimiento, ya que los bancos de vistas pueden haberse corrompido en algún momento, incorporando vistas extrañas. Tan sólo el seguimiento de vistas no actualizadas puede considerarse fiable.

2. Utilidad: la fiabilidad de una vista viene dada por su utilidad. Una vista con alta persistencia y baja obsolescencia es idealmente una vista útil. Sin embargo, una partícula perdida también puede acumular una alta utilidad si se persigue una misma vista incorrecta durante mucho tiempo.

3. Combinación de la calidad del seguimiento y la utilidad: idealmente, el seguimiento de calidad debería corresponder a un buen valor de la función de similitud unido a una alta utilidad. Sin embargo, dado que ninguna de las dos medidas es de por sí completamente fiable, tampoco su combinación lo es.

4. Análisis de la peor vista: la vista con el peor valor de similitud es aquella que más difiere de la posición localizada por la mejor vista, y por lo tanto la más diferente de todo el banco de vistas. Su medida de similitud representa cómo de heterogéneo es el banco de vistas: si es demasiado alta, significa que el banco contiene patrones muy diferentes entre sí, lo que puede ser una pista que indique que en algún momento se produjo una actualización incorrecta. El peso de las partículas podría ser asignado según este peor valor, dándole mayor importancia a aquellas partículas cuyos bancos de vistas sean más homogéneos.

   Sin embargo, la diferencia entre la peor y la mejor vista podría venir dada no por un error en la actualización, sino porque realmente la apariencia del objeto haya sufrido un cambio drástico y haya sido necesario reflejarlo actualizando.

5. Estrés de los enlaces: el nivel de estrés en los enlaces salientes de una partícula aumenta si ésta se aleja excesivamente de su posición relativa en el grupo. Un alto nivel de estrés puede indicar que una partícula está siguiendo un objetivo equivocado, pero también puede significar que es la única que ha descubierto un cambio significativo en el objeto. En este segundo caso, penalizarla puede conllevar la pérdida del objeto seguido.

6. Pesos homogéneos: dado que ninguna de las medidas anteriores parece ser realmente fiable, tal vez las partículas deberían tener todas un peso unitario y permitir que sea la dinámica y el reajuste de la estructura por restricciones internas el mecanismo corrector de las partículas mal colocadas.

Elasticidad del Ragdoll

El número de iteraciones en el proceso de solución de restricciones limita la elasticidad de toda la estructura e influencia el seguimiento.

Las estructuras elásticas permiten que las partículas sigan sus objetivos con un mayor grado de independencia. Las partículas pueden explorar su espacio de búsqueda y moverse sin demasiados límites, deformando al ragdoll como se muestra en la Figura 8.26 a.

Un mayor número de iteraciones produce estructuras más rígidas. Las partículas todavía pueden explorar su espacio de búsqueda, pero no se les permite alejarse demasiado de su posición relativa esperada dentro de la estructura. Las partículas mal colocadas son rápidamente corregidas como se muestra en la Figura 8.26 b, limitando las deformaciones.

En ambos casos, la forma original es recuperada después de ser deformada gracias a aquellas partículas que permanecen sobre partes visibles del objeto seguido. Su valor de utilidad en esos casos debería ser mayor que el de las partículas que perdieron su objetivo, las cuales serán arrastradas hacia la posición adecuada.

Figure 8.26: La rigidez limita la libertad de movimientos. Con tan solo dos iteraciones en el paso de resolución de restricciones, las estructuras resultantes son muy elásticas (a). Con diez iteraciones, la estructura es mucho más rígida (b). Aun así, las estructuras en rejilla como las usadas en el ejemplo pueden plegarse sobre sí mismas, no importa su rigidez. En ambos casos, la estructura es capaz de recuperar su forma después de fuertes deformaciones si suficientes partículas permanecen sobre sus objetivos adecuadamente.

La recuperación de la forma puede verse afectada por las actualizaciones de los patrones. Si las partículas actualizan sus vistas, pueden incorporar partes de objetos extraños y acumular utilidad (por ejemplo, guardando vistas del objeto que oculta momentáneamente al objeto seguido). A no ser que el cuerpo articulado contenga suficientes restricciones, estas partículas pueden no ser devueltas a su lugar dentro de la estructura.

8.5 Conclusiones

La presente tesis ha abordado el tema de la Inteligencia de Enjambres en la Visión por Ordenador, aplicándola al seguimiento de objetos. La parte principal del trabajo se ha centrado en la propuesta y desarrollo de dos novedosos sistemas basados en enjambres que son capaces de seguir objetos visuales en secuencias de video en una gran variedad de escenarios reales.

El primer modelo considera una estructura pura de enjambres cuyos miembros se guían por factores psico-sociales en una metáfora depredador-presa, mientras que el segundo modelo propone un enjambre cuyos individuos están sujetos a restricciones estructurales que conforman una estructura elástica. La tesis incluye resultados cuantitativos obtenidos al aplicar ambos métodos en cinco secuencias representativas (aunque durante el trabajo se utilizaron muchas otras). El trabajo presentado en esta tesis puede ser resumido en las siguientes conclusiones:

1. La Inteligencia de Enjambres y las soluciones basadas en poblaciones están inspiradas en modelos biológicos. El comportamiento de los insectos sociales ha probado ser especialmente adecuado para el control distribuido y para la resolución de problemas de optimización en un amplio rango de aplicaciones. Aunque algunas aproximaciones consideran modelos teóricos detallados del comportamiento de la criatura, grupo o actividades que simulan (como el depósito y evaporación de feromonas en colonias de hormigas), muchas otras se apoyan en potentes metáforas (PSO y algoritmos genéticos) para lograr sistemas exitosos.

2. Abordar un problema a través de metáforas generalmente permite desarrollar el pensamiento lateral, lo que suele conllevar el descubrimiento de soluciones alternativas que de otra forma no habrían sido consideradas. Mediante el uso de metáforas, el problema se replantea como una situación análoga. La atención se centra entonces en encontrar soluciones al meta-problema. Finalmente, las ideas generadas son aplicadas a la tarea original, si fuese posible. Este procedimiento es ampliamente aceptado en el arte y el diseño, donde la creatividad es crucial.

3. La aplicación de la Inteligencia de Enjambres a tareas conocidas de la Visión por Ordenador como el seguimiento precategórico de objetos crea nuevas y estimulantes aproximaciones a los problemas visuales. Según los miembros del enjambre se centran en sus trabajos, cuidadosamente diseñados para ser relativamente simples e independientes, las soluciones a problemas complejos emergen gracias a la sinergia y/o estigmergia. El seguimiento, en el trabajo presente, emerge de las interacciones sociales y físicas entre individuos.

4. La mayor parte de las soluciones al seguimiento encontradas en la literatura utilizan un único óptimo para localizar la nueva posición del objeto seguido, desestimando otras posiciones alternativas. El seguimiento en estos casos continúa en cada iteración desde una única posición anterior. Las soluciones basadas en Inteligencia de Enjambres pueden aprovecharse de la naturaleza distribuida de sus miembros para seguir múltiples partes de un objeto o múltiples representaciones del mismo simultáneamente.

5. Todo sistema de seguimiento precategórico, incluso siguiendo una

política de actualización de patrones prudente como la basada en contexto, es propenso a perder el objeto seguido en numerosas circunstancias. Las soluciones distribuidas como las propuestas alivian en parte el problema principal, ya que los mismos individuos que componen las poblaciones se autoregulan mediante diversos mecanismos (comportamientos direccionales, restricciones internas, mapas de feromonas...).

8.5.1 Trabajo Futuro

El trabajo presentado ha generado varias líneas de investigación que merecen atención:

1. Exploración de rutas alternativas. Los óptimos locales descartados en las ventanas de búsqueda podrían servir para crear nuevas partículas, eliminando otras según su edad, salud o densidad de población local. Estos óptimos locales representan objetos en la vecindad de cada partícula cuya apariencia es muy semejante a la del objeto seguido. Al crear nuevas partículas en dichas posiciones, la colonia sería capaz de registrar la evolución de estas rutas alternativas, lo que podría permitirle superar situaciones complicadas.

2. Extracción automática de características discriminantes. Cada partícula podría, en cada instante, decidir qué características maximizan la diferencia entre el objeto seguido y el entorno circundante. De esta manera podrían adaptarse a los cambios de apariencia según el contexto y optimizar la función de similitud.

3. Considerar nuevos tipos de restricciones en el modelo ragdoll. Actualmente solo se utilizan restricciones de distancia, pero podrían considerarse otros tipos como restricciones angulares o restricciones maleables.

4. Enjambres compuestos. Actualmente, docenas de partículas seguidoras independientes crean una única entidad seguidora. Podrían docenas de estas entidades agruparse para crear un superorganismo? Semejante estructura

podría estar compuesta de entidades heterogéneas con distintos objetivos, procedimientos y reglas. Por ejemplo, unas entidades podrían centrarse en la detección de características u objetos mientras que otras se especializan en su seguimiento. Las dos soluciones propuestas permiten cuatro tipos de enjambres compuestos: enjambres de enjambres, enjambres de ragdolls, ragdolls de enjambres y ragdolls de ragdolls.

5. Explorar nuevas aplicaciones de las estructuras ragdoll propuestas. La aplicación de estas estructuras al análisis de gestos y poses (del cuerpo humano, de caras, de manos...) es directa. Estructuras predefinidas con la forma del objeto a analizar (por ejemplo, un grafo similar al esqueleto humano), combinadas con conocimiento del objeto adquirido mediante entrenamiento, tanto de su dinámica como de su apariencia, parece una línea de trabajo prometedora. Estos sistemas podrían ser utilizados para obtener detectores, seguidores y reconocedores de clases específicas.

Bibliography

Abraham, A., Das, S., and Roy, S. (2007). Swarm intelligence algorithms for data clustering. *Soft Computing for Knowledge Discovery and Data Mining*.

Adam, A., Rivlin, E., and Shimshoni, I. (2006). Robust fragments-based tracking using the integral histogram. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 798–805, Washington, DC, USA. IEEE Computer Society.

Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., and Ogden, J. M. (1984). Pyramid methods in image processing. *RCA Engineer*, 29(6):33–41.

Allen, J. G., Xu, R. Y. D., and Jin, J. S. (2004). Object tracking using camshift algorithm and multiple quantized feature spaces. In *VIP '05: Proceedings of the Pan-Sydney area workshop on Visual information processing*, pages 3–7. Australian Computer Society, Inc.

Antón-Canalís, L., Hernández-Tejera, M., and Sánchez-Nielsen, E. (2006a). Addcanny: Edge detector for video processing. In *ACIVS06*, pages 501–512.

Antón-Canalís, L., Hernández-Tejera, M., and Sánchez-Nielsen, E. (2006b). Particle swarms as video sequence inhabitants for object tracking in computer vision. *Intelligent Systems Design and Applications, 2006. ISDA '06. Sixth International Conference on*, 2:604–609.

Antón-Canalís, L., Hernández-Tejera, M., and Sánchez-Nielsen, E. (2006c). Swarmtrack: A particle swarm approach to visual tracking. *VISAPP 2006 Proceedings*.

Antón-Canalís, L., Hernández-Tejera, M., and Sánchez-Nielsen, E. (2007). Analysis of relevant maxima in distance transform. an application to fast coarse image segmentation. In *IbPRIA '07: Proceedings of the 3rd Iberian conference on Pattern Recognition and Image Analysis, Part I*, pages 97–104, Berlin, Heidelberg. Springer-Verlag.

Arikan, O., Forsyth, D. A., and O'Brien, J. F. (2003). Motion synthesis from anotations. In *Proceedings of ACM SIGGRAPH 2003*, pages 402–408.

Arnaud, Doucet, and Johansen (2008). A tutorial on particle filtering and smoothing: Fifteen years later. Technical report.

Bagon, S., Boiman, O., and Irani, M. (2008). What is a good image segment? a unified approach to segment extraction. In Forsyth, D., Torr, P., and Zisserman, A., editors, *Computer Vision – ECCV 2008*, volume 5305 of *LNCS*, pages 30–44. Springer.

Bay, H., Tuytelaars, T., and Gool., V. (2006). Surf: Speeded up robust features. In *9th European Conference on Computer Vision*, Graz Austria.

Bazazi, S., Buhl, J., Hale, J. J., Anstey, M. L., Sword, G. A., Simpson, S. J., and Couzin, I. D. (2008). Collective motion and cannibalism in locust migratory bands. *Current Biology*.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522.

Beni, G., W. J. (1989). Swarm intelligence in cellular robotic systems. In *Proceed. NATO Advanced Workshop on Robots and Biological Systems*.

Birchfield, S. and Rangarajan, S. (2005). Spatiograms versus histograms for region-based tracking. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2:1158–1163 vol. 2.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence. From Natural to Artificial Systems*. Oxford University Press, Inc., New York, Oxford, USA.

Borenstein, E. and Ullman, S. (2002). Class-specific, top-down segmentation. *Computer Vision - ECCV 2002 : 7th European Conference on Computer Vision, Copenhagen, Denmark, May 28-31, 2002. Proceedings, Part II*, pages 639–641.

Borenstein, E. and Ullman, S. (2004). Learning to segment. In *ECCV (3)*, pages 315–328.

Borgefors, G. (1988). Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865.

Bourjot, C., Chevrier, V., and Thomas, V. (2003). A new swarm mechanism based on social spiders colonies: From web weaving to region detection. *Web Intelligence and Agent Systems*, 1(1):47–64.

Bradski, G. R. (1998). Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*.

Buchner, L. (1881). *La vie psychique des bêtes*. Paris: Reinwald.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G., and Bonabeau, E. (2001). *Selforganization in biological systems*. Princeton University Press.

Canny, J. (1986). A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698.

Capell, S., Green, S., Curless, B., Duchamp, T., and Popović, Z. (2002). Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph.*, 21(3):586–593.

Casakin, H. P. (2007). Metaphors in design problem solving: Implications for creativity. *International Journal of Design*, 1(2).

Chen, D. and Yang, J. (2005). Online learning of region confidences for object tracking. In *ICCCN '05: Proceedings of the 14th International Conference on Computer Communications and Networks*, pages 1–8, Washington, DC, USA. IEEE Computer Society.

Chen, J., Pappas, T. N., Mojsilovic, A., and Rogowitz, B. E. (2003). Image segmentation by spatially adaptive color and texture features. *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, 1:I–1005–8 vol.1.

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):790–799.

Collins, R. T. (2003). Mean-shift blob tracking through scale space. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2:II–234–40 vol.2.

Collins, R. T. and Liu, Y. (2003). On-line selection of discriminative tracking features. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 346, Washington, DC, USA. IEEE Computer Society.

Comaniciu, D. and Meer, P. (2002). Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619.

Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564–577.

Consoli, S., Moreno-PÃ©rez, J. A., Darby-Dowman, K., and Mladenovic, N. (2007). Discrete particle swarm optimization for the minimum labelling steiner tree problem. In Krasnogor, N., Nicosia, G., Pavone, M., and Pelta, D. A., editors, *NICSO*, volume 129 of *Studies in Computational Intelligence*, pages 313–322. Springer.

Cootes, T. F., Edwards, G. J., and Taylor, C. J. (1998). Active appearance models. *Lecture Notes in Computer Science*, 1407:484–??

Cootes, T. F., Taylor, C. J., Cooper, D. H., and Graham, J. (1995). Active shape models—their training and application. *Comput. Vis. Image Underst.*, 61(1):38–59.

Davies, R. H., Cootes, T. F., and Taylor, C. J. (2001). A minimum description length approach to statistical shape modelling. In *IPMI '01: Proceedings of the 17th International Conference on Information Processing in Medical Imaging*, pages 50–63, London, UK. Springer-Verlag.

Dehuri, S. and Cho, S.-B. (2009). Multi-criterion pareto based particle swarm optimized polynomial neural network for classification: A review and state-of-the-art. *Computer Science Review*, 3(1):19 – 40.

Delaunay, B. (1934). Sur la sphÃ¨re vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:793–800.

Denebourg, J., Pasteels, J., and Verhaeghe, J. (1983). Probabilistic behaviour in ants: a strategy of errors. *Journal of Theoretical Biology*.

Deriche, R. (1987). Using canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1(2):167–187.

Deutscher, J., Blake, A., and Reid, I. (2000). Articulated body motion capture by annealed particle filtering. *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, 2:126–133 vol.2.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms, PhD thesis*. Politecnico di Milano.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. MIT Press.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern classification (2nd edition)*. Wiley.

Edelman, S. (1999). *Representation and recognition in vision*. MIT Press, Cambridge, MA, USA.

Engelbrecht, A. P. (2005). *Fundamentals of Computational Swarm Intelligence*. Wiley.

Fieguth, P. and Terzopoulos, D. (1997). Color-based tracking of heads and other mobile objects at video frame rates. In *in Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 21–27.

Fogel, I. and Sagi, D. (1989). Gabor filters as texture discriminator. *Biological Cybernetics*, 61:102–113.

Forel, A. (1921). *Le monde social des fourmis du globe comparé à celui de lh́omme*. Genève: Librairie Kundig.

Forsyth, D. and Ponce, J. (2003). *Computer Vision: A Modern Approach*. NJ: Prentice-Hall.

Fourcassi, V. and Deneubourg, J. (1994). The dynamics of collective exploration and trail-formation in monomorium pharaonis: experiments and model. *Physiological Entomology*.

Freddolino, P. L., Arkhipov, A. S., Larson, S. B., Mcpherson, A., and Schulten, K. (2006). Molecular dynamics simulations of the complete satellite tobacco mosaic virus. *Structure*, 14(3):437–449.

Fritsch, J., Lang, S., Kleinehagenbrock, A., Fink, G., and Sagerer, G. (2002). Improving adaptive skin color segmentation by incorporating results from face detection. *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on*, pages 337–343.

Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32–40.

Goldman, D. B., Gonterman, C., Curless, B., Salesin, D., and Seitz, S. M. (2008). Video object annotation, navigation, and composition. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 3–12, New York, NY, USA. ACM.

Gordon, D. M. (2007). Control without hierarchy. *Nature*.

Grasp, C. T. and Taylor, C. J. (2000). Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Computer Vision and Image Understanding*, 80:677–684.

Grass, P. P. (1959). La reconstruction du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la thorie de la stigmergie : essai dinterprtation du comportement des termites constructeurs. *Insectes Sociaux*.

Greenspan, H., Belongie, S., Goodman, R., and Perona, P. (1994). Rotation invariant texture recognition using a steerable pyramid. In *in Proc. Int. Conf. on Pattern Recognition*, pages 162–167.

Guerra, C. (2002). *Contribuciones al seguimiento visual precategorico*. PhD thesis, Universidad de Las Palmas de Gran Canaria.

Guerra, C., Hernandez, M., Domínguez, A., and Hernandez, D. (2005). A new approach to the template update problem. *Lecture Notes in Computer Science LNCS*, (3522):217–224.

Hagedoorn, M. (2000). *Pattern Matching using similarity measures*. PhD thesis, Utrecht University.

Hager, G., Dewan, M., and Stewart, C. (2004). Multiple kernel tracking with ssd. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, 1:I–790–I–797 Vol.1.

Handl, J., Knowles, J., and Dorigo, M. (2003). On the performance of ant-based clustering. In *Design and application of hybrid intelligent systems*, pages 204–213. IOS Press, Amsterdam, The Netherlands, The Netherlands.

Harris, C. and Stephens, M. (1988). A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151.

Heikkila, M. (2006). A texture-based method for modeling the background and detecting moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):657–662.

Heinlein, R. A. (1987). *Starship Troopers*. Ace.

Heppner, F. and Grenander, U. (1990). A stochastic nonlinear model for coordinated bird flocks. *The Ubiquity of Chaos*.

Hinrichs, N. S. and Pande, V. S. (2007). Calculation of the distribution of eigenvalues and eigenvectors in markovian state models for molecular dynamics. *The Journal of Chemical Physics*, 126(24).

Holland, J. H. (1992). *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA.

Hölldobler, B. (1990). *The ants*. Cambridge: Harvard University Press.

Hu, X., , Y. S., and Eberhart, R. (2004). Recent advances in particle swarm. *Congress on Evolutionary Computation. CEC2004*, 1:90–97.

Hu, X. and Eberhart, R. C. (2002). Adaptive particle swarm optimization: Detection and response to dynamic systems. *Proceedings of the IEEE Congress on Evolutionary Computation. CEC 2002*, 2(12-17):1666–1670.

Huttenlocher, D., Klanderman, G., and Rucklidge, W. (1993). Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863.

Isard, M. and A., B. (1998). Condensation-conditional density propagation for visual tracking. *International Journal of Computer Vision*, (29(1)):5–28.

Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(11):1254–1259.

Jakobsen, T. (2001). Advanced character physics. *TechReport, IO Interactive*.

Jha, S., Casey-Ford, R. G., Pedersen, J. S., Platt, T. G., Cervo, R., Queller, D. C., and Strassmann, J. E. (2006). The queen is not a pacemaker in the small-colony wasps polistes instabilis and p. dominulus. *Animal Behaviour*, 71(5):1197–1203.

Jiang, M., Mastorakis, N., Yuan, D., and Lagunas, M. (2007). Multi-threshold image segmentation with improved artificial fish swarm algorithm. In *European Computing Conference (ECC).*

Jones, J. and Saeed, M. (2007). Image enhancement - an emergent pattern formation approach via decentralised multi-agent systems. *Multiagent Grid Syst.*, 3(1):105–140.

Jost, C., Verret, J., Casellas, E., Gautrais, J., Challet, M., Lluc, J., Blanco, S., Clifton, M. J., and ., G. T. (2006). The interplay between a self-organized process and an environmental template: corpse clustering under the influence of air currents in ants. *Journal of the Royal Society Interface*, 4:107–116.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45.

Kambhamettu, C., Goldgof, D. B., Terzopoulos, D., and Huang, T. S. (1994). Nonrigid motion analysis. *Handbook of Pattern Recognition and Image Processing: Computer vision*, 2:405–430.

Karsai, I. and Theraulaz, G. (1995). Nest building in a social wasp: postures and constraints. *Sociobiology*.

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, V1(4):321–331.

Kennedy and James (2007). Review of engelbrecht's fundamentals of computational swarm intelligence. *Genetic Programming and Evolvable Machines*, 8(1):107–109.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, IV:1492–1948.

Kennedy, J. and Eberhart, R. C. (2001). *Swarm intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Kobayashi, T., Nakagawa, K., Imae, J., and Zhai, G. (2007). Real time object tracking on video image sequence using particle swarm

optimization. *Control, Automation and Systems, 2007. ICCAS '07. International Conference on*, pages 1773–1778.

Kolsch, M. and Turk, M. (2005). Hand tracking with flocks of features. *Video Proc. CVPR IEEE Conference on Computer Vision and Pattern Recognition*.

Koza, J. (1994). *Genetic programming II: Automatic discovery of reusable programs*. Massachusetts Institute of Technology, Cambridge, Massachusetts.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*. The MIT Press.

Koza, J. R. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, Norwell, MA, USA.

Koza, J. R., Andre, D., Bennett, F. H., and Keane, M. A. (1999). *Genetic Programming III: Darwinian Invention & Problem Solving*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Leibe, B., Leonardis, A., and Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1):259–289.

Lettvin, J. Y., Maturana, H. R., Mcculloch, W. S., and Pitts, W. H. (1940). What the frog's eye tells the frog's brain. In *Proceedings of the IRE*.

Lindeberg, T. (1993). Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention. *International Journal of Computer Vision*, 11:283–318.

Lindeberg, T. (1998). Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30:79–116.

Liu, T., Sun, J., Zheng, N.-N., Tang, X., and Shum, H.-Y. (2007). Learning to detect a salient object. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8.

Lowe, D. (1999). Object recognition from local scale-invariant features. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 2:1150–1157 vol.2.

Maskell, S. and Gordon, N. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188.

Masson, L., Dhome, M., and Jurie, F. (2005). Tracking 3d objects using flexible models. In *Proceedings of the 2005 IEEE British Machine Vision Conference*.

Matthews, I., Ishikawa, T., and S., B. (2004). The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (26(6)):810–815.

Meyer, F. and Beucher, S. (1990). Morphological segmentation. *Journal of Visual Communication and Image Representation*.

Mikolajczyk, K. and Schmid, C. (2001). Indexing based on scale invariant interest points. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, pages 525–531.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Van Gool, L. (2005). A comparison of affine region detectors. *Int. J. Comput. Vision*, 65(1-2):43–72.

Miller, P. (2007). Swarm theory. the genius of swams. *National Geographic*.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. The MIT Press.

Moravec, H. (1979). Visual mapping by a robot rover. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, pages 599–601.

Mori, G. (2005). Guiding model search using segmentation. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2:1417–1423 Vol. 2.

Mori, G. and Malik, J. (2006). Recovering 3d human body configurations using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1052–1062.

Mori, G., Ren, X., Efros, A. A., and Malik, J. (2004). Recovering human body configurations: combining segmentation and recognition. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–326–II–333 Vol.2.

Mundhenk, T. N., Everist, J., Landauer, C., Itti, L., and Bellman, K. (2005). Distributed biologically-based real-time tracking in the absence of prior target information. In Casasent, D. P., Hall, E. L., and Roning, J., editors, *Proc. SPIE International Conference on Intelligent Robots and Computer Vision XXIII: Algorithms, Techniques, and Active Vision*, volume 6006, pages 142–153, Bellingham, WA. SPIE Press.

Murray, C., Merrick, D., and Takatsuka, M. (2004). Graph interaction through force-based skeletal animation. In *APVis '04: Proceedings of the 2004 Australasian symposium on Information Visualisation*, pages 81–90, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

Ngo, J. T. and Marks, J. (1993). Spacetime constraints revisited. In *SIGGRAPH*, pages 343–350.

Ofria, C. and Wilke, C. O. (2004). Avida: a software platform for research in computational evolutionary biology. *Artificial Life*, 10(2):191–229.

Ojala, T., Pietikäinen, M., and Mäenpää, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):971–987.

Ouadfel, S. and Batouche, M. (2007). An Efficient Ant Algorithm for Swarm-Based Image Clustering. *Journal of Computer Science*, 3(3):162–167.

Owechko, Y. and Medasani, S. (2005). Cognitive swarms for rapid detection of objects and associations in visual imagery. *Swarm Intelligence Symposium. SIS 2005. Proceedings 2005 IEEE*, (8-10):420–423.

Park, S.-J., Shin, J.-K., and Lee, M. (2002). Biologically inspired saliency map model for bottom-up visual attention. In *BMCV '02: Proceedings of the Second International Workshop on Biologically Motivated Computer Vision*, pages 418–426, London, UK. Springer-Verlag.

Partridge, B. L. and Pitcher, T. J. (1980). The sensory basis of fish schools: relative roles of lateral line and vision. *J. Comp. Physiol.*, (135A):315325.

Perlin, K. (1984). Acm siggraph. In *Course in Advanced Image Synthesis*.

Perlin, K. (1985). An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296.

Perlin, K. (2002). Improving noise. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 681–682, New York, NY, USA. ACM.

Perlin, K. and Hoffert, E. M. (1989). Hypertexture. *SIGGRAPH Comput. Graph.*, 23(3):253–262.

Perona, P. and Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639.

Pietikäinen, M. (2005). Image analysis with local binary patterns. *Image Analysis*, pages 115–118.

Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *J. Artif. Evol. App.*, 8(2):1–10.

Press, W. H., Vetterling, W. T., Teukolsky, S. A., and Flannery, B. P. (2002). *Numerical Recipes in C++: the art of scientific computing*.

Puig, D. and Garcia, M. A. (2006). Automatic texture feature selection for image pixel classification. *Pattern Recognition*, 39(11):1996 – 2009.

Ramos, V. and Almeida, F. (2000). Artificial ant colonies in digital image habitats - a mass behaviour effect study on. In *In Dorigo, M., Middendorf, M., Stuzle, T. (Eds.): From Ant Colonies to Artificial Ants - 2 nd Int. Wkshp on Ant Algorithms*, pages 113–116.

Ramos, V., Fernandes, C., and C.Rosa, A. (2005). Social cognitive maps, swarm collective perception and distributed search on dynamic landscapes. *Brains, Minds and Media, Journal of New Media in Neural and Cognitive Science, NRW*.

Ray, T. S. (1991). An approach to the synthesis of life. In Langton, C. G., Taylor, C., Farmer, D. J., and Rasmussen, S., editors, *Artificial Life II*, pages 371–408, Redwood City, CA. Addison-Wesley.

Reeve and Gamboa (1983). Colony activity integration in primitively eusocial wasps: the role of the queen (polistes fuscatus, hymenoptera: Vespidae). *Behavioral Ecology and Sociobiology*.

Reeve and Gamboa (1987). Queen regulation of worker foraging in paper wasp: a social feedback control system (polistes fuscatus, hymenoptera: Vespidae). *Behaviour*.

Ren, X. and Malik, J. (2003). Learning a classification model for segmentation. *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 10–17 vol.1.

Ren, X. and Malik, J. (2007). Tracking as repeated figure/ground segmentation. In *Proc. IEEE Conf. Comput. Vision and Pattern Recogn.*

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34.

Ross, K. G. and Matthews, R. W. (1991). *The social biology of wasps*, chapter Evolution of nest architecture. Cornell University Press.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education.

Sand, P. and Teller, S. (2006). Particle video: Long-range motion estimation using point trajectories. *Computer Vision and Pattern Recognition*, 02:2195–2202.

Schmidt, J. and Castrillon, M. (2008). Automatic initialization for body tracking - using appearance to learn a model for tracking human upper body motions. *3rd International Conference on Computer Vision Theory and Applications (VISAPP)*.

Scott-Card, O. (1986). *Ender's Game*. Tor Books.

Shapiro, A., Chu, D., Allen, B., and Faloutsos, P. (2007). The dynamic controller toolkit. In *Sandbox '07: Proceedings of the 2007 ACM SIGGRAPH symposium on Video games*, pages 15–20, New York, NY, USA. ACM.

Shechtman, E. and Irani, M. (2007). Matching local self-similarities across images and videos. In *IEEE Conference on Computer Vision and Pattern Recognition 2007 (CVPR'07)*.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905.

Shi, J. and Tomasi, C. (1994). Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600.

Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. *Proc. IEEE Int. Conf. on Evolutionary Computation*, pages 69–73.

Shim, W. M. and Cavanagh, P. (2004). Attention shift induced by apparent motion can cause position compression. *Journal of Vision*, 4(8).

Shotton, J., Blake, A., and Cipolla, R. (2008). Multiscale categorical object recognition using contour fragments. 30(7):1270–1281.

Sims, K. (1994). Evolving 3d morphology and behavior by competition. In *Artificial Life IV Proceedings*.

Smith, T. and Guild, J. (1931). The c.i.e. colorimetric standards and their use. *Transactions of the Optical Society*.

Takala, V. and Pietikainen, M. (2007). Multi-object tracking using color, texture and motion. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–7.

Tao, Q. and Veldhuis, R. (2007). Illumination normalization based on simplified local binary patterns for a face verification system. In *Proc. of the Biometrics Symposium*, pages 1–6.

Tao, W., Jin, H., and Zhang, Y. (2007). Color image segmentation based on mean shift and normalized cuts. *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, 37(5):1382–1389.

Tautz, J. (2008). *The Buzz about Bees*. Springer, Berlin.

Taylor, T. (1997). The cosmos artificial life system. Technical report, Department of Artificial Intelligence, University of Edinburgh.

Theraulaz, G., Bonabeau, E., and Deneubourg, J.-L. (1998). The origin of nest complexity in social insects. *Complex.*, 3(6):15–25.

Theraulaz, G., Bonabeau, E., Nicolis, S. C., Sole, R. V., Fourcassi, V., Blanco, S., Fournier, R., Joly, J., Fernandez, P., Grimal, A., Dalle, P., and Deneubourg, J. L. (2002). Spatial patterns in ant colonies. In *Proceeding of the National Academy of Sciences*.

Thomas, M. and Kambhamettu, C. (2006). An approximation to mean-shift via swarm intelligence. *Tools with Artificial Intelligence, 2006. ICTAI '06. 18th IEEE International Conference on*, pages 583–590.

Thorpe, W. H. (1963). *Learning and instinct in animals*. London:Methuen.

Tola, E., Lepetit, V., and Fua, P. (2008). A fast local descriptor for dense matching. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8.

Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 839, Washington, DC, USA. IEEE Computer Society.

Toyama, K., Krumm, J., Brumitt, B., and Meyers, B. (1999). Wallflower: principles and practice of background maintenance. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 255–261 vol.1.

Tuceryan, M. and Jain, A. K. (1993). *Texture analysis*. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Verlet, L. (1967). Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Phys. Rev.*, 159(1):98.

Viola, P. and Jones, M. (2001). Robust real-time object detection. In *International Journal of Computer Vision*.

von Bertalanffy, L. (1975). *Perspectives on general system theory : scientific-philosophical studies*. New York : G. Braziller.

White, C., Tagliarini, G., and Narayan, S. (2004). An algorithm for swarm-based color image segmentation. *SoutheastCon, 2004. Proceedings. IEEE*, (26-29):84–89.

Wiskott, L., Fellous, J.-M., Krüger, N., and von der Malsburg, C. (1999). Face recognition by elastic bunch graph matching. pages 355–398.

Yan, J. and Pollefeys, M. (2006). Automatic kinematic chain building from feature trajectories of articulated objects. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 712–719, Washington, DC, USA. IEEE Computer Society.

Yang, C., Duraiswami, R., and Davis, L. (2005). Fast multiple object tracking via a hierarchical particle filter. In *In: International Conference on Computer Vision*, pages 212–219.

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Comput. Surv.*, 38(4).

Yilmaz, A., Li, X., and Shah, M. (2004). Object contour tracking using level sets. In *Asian Conference on Computer Vision, ACCV 2004, Jaju Islands, Korea*.

Zhang, X., Hu, W., Maybank, S., Li, X., and Zhu, M. (2008). Sequential particle swarm optimization for visual tracking. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8.

Zhao, Q. and Tao, H. (2009). A motion observable representation using color correlogram and its applications to tracking. *Compuer Vision and Image Understanging*, 113(2):273–290.

Zheng, Y. and Meng, Y. (2007). The pso-based adaptive window for people tracking. *Computational Intelligence in Security and Defense Applications, 2007. CISDA 2007. IEEE Symposium on*, pages 23–29.

Zhou, H., Yuan, Y., and Shi, C. (2008). Object tracking using sift features and mean shift. *Computer Vision and Image Understanding*.

Zimmer, C. (2007). From ants to people. *The New York Times*.

Ziou, D. and Tabbone, S. (1998). Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, 8:537–559.

Zivkovic, Z. (2004). Improved adaptive gaussian mixture model for background subtraction. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2:28–31 Vol.2.