OBJECT PARAMETER EXTRACTION FOR CONTROL IN VISION-BASED TABLETOP SYSTEMS

Octavi Estapé Ortega

Master Thesis submitted in partial fulfillment of the requirements for the degree:
Màster en Tecnologies de la Informació, la Comunicació i els Mitjans Audiovisuals

Supervisor: Sergi Jordà

Department of Information and Communication Technologies
Universitat Pompeu Fabra
Barcelona, Spain
September 2008

Abstract

reacTIVision is an open source computer vision framework for recognition and tracking of fiducials. It finds its main application in development of table-based tangible user interfaces (TUI) and multi-touch interactive surfaces (tabletops). This Master Thesis aims to improve the current reacTIVision system. It reviews the state of the art of the algorithms that can be used for parameter extraction from generic objects, focusing in the algorithms that can be used for control and real-time interaction. Among all shape description techniques, three different and complementary techniques have been implemented: some simple global descriptors give us the position and orientation of the shape with a rough classification, the polygon simplification can be used for user feedback and polygon recognition, and finally a simplification of the Medial Axis Transform (skeleton) is better suited for deformable objects.

Acknowledgments

This Master Thesis wouldn't have been possible without the unconditional support since the beginning from Sergi Jordà and Martin Kaltenbrunner, from the reactable team and Xavier Serra, director of the Music Technology Group. I would like to thank the rest of students at the MTG for their ideas and also Joanna Pleszek for her patience.

Contents

1.Introduction

This Master Thesis focuses on the implementation of algorithms for the extraction of parameters from generic objects that can be useful for control. The extraction of parameters has been implemented in the current reacTIVision [Kaltenbrunner and Bencina 2007] framework, a computer vision system for recognition and tracking of fiducials that is primary meant for development of table-based tangible user interfaces (TUI) and multi-touch interactive surfaces.

Chapter 2 describes briefly some of the novel input devices paying a special attention to table-based user interfaces and multi-touch systems, also known as tabletops. As the reacTIVision is the system to be extended in that project, its current state is also reviewed it this place.

In chapter 3 we will see the motivation that lays behind the project and we will review some of the existing image-based algorithms that can be applied for the extraction of parameters that can be used for control.

In chapter 4 we will review some of the techniques available for shape and skeleton description and in chapter 5 we will present the experimental results of the selected algorithms, namely some simple global descriptors, polygonalization using Discrete Curve Evolution and skeleton extraction using the Image Foresting Transform. In this chapter we also present an original idea that combines the skeleton with Discrete Curve Evolution in order to simplify it. Finally, in the last chapter we will conclude this Master Thesis with some ideas for future research.

2.Context and State of the Art

2.1.Input devices and Tabletops

2.1.1.Single focus devices

Until today the most frequently used and versatile computer input devices have been the keyboard and the mouse. The keyboard sends on/off events (key press/key release) as the user types and the mouse sends integer (relative) coordinates and also on/off events from its buttons.

For some applications there are other common input devices that send on/off and numerical events that are interpreted in different ways. For example joysticks have two numerical values x, y that usually are interpreted as speed; pen tablets send absolute coordinates (that can be interpreted as relative coordinates for compatibility with the mouse), a numerical value for pressure and another for tilt. MIDI devices also return numerical values for controlling music instruments, for example in MIDI keyboards there is a numerical value related to the pressure applied to every key. The applications have to be programmed expressly to support any of those input devices.

There are other less common input sensors that can be used in computers, like accelerometers, gyroscopes (that sense relative changes of speed in the three axis), magnetometers (which return the orientation respect the magnetic field of the Earth), ultrasounds to measure distance, Radio Frequency IDentification (RFID) cards [Finkenzeller 2003] to detect presence, and so on. Most of the those sensors are commonly used in robotics and lately are being integrated in computers and other consumer devices. For example the Apple iPhone has a built-in accelerometer, ambient light and proximity sensors together with a multi-touch display [1].

Input device	Physical values	Other interpreted values	Typical application
Keyboard	multiple on/off key events	key combinations (Caps, Ctrl, Alt)	Word processing
Mouse	relative x, y, key events	absolute x, y, drag and drop	Pointing
Joystick	absolute x, y, key events	speed	Games
Pen tablets	absolute x, y, pressure and tilt		Drawing
MIDI keyboard	multiple on/off key events with pressure		Music
Accelerometer and gyroscope (Wii, iPhone)	acceleration in x, y, z directions	relative x, y, z and angles (x, y, z)	Robotics, games
Magnetometer	absolute direction respect the north pole		Robotics
RFID cards	Identificator	presence	Identification

Table 1: Examples of input devices and the information obtained from them.

The table 1 contains all the input devices we have mentioned, along with their typical applications. In many cases an input device can be substituted by another, for example it is possible to move the pointer using the arrow keys of the keyboard or playing golf games using the mouse, but in many cases the success of an application depends on the input device chosen for it.

[1]Apple Reinvents the Phone with iPhone http://www.apple.com/pr/library/2007/01/09iphone.html

2.1.2.Multiple pointing devices

The current operating systems (at least Windows, Mac OS and Linux) and their applications have the assumption that there is only one point of attention or focus. Thus there must be only one pointing device (mouse, pen tablet, touchpad) working at the same time and only one GUI component (button, text field) with the focus. This greatly simplifies the programming of applications as it limits the possible concurrence problems, but also limits the use of the computer to one user at the same time. This is a handicap for using the computer in cooperative tasks, where several users share a computer or need to modify the same document, for example.

For those cooperative applications, lately there have been released libraries for Windows [Pawar et al. 2006][2] and Linux [Hutterer and Thomas 2007][3] that allow multiple pointing devices, i.e. more than one mouse pointer in the same computer. Nevertheless, the applications that want to take advantage of this new feature must be programmed accordingly.

Finally there are a number of devices that eliminate the separation between the input (pointer, buttons) and output (screen), such as the touch-screens. Today, the touch-screens are present everywhere, from cash machines to PDAs and mobile phones. Most of these touch screens support only a single pointer as they target single pointer operating systems, but recent products such as the iPhone have a touch-screen with multi-touch capabilities.

2.1.3.Tabletops

Tabletops address all those cited aspects: they have special sensors depending on the application, multi-touch sensing capabilities (multiple pointers) in the big screen that is in their surface. They are specially useful for collaborative use with multiple users, because the users interact directly with the table, touching it with the hands and fingers or moving everyday objects ("tangible user interfaces" - TUI) on it. The table reactions depend on what is on its surface and where it is, projecting images or producing sounds accordingly.

There are multiple projects related to tabletops. The DigitalDesk [Wellner 1993] was one of the first prototypes to merge a physical desk (with paper, pens and hands) with a computer that projected virtual objects on it. The metaDESK [Ullmer and Ishii 1997] presents a map that is moved/zoomed depending on the position of the plastic buildings that are on it. SenseTable [Patten et al. 2001] and BlueTable [Wilson and Sarin 2007] also use Bluetooth to retrieve information (such as pictures) from the objects put on the table. BlueTable is related to the commercially available product Microsoft Surface.

2.2.Common issues related to Tabletops

One of the tasks of tabletops is to recognize the objects that are on their surface and track their position accurately. This has been solved in various ways: using computer vision (as it will be discussed next), using FTIR [Han 2005] or using capacitive or resistive surfaces sensitive to pressure. Moreover, tabletops must give feedback in real-time and the images projected must be accurately attributed to the objects.

Most of the tables use hands and fingers as a pointer. As fingers occupy a large number of pixels, some tricks [Benko et al. 2006] have to be implemented for applications where there is a need of accuracy. As all fingers look the same, the position of the arm can be used to identify the user. It may also be important to identify postures (static positions of the hand) and gestures (movements) [Wilson

[2]Microsoft Windows MultiPoint Software Development Kit (SDK) http://www.microsoft.com/downloads/details.aspx?FamilyID=a137998b-e8d6-4fff-b805-2798d2c6e41d

[3]MPX: The Multi-Pointer X Server http://wearables.unisa.edu.au/mpx/

2006] to interpret the user's intentions like: changing the tool, clicking, moving, dragging, dropping.

As previously mentioned, multiple pointers (due to various users and hands) can originate concurrency problems that common applications and operating systems usually do not take into account.

Tables do not have a default orientation, usually people sit down in any of its sides (the most general case are round tables) while the usual computers' operating system have a clear top-down orientation, as screens are usually vertical. DiamondSpin [Shen et al. 2004] is a Java Toolkit that addresses this issue. For the reuse of existing applications, the operating system should allow every window to have an independent orientation.

2.3.Vision-based systems

Many of the tabletop systems cited before, such as PlayAnywhere [Wilson 2005], BlueTable [Wilson and Sarin 2007] or Microsoft Surface, are based on vision mostly because cameras are easily available, with acceptable spatial and temporal resolution and allow to sense and track multiple objects. These vision-based tabletops have different algorithms for recognition and tracking the objects on their surface, for example, PlayAnywhere [Wilson 2005] uses techniques for finger recognition and motion flow.

There are other fields with vision-based systems that also have to recognize objects in real-time, such as robotic vision and Augmented Reality. ARToolkit [Kato and Billinghurst 1999][Billinghurst et al. 2001], ARTag [Fiala 2005][4] and Studierstube [Schmaistieg et al. 2002] are open-source software libraries for developing vision-based augmented reality applications using fiducials. GraphTracker [Smit et al. 2007] uses graphs (lines connected in a tree or graph shape) and is robust to partial occlusions. Also, there are other more general libraries for computer vision like Mimas [Amavasai et al. 2005][5], XVision [Hager and Toyama 1998][6] and OpenCV [Bradski 2000][7].

2.4.The reactable

The reactable **[**Jordà et al. 2005] [Jordà et al. 2007] is a multi-user musical instrument implemented as a tabletop. Later on we will concentrate in reacTIVision, its vision system, as this is the main purpose of this work, but in this place we consider it important to show the reactable as an example of a successful tabletop application, outlining its technical requirements and as an inspiration for new applications.

The hardware of the reactable is a vision-based tabletop system with one infrared camera, infrared illumination for the visual system and a projector for the visual feedback as shown in figure 2. As it was conceived as a musical instrument for multiple performers, and none of them should have a predominant position, the shape of the table is circular.

When playing the reactable, the performer has several pucks (tangibles tagged with fiducials) available, each of them representing a module in an electronic music system. These pucks have different shapes (square, circular, pentagonal, ...) according to which family they belong: generators (wave generators, sample players), audio filters, controllers (for additional variables of other objects), global objects (that affect to many objects at the same time) or selectors. The specific function is represented by an icon on the puck. These tangible modules are connected automatically when they are placed on the table according to their proximity and other rules (only possible connections are created).

[4] http://www.artag.net/

[5] http://www.shu.ac.uk/mmvl/research/mimas

[6] http://www.cs.jhu.edu/CIRL/xvision.html

[7] http://opencvlibrary.sourceforge.net/

These connections, along with the waveform of the signals that are passing through them, are painted on the table by the projector. The main parameter of every module (its frequency for example) is set by the angle of orientation of the puck and some other parameter (active steps of a sequencer or amplitude) can be set touching with the finger some virtual buttons or a slider on the screen near the object. In figure 1 we can see some of these pucks and their connections.

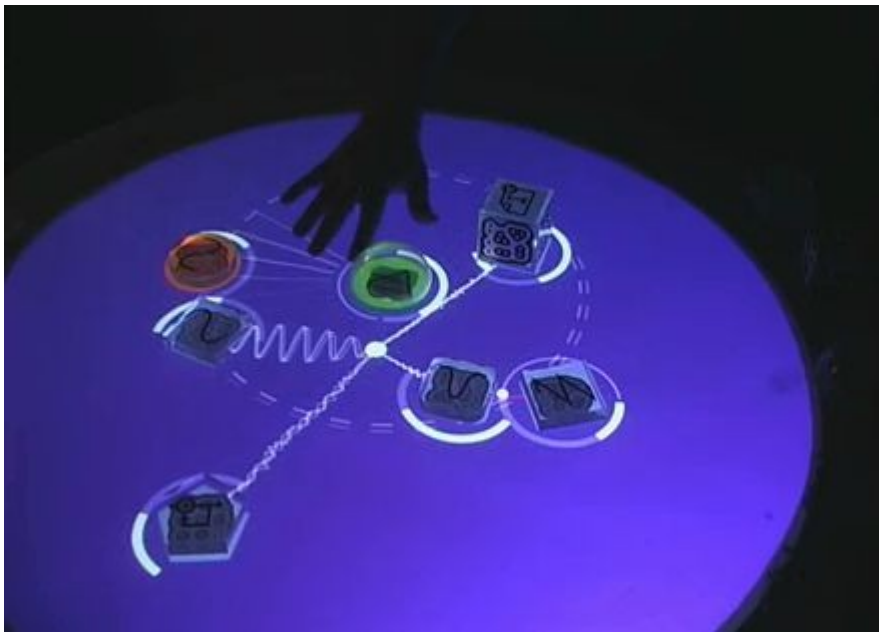


Figure 1: Performing on the reactable: a sine generator modulated by a sinusoidal controller (left), a triangular generator band pass filtered (right) and a sample player (top), all of them controlled with a global metronome (center) and with feedback (bottom). Frame of a demo video from http://www.iua.upf.es/mtg/reacTable/

The modularity of the reactable allows the performer to learn little by little, adding one puck at a time without being overwhelmed by a user interface with hundreds of controls. On the other hand, as there are so many possible modules available with practically an endless number of possible combinations of connections between them, even a skilled player may be continually learning. This is why it is enjoyed by very different people, ranging from families with children to skilled musicians.

From the technical point of view, any musical instrument needs a real-time interaction with the user, which in the case of the reactable imposes strict specifications (latency and temporal and spatial resolution) to its vision system.

2.5.reacTIVision

reacTIVision is a computer vision framework for recognition and tracking of fiducials. Originally, reacTIVision was designed as the reactable vision system to fulfill its technical requirements, but as it is an Open Source framework, it can be used in many other tabletop systems or other applications that need to track objects that can be tagged with fiducials.

A *fiducial* or *fiduciary marker* is an object specially designed to be quickly recognized by the computer vision system. Fiducials are used in the computer vision applications where it is possible to attach a marker to the objects that need to be tracked; they also can be attached to the background (walls of a room, for example) in order to track the movement of the camera [Naimark et al. 2002][Piekarski et al. 2004]. Fiducials have been used in robotics, augmented reality and even in microscopic observations [Dangaria et al. 2007].

The fiducials in the reacTIVision framework [Bencina and Kaltenbrunner 2005][Bencina et al. 2005]

are attached to the physical objects that need to be tracked. ReacTIVision analyzes in real-time the images captured by a camera, extracts the position and orientation of the fiducials and sends the information using a protocol (TUIO [Kaltenbrunner et al. 2005] or MIDI) to a higher level application (such as the reactable).

Even if reacTIVision could be used in any tracking application where the objects could be marked with a fiducial, its primary usage is to develop the table-based tangible user interfaces (TUI) and multi-touch interactive surfaces.

In those applications, a camera is situated under a transparent or translucent surface (e.g. smoked glass) which is used as a table where the physical (tangible) objects are situated (see Fig.2). Every object has a fiducial attached on its base, which is in the field of view of the camera. The user can move and rotate the objects over the table and, thanks to the vision system, interact with the application. It is also possible to use the table to project images in correspondence with the physical objects to substitute the computer monitor.

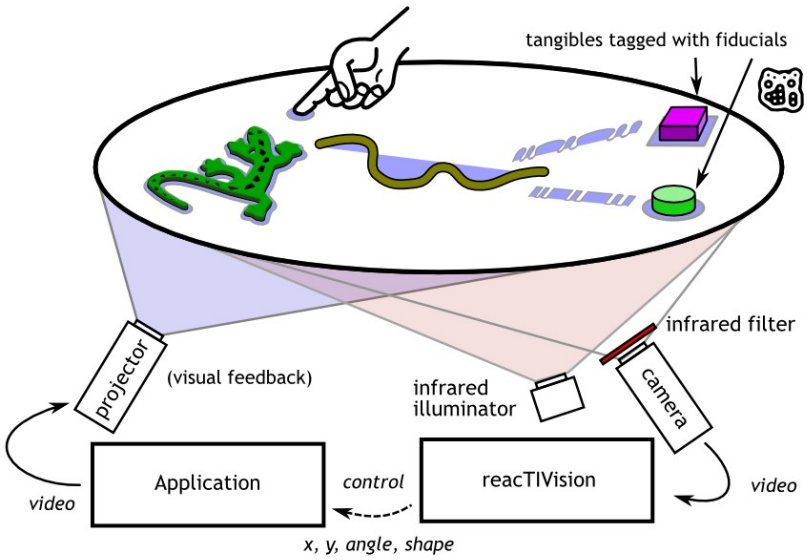


Figure 2: Diagram of the elements present in a tabletop using reacTIVision. There is infrared illumination that is invisible to the user, a projector that gives feedback to the user and the camera that has an infrared filter to filter this projection. The fiducials are attached to the objects (TUI) that are on the translucent surface of the table. The camera captures the images of the fiducials and fingers, computes their positions and rotation and sends the information to the application that gives video (and audio) feedback to the user. Our focus here is in the descriptors of the untagged objects. Figure based on Fig. 1 in [Kaltenbrunner and Bencina2007]

2.5.1.Vision algorithm in reacTIVision

The vision engine of reacTIVision processes every frame of the video with a chain of processors: equalization, threshold and extraction of the fiducial tree.

When the table is empty, all the image should have the same intensity of light. If there are differences in the illumination, the equalization adjusts the intensity of every pixel. Then an adaptive threshold is performed to convert the image from grayscale to black and white.

After that, the pixels of the same color are grouped in *blobs* and some statistics are computed for every blob, like the bounding box and area. Those statistics take into account a calibration grid defined to correct the distortion introduced by lenses, mirrors and angle of the camera. Every blob can contain other blobs inside it; black blobs can contain white blobs and vice versa. Then all the image can be

represented as a tree of blobs where every branch and leave have some statistical information of position and size.

The fiducials (see Fig.3) have been created in a way that every tree of blobs is unique. Every fiducial can be translated to a string, like if it was a barcode. To recognize the fiducials in the image the system has to find if any of the branches of the blob tree of the image corresponds to the blob tree of a fiducial.

Once a fiducial is recognized, the position and orientation is calculated using the final leaves in the tree that correspond to the fiducial. The leaves correspond to blobs in the fiducial that does not have any other blobs inside. To find the center of the fiducial we must add the center of all these leaves (black and white), which gives subpixel accuracy. To find the angle (orientation) we must add the center of only black leaves and calculate the vector from the center (see Fig.3).

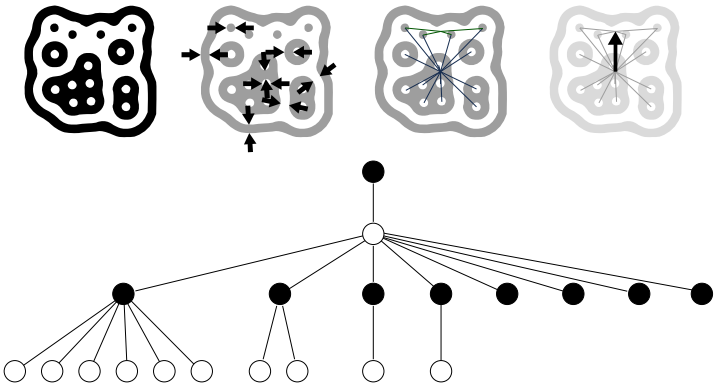


Figure 3: At the top there is a fiducial, the minimum separation between blobs, computing the center of all leaves and the center of black leaves (small points), and the orientation. At the bottom there is the blob tree for that fiducial: a big black blob, that has a white blob inside, with seven black blobs, that some of them have white blobs inside

The fiducials have been created with a genetic algorithm in a way that all this constraints are fulfilled (depth of the tree, uniqueness of the tree, the center of the leaves corresponds to the real center and the center of black leaves is in the upper center).

Fingers are recognized as leaves in the tree with a limited size and a round shape.

When a fiducial disappears from one frame to the next due to the noise or blurring motion, the system searches for similar blobs (size and position) in the tree to track it. This tracking is very fast as only uses the tree of blobs, which is very small if compared to the original video frame. The current implementation can track only changes in the position, but not in orientation. Moreover, this tracking allows many identical fiducials placed on the table to have a unique identification.

Finally, the fiducial information (fiducial number, unique identification, position and orientation) is sent to the application using MIDI or TUIO (an IP based protocol [Kaltenbrunner et al. 2005]).

3.New shape descriptors for vision-based tabletops

A fiducial based vision system like reacTIVision is robust and accurate, but it has many drawbacks. The parameters extracted from every object are limited to position and orientation and the objects have to be tagged which is not possible in small objects and it is not practical in many other objects such as hands or common use objects.

This project aims at an extension of the video-analysis to generic objects (without the fiducial tags) that can be rigid or deformable (see figure 4).

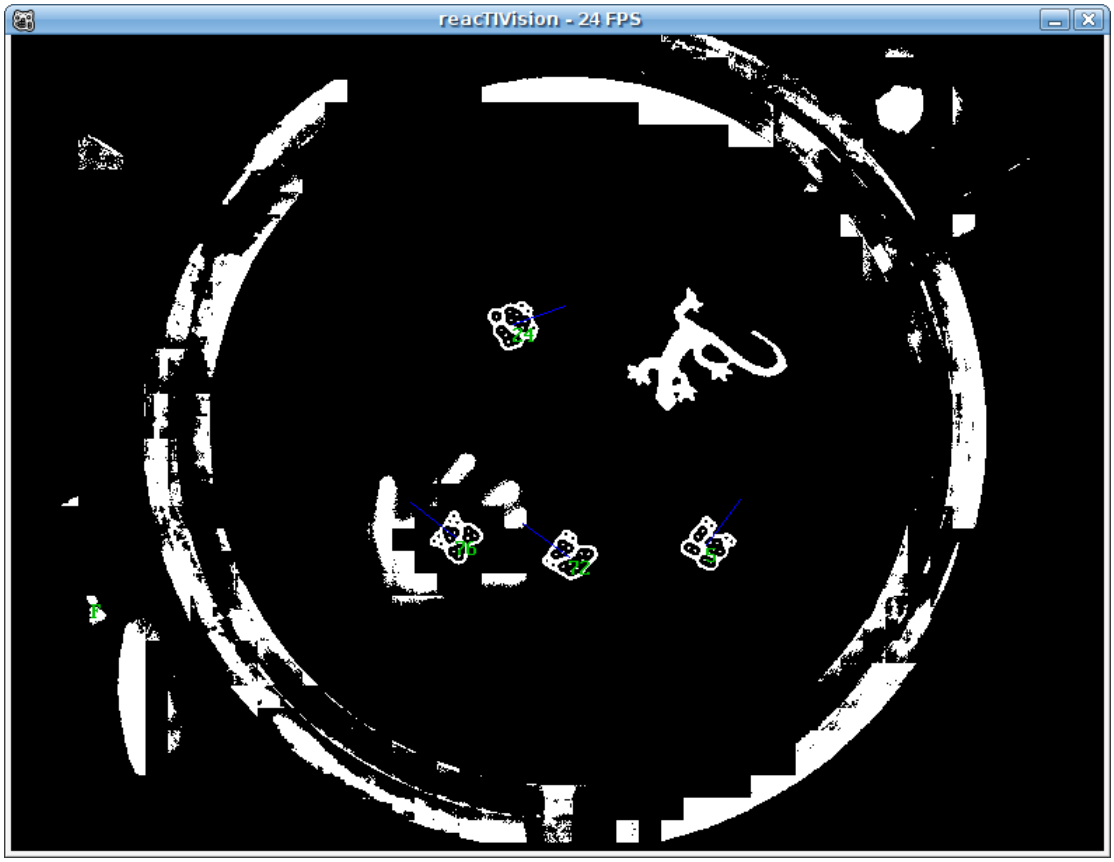


Figure 4: Segmented image in reacTIVision. There are four fiducials that are correctly identified and a deformable object (gecko) from which we want to extract parameters for control.

3.1.Applications

As reacTIVision is an Open Souce project, it is not easy to track all kinds of applications it is being used in. As reacTIVision was developed as a vision system for a tabletop, many of its users construct homemade tabletop setups. Nevertheless, as homemade tabletops are quite complex (computer, projector, table surface, lenses, mirrors and calibration) some users apply the reacTIVision system in simpler setups (for example moving fiducials directly in front of the camera).

Now we will review possible reacTIVision applications both in tabletop setups as well as in other environments.

3.1.1.Tabletop applications

There are several kinds of applications that are being implemented in a tabletop systems most of

them "ported" from the computer world or the table world: computer-like applications (such as photo organizers or map navigation), classic video games (like Shoot-them-all or Arkanoid), classic table games (like chess or cards) and other table activities (like drawing or cooking with a table that recognizes the available ingredients). In all these applications the mix of computer an table may enhance the user experience: for classical computer activities the multi-touch technology in a table allows a more natural and collaborative interaction, and for classical table activities the computer allows a dynamic user feedback with video and audio and also makes possible a remote interaction connecting two (or more) tabletops.

Nevertheless, a killer application for tabletops may come from a world not related to computers or tables, or at least it may not be just ported from an old application in the computer or table world. The reactable is probably the best example of such an application. Even if electronic music is performed on computers, the interaction with the user is totally different in a tabletop and existing metaphors such as Windows Icons Menus and Pointers (WIMP) are not the best inspirational source [Jordà 2008].

If we look at how we interact with tangible objects in the real world, we will see many differences with the usual computer interaction. The computer is conceived as having a single user with single pointer performing sequential actions, but in the real world many people (using both hands) can interact at the same time with any object. Moreover, in the real world the parameters to control an object are multidimensional, continuous, non-linear and coupled, but in the computer the typical controls (like buttons, checkboxes or sliders) have only one dimension, they are discrete (clicks) and they are perfectly linear and independent.

For example, compare the same task performed on a computer (figure 5) and with a real object (figure 6). When dragging a point on the computer, the other points do not move (they are independent) and the final result depends only on where the mouse is released. On the other hand, if we move a link of the chain of figure 6, neighboring links will move non-linearly and the final result will depend on all the gesture. The restrictions and simplifications in the computer controls may facilitate some tasks (allowing more precision for example), but they may forbid other tasks (in arts for example) that need intuitiveness, improvisation or expression.

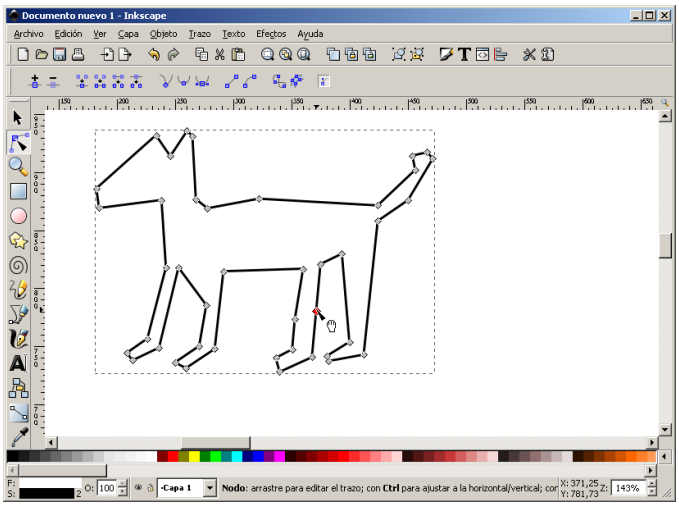


Figure 5: Drawing on a vector graphics application.

As the tabletop is somewhere between the real and the computer world, an oversimplification of their controls may impede the appearance of some important applications. In our case, the fiducials simplify greatly the description of the objects on a table, but they cannot describe correctly flexible objects that may be the key in some hypothetical killer application .

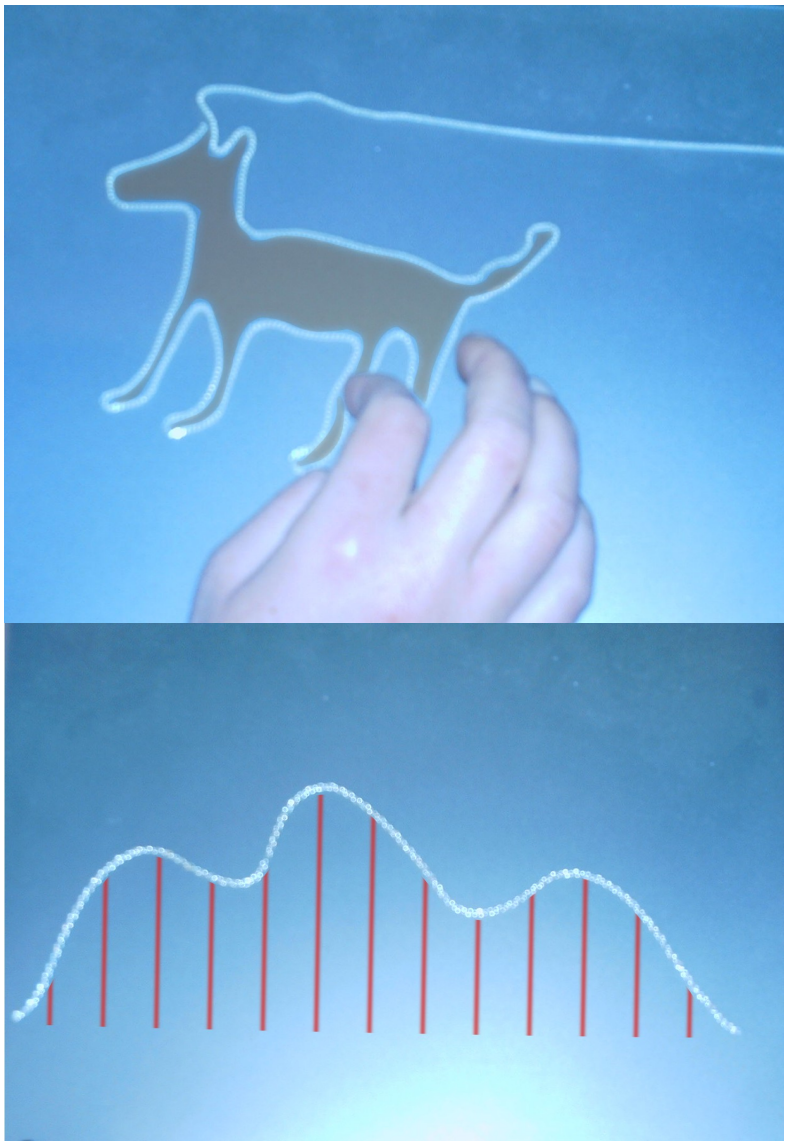


Figure 6: A drawing and a function defined with a thin chain on a tabletop.

At the same time, some classic table applications that are not possible or practical with the fiducial approach may benefit from shape description. Examples of such applications include table games with pieces too small to be tagged (Risk, Monopoly, and so on) or other usual table activities with generic objects (cooking, for example).

3.1.2.Augmented Reality applications

Fiducials are also being used in applications not related to tabletops. For example ARTag [Fiala 2005] uses fiducials for Augmented Reality applications, tagging people, objects and the background with fiducials. As opposed to the tabletop setup, here the fiducials may be at different distances from the camera, in any 3D orientation and may be occluded by other objects. For this reason every object has to be tagged many times to assure that at least one fiducial will face the camera directly. Moreover, if we need a rough description of the pose of a person, all their body parts (head, chest, arms, hands,

legs, and so on) must be tagged independently.

In those Augmented Reality applications, the use of so many fiducials to position deformable objects (like the human body) may not practical in many cases.

3.1.3.General purpose vision framework

Finally, shape description is useful in many other industrial quality control applications, such as part inspection to find shape irregularities or other defects. Even if reacTIVision is not intended for such applications and lacks many other important features such as preprocessing operations, filters or different segmentation techniques, some of the shape descriptors that we will review here may be implemented in existing general purpose vision systems (like OpenCV [Bradski 2000]).

3.2.Extraction of parameters from shape

The parameters that can be extracted from rigid objects are analogous to those that can already be extracted from fiducial marked objects. In rigid objects, most of the parameters that can be extracted from the image (derived from shape and texture) can be useful to recognize the object and just a few parameters can be used effectively for control (presence, position and orientation). Recognition of objects is useful as many objects that usually are on the tables cannot be marked with a fiducial because they are too small (like a pen, for example) or because there is important information on their surface (like documents or photographs). Even in the objects that can be marked with a fiducial, it would be more convenient not to mark them, but it is necessary that the strengths that the system currently has are not sacrificed: low recognition error, good accuracy and real-time performance.

Deformable objects can potentially have an infinite number of parameters useful for control. For example, using a long and flexible object (like a string or a small chain) we can extract the position of every point of its length and use it for a variety of applications like drawings or the definition of the frequency response of an audio filter (see figure 6).

3.3.Desired characteristics of the parameters

This research focuses on parameters that are modifiable by the user and are robust to noise (low jitter). If a parameter can change even if there are no changes in the real object or if a parameter does not change at all whatever the user does, this parameter is not useful in our case. We also have to keep in mind that those parameters must be sent through an IP based protocol (TUIO) and reducing the non significant changes in the parameters we also reduce the amount of transmitted data and the latency of the overall system.

That system is intended for real-time applications where exists a need for a small and constant latency. Then, the parameters should not be very computationally demanding and this computational cost must be regular (or at least with a small upper bound) for all the frames of the video.

It is also desirable that the original image (or a simplification of it) can be reconstructed from the parameters in order to be able to give user feedback with the projector of the table (see fig. 2). In some applications it is useful to have the contour of an object painted on the tabletop surface to let the user know that the object and its movements are recognized. This reconstruction of the object can be as rough as a circle or a square or the exact shape of the object.

To reduce the computation work, it is better to take advantage of the current segmentation in reacTIVision that binarizes the image. The original grayscale image is available but should be used only if it is completely necessary.

To summarize, the desired characteristics of the new descriptors for reacTIVision are:

1. Modifiable by the user, because they are intended for control

2. Robust to noise
3. Easy to understand because they must be used by many client applications
4. Capable to reconstruct of the original shape, to allow user feedback.
5. Fast to compute, as it is a real-time system

3.4.Shape description techniques

In this Master Thesis we will concentrate on the extraction of parameters from the shape of the object as it is more likely modifiable by the user. Other parameters like the texture could be useful for the recognition of the object, but it would involve the use of the original grayscale image and only would solve the problem of recognition, and wouldn't give any additional control parameter. The description of the shape can be useful for both purposes.

In [Zhang and Lu 2004] there is a review of the state of the art in shape description techniques. It reviews most of the techniques for shape description and representation. The algorithms are divided in contour-based (if the features are extracted only from the contour) and region-based (if the features are extracted from the whole region). Both categories are divided in structural and global approaches. The structural approach divides the shape in parts (called primitives) and the global approach analyzes it as a whole. For example, the area and moments are region-based with a global approach, Fourier and wavelet descriptors and perimeter are contour-based using a global approach and polygon and b-splines are also contour-based but use a structural approach because they divide the contour in parts (see figure 7). For a deeper review of these and other algorithms refer for example to [Loncaric 1998] and [Zhang and Lu 2004].

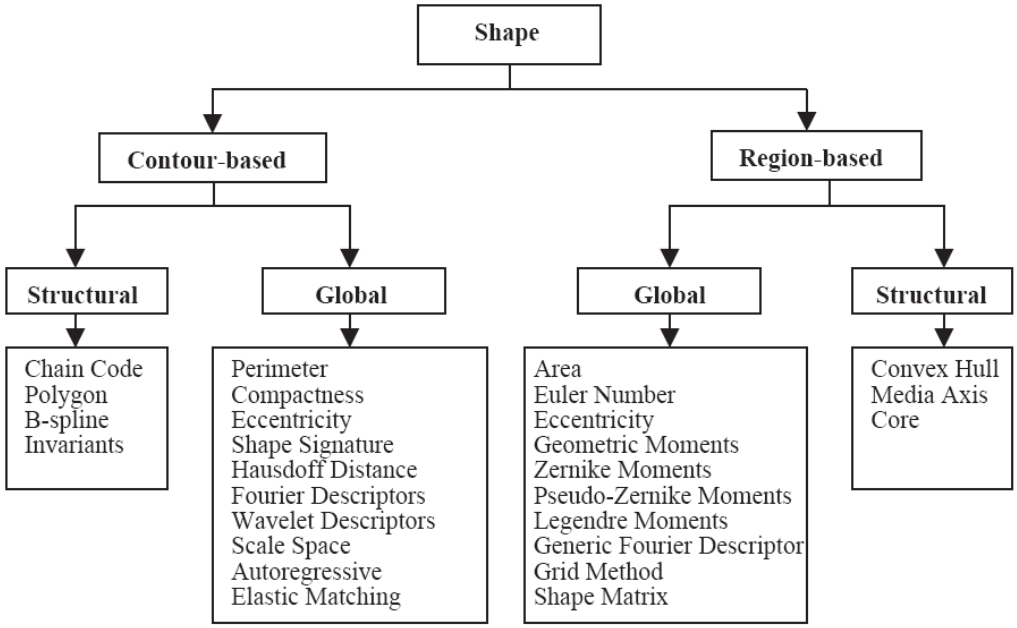


Figure 7: Classification of the different shape description techniques. This figure has been extracted from [Zhang and Lu 2004].

3.5.Choosing shape descriptors for tabletops

We will review deeply some of this descriptors in chapter 4, but here we will overview which of them are likely to be useful in the reacTIVision framework.

Polygons and b-splines are structural contour-based descriptors. In these cases, the contour is divided in parts that are approximated with a straight line or a cubic bézier curve for example. In figure 8 there is a binary image represented with a polygonal approximation and a cubic b-spline using potrace[8]. If a matching between different shapes must be performed, the polygon can be simplified [Latecki and Lakämper 1999a] and only the sharpest edges may be taken into account.

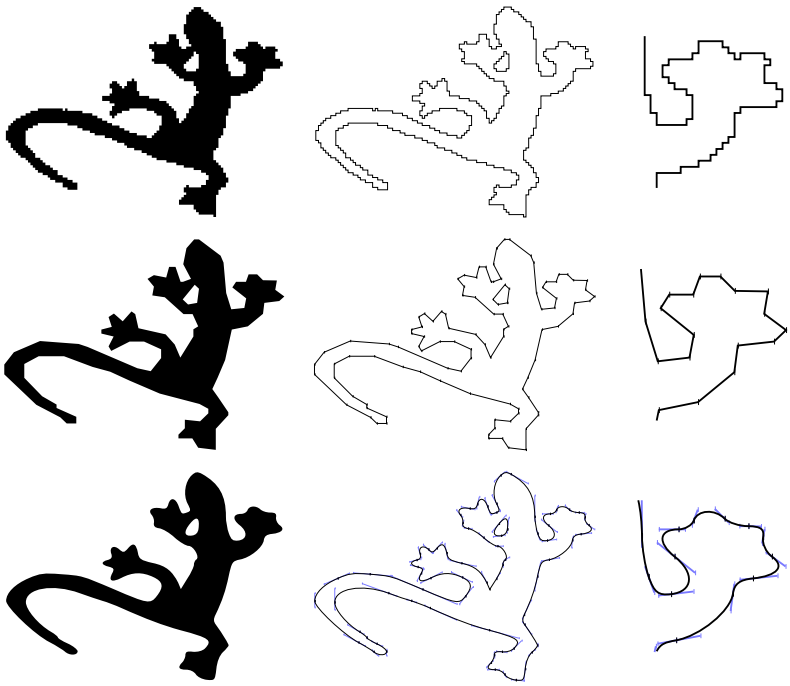


Figure 8: Binary image (112x85 pixels), the image traced as a polygon (106 nodes) and as a cubic b-spline (72 nodes). Tracing performed with potrace

In our case, is easy to use polygons for user feedback. The coordinates of the vertices of a simplified polygon can be sent using the TUIO protocol and it is easy to be drawn by the receiving application. In this case, it is important a consistent vertex selection in order that the position of the vertices does not change randomly because of the noise.

Among the global contour-based descriptors, Fourier and wavelet descriptors would be interesting in our case. These descriptors apply a Fourier or wavelet transformation to a shape signature. The shape signature is a transformation of the contour to 1D. Examples of shape signatures are the conversion of the contour to polar or complex coordinates, the distance from the centroid and the cumulative angle [Zhang and Lu 2001]. These descriptors are easy to compute and the shape can be simplified as desired. In figure 9 there is the representation of a contour using the first Fourier descriptors.

It is possible to reconstruct the original shape from the Fourier descriptors, but it is not as straightforward as in the case of the polygon. Moreover, the meaning of these descriptors is not easy to interpret for a general user and as they are global descriptors, a change in a part of the object may affect to all the descriptors.

Fourier Descriptors with different shape signatures and combined with other techniques have been successfully used for shape recognition and classification. For example in [Bartolini et al. 2005] use complex coordinates for the shape signature and combines it with Dynamic Time Warping, showing that the phase is important when using Fourier Descriptors.

[8]potrace is an open source library available at http://potrace.sourceforge.net/

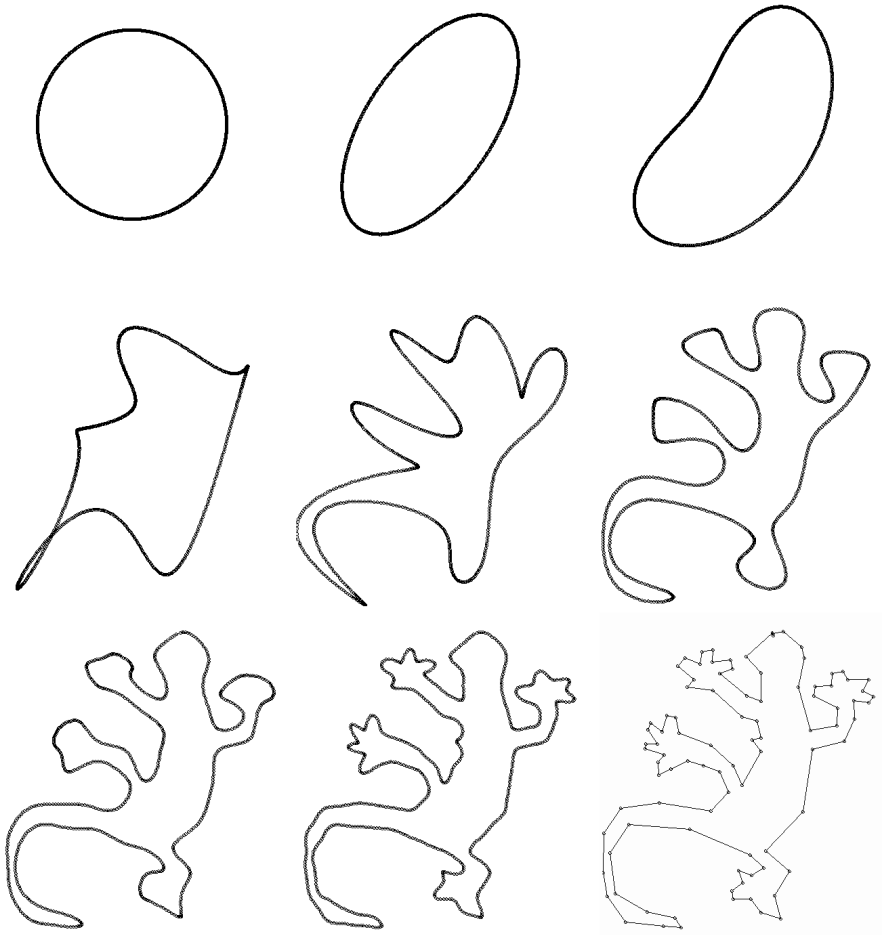


Figure 9: A contour of an image represented using only the firsts Fourier descriptors: 1, 2, 3, 10, 20, 40, 80, 160. The last one is the original image.

Most of the global descriptors (both contour or region-based), like some of the moments, are targeted to object recognition, as they are invariant to translation, rotation, affine transformation and deformations. In our case, this invariance is not always desirable as the parameters of translation, rotation and deformation are the ones that the user can modify and thus are useful for control.

Nevertheless, some of the global region-based descriptors such as the area, centroid and other low order moments are extremely simple to compute and may give some interesting parameters for control. For example, the centroid can give us an accurate and stable position of the object and using moments we can also find the orientation of the main axis of the shape. With this two simple descriptors we can substitute the fiducials in applications where there is no need of identification.

The Medial Axis and the skeleton are structural region-based descriptors that can be very useful for control. Roughly, the skeleton of a shape is the central line and the Medial Axis Transform is a skeleton that includes the width of the shape at every point (see figure 10). From the Medial Axis Transform it is possible to reconstruct the original shape.

Other structural region-based techniques include finding the skeleton of the polygon or dividing the polygon in triangles [Alexa et al. 2000][Felzenszwalb 2005].

As we have seen, global descriptors (such as Fourier Descriptors or moments) are useful mostly for recognition. Some of them (like Fourier Descriptors) can change in deformable objects and allow the reconstruction of the original object but they are not easy to understand and their local changes may affect all the descriptors.

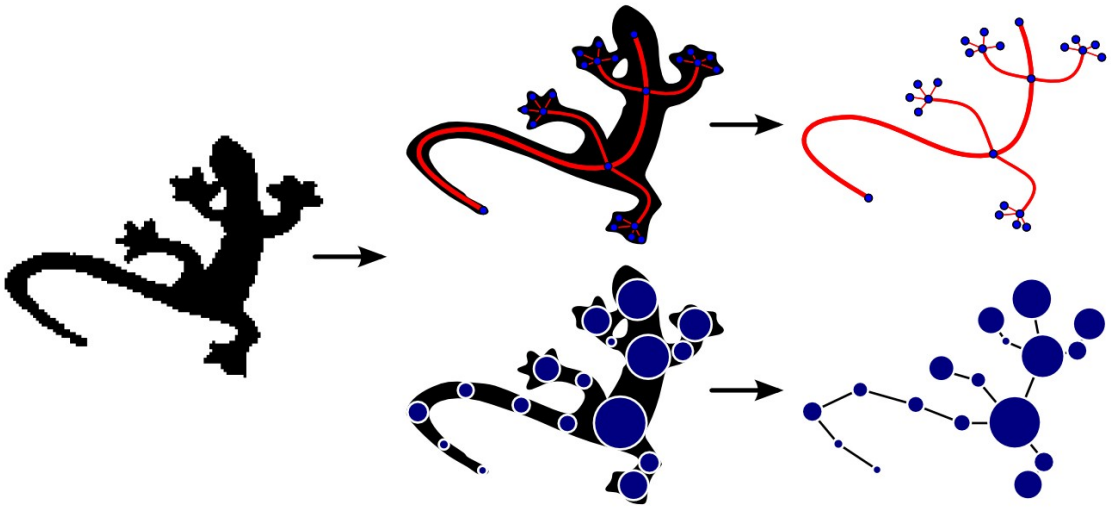


Figure 10: The skeleton represents the central lines of the shape, and the Medial Axis Transform includes the width of the shape at every point of the skeleton

On the other hand, structural descriptors (like polygon simplification or the Medial Axis Transform) keep the locality of the changes and are much more intuitive to understand and use for control. At the same time, those descriptors allow the reconstruction of the original shape for user feedback in the reacTIVision framework.

In the following chapter we will look deeper at these structural descriptors and some simple global descriptors and how can they be implemented in the reacTIVision framework.

4.Implemented shape and skeleton descriptors

In this chapter we will describe the concepts and methods implemented in the reacTIVision framework related to shape analysis. Among the variety of shape descriptors available, we have selected three of them that look at the shape from different points of view and have different goals and complexity.

Fist we look at global descriptors searching for a fast and rough approximation of the object. We compute the main axes of the shape and return an elliptical approximation of the object, that includes several parameters useful for control (position, angle, size and relation between the two axes) and allows a simple graphical representation for user feedback.

The second description of the shape is a polygonal simplification. The polygonal representation of the object is a structural contour-based technique that allows an accurate, compact and intuitive description of the object.

Finally, we explore the Medial Axis Transform (or skeleton) which is also a structural technique as it allows a piecewise decomposition of the shape. In contrast to the polygon, the skeleton is region-based, which means that it looks at the inside of the shape, returning information of its widths.

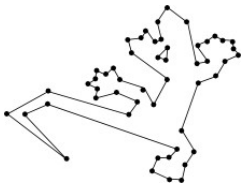
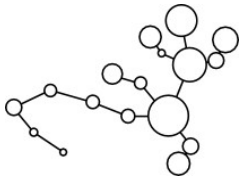
Operation	**Main parameters**	**Derived parameters**	**Graphical representation**
Blob extraction	area	size, pressure	
Fiducial recognition	x, y, angle, id	speed, acceleration	
Finger recognition	x, y		
Simple shape descriptors	x, y, length axis 1 and 2, orientation axis 1, perimeter, maximum direction, maximum distance	elliptical approximation, compactness, ratio of principal axes (elongation), shape classification	
Polygon simplification	x_i, y_i	number of vertices, angles, edge lengths, polygon recognition, shape reconstruction	
Skeleton (MAT)	x_i, y_i, r_i, links between points	length, max. and min. width, endpoints (protusions), necks, core (bends), seeds, shape reconstruction	

Table 2: Parameters extracted using every technique available in reacTIVision.

The main characteristic of these descriptors is its intuitiveness: it is important that any user of the framework can understand what these descriptors mean. This is why we deliberately rejected other descriptors like Fourier or higher order moments that have proven to be useful for shape analysis, classification and recognition but it is difficult to find a physical meaning for them. At the same time, these descriptors fulfill the characteristics pointed in section 3.3: they are modifiable by the user, robust to noise and useful to reconstruct an approximation of the shape of the object. The computational cost of each of them will be analyzed in chapter 5.

4.1.Contour extraction

Contour extraction is the first step in all contour-based algorithms (as they use only contour pixels) and in our case will be also useful in the skeleton even if it is a region-based algorithm.

The segmentation of the image and its division in objects (called *blobs*) is performed by previous steps in the image analysis of reacTIVision [Kaltenbrunner and Bencina 2007] . As a result, the contour extraction must transform a black and white image to an ordered list of points (clockwise or counter clockwise) for every object or blob.

There are many possible choices for the contour extaction that will affect the final result. First, the points of the contour may be centered at the center of an object pixel, at the center of a background pixel, at the sides of the pixels or at the vertices of the pixels (see figure 11). Second, we must choose the neighborhood of every pixel: we may consider only top, bottom, left and right pixels to be neighbors (4 pixel neighborhood) or we can allow diagonal pixels (8 pixel neighborhood). Third, if the contour can continue in different directions at a certain point, we must choose which one has priority (for example we can give priority to the diagonals, continue in the same direction or turn as much as possible clockwise or counter clockwise).

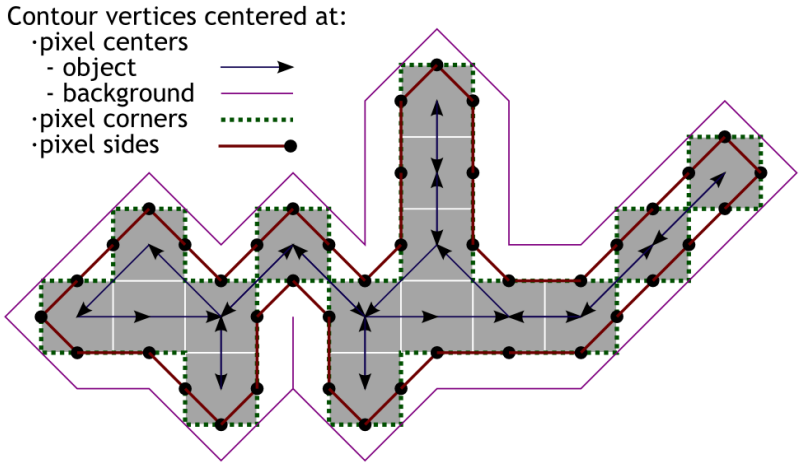


Figure 11: Choosing the position of the vertices when tracing a contour of a binary image

4.2.Global descriptors

As noted in [Peura and Iivarinen 1997], most of the shape descriptors for object recognition that have been proposed (like higher order moments, Fourier descriptors, or more sophisticated techniques), try to remove as much redundancy as possible but they lack a clear physical meaning and they are computationally expensive. They claim that descriptors for general applications should be computationally and semantically simple, with some degree of orthogonality between them, but also

allowing some redundancy (this redundancy can always be removed by standard techniques like principal component analysis).

They propose five simple descriptors derived from the boundary pixels of the shape: convexity, ratio of principal axes, compactness, circular variance and elliptic variance (see figure 12). These descriptors have an intuitive physical meaning, they are easy and fast to compute and they allow a rough shape classification.

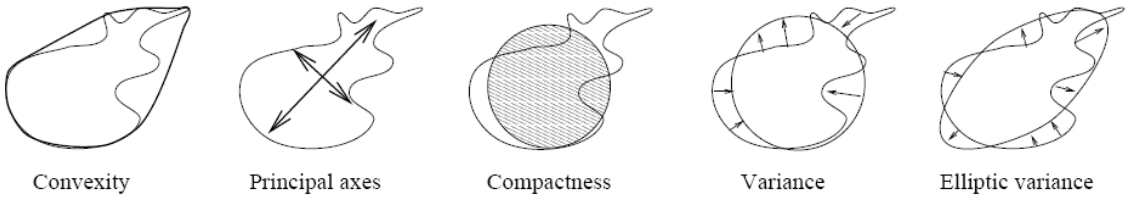


Figure 12: Simple shape descriptors derived from the boundary pixels. Figure extracted from [Peura 1997]

Unfortunately, as these descriptors are targeted for shape recognition, they are invariant to position, rotation and scale which are the main features that we use for control in the reacTIVision framework.

Anyway, following this simplicity guideline, we can find an elliptic approximation of the shape that will give us information of the centroid of the shape, an angle of orientation and the area. Moreover, the computation of the ratio of principal axes is straightforward from the ellipse axes (just a division) and the compactness is calculated directly from the perimeter and the area.

Other simple descriptors such as *convexity*, *circular variance* and *elliptic variance* [Peura and Iivarinen 1997] and many other descriptors [Zhang and Lu 2004] that can help us in shape classification or recognition could be also implemented. As our focus is in parameters for control and we don't want to increase excessively the computation work of the system, we will use only the ratio of principal axes and the compactness as they do not add any extra computation and, as we will see in section 5.2 they are appropriate for a shape classification that fits our needs.

4.2.1.Area and centroid

The area and the centroid (or center of gravity) are not considered shape descriptors, but only an intermediate step to compute them. Usually the descriptors are computed centered at the centroid (for translation invariance) and scaled by the area (so they are scale invariant). In our case, the area and centroid are important parameters for control: the centroid is an accurate position of the shape (with subpixel accuracy) and the area is often related to the distance from the camera or pressure, because if we press a flexible ball against the glass in figure 2 the area of contact will grow with the pressure. The area is the number of pixels of the shape and the centroid $\boldsymbol{\mu}$ is the mean position of these pixels:

$$Area = N \quad (1)$$

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{p_i} \quad (2)$$

where p_i is the position of every pixel.

4.2.2.Ellipse approximation

To compute the principal axes we must find the eigenvectors and eigenvalues of the covariance matrix:

$$\boldsymbol{C}=\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{p_i}-\boldsymbol{\mu})(\boldsymbol{p_i}-\boldsymbol{\mu})^T \quad (3)$$

where $\boldsymbol{\mu}$ is the centroid in (2) and $\boldsymbol{p_i}$ is the position of every pixel. Assuming that the covariance matrix has the form:

$$\boldsymbol{C}=\begin{pmatrix} C_{xx} & C_{xy} \\ C_{xy} & C_{yy} \end{pmatrix} \quad (4)$$

then using the characteristic equation we can compute the eigenvalues as:

$$det\begin{bmatrix} C_{xx}-\lambda & C_{xy} \\ C_{xy} & C_{yy}-\lambda \end{bmatrix}=0$$

$$\lambda^2-\lambda(C_{xx}+C_{yy})+C_{xx}C_{yy}-C_{xy}^2=0$$

$$\lambda_{1,2}=\frac{C_{xx}+C_{yy}\pm\sqrt{(C_{xx}+C_{yy})^2-4(C_{xx}C_{yy}-C_{xy}^2)}}{2} \quad (5)$$

and, if $C_{xx}<C_{yy}$ the normalized eigenvectors will be as:

$$\begin{pmatrix} \frac{C_{xy}}{\sqrt{C_{xy}^2+(\lambda_1-C_{xx})^2}} \\ \frac{\lambda_1-C_{xx}}{\sqrt{C_{xy}^2+(\lambda_1-C_{xx})^2}} \end{pmatrix} \text{ and } \begin{pmatrix} \frac{\lambda_2-C_{yy}}{\sqrt{C_{xy}^2+(\lambda_2-C_{yy})^2}} \\ \frac{C_{xy}}{\sqrt{C_{xy}^2+(\lambda_2-C_{yy})^2}} \end{pmatrix} \quad (6)$$

if $C_{xx}>C_{yy}$ then we just interchange them in the formula; if $C_{xx}=C_{yy}$ then the eigenvectors will be $\begin{pmatrix}1\\0\end{pmatrix}$ and $\begin{pmatrix}0\\1\end{pmatrix}$, just to avoid the negative square root or a division by zero.

If we compute the eigenvectors λ of an ellipse, they will be related with its radius as

$$r_{1,2}=2\sqrt{\frac{\lambda_{1,2}}{area}} \quad (7)$$

If we use the area of the shape, the radius that we find can be multiplied by their corresponding normalized eigenvectors (6) to define an ellipse that approximates the shape. In figure 13 we can see this ellipse that fits better the shape, and the same ellipse scaled to have the same area as the shape.

The ratio of principal axes (using r_1 and r_2 or λ_1 and λ_2) is a descriptor that can be used in shape classification [Peura and Iivarinen 1997] and it is also called *eccentricity* [Zhang and Lu 2004] or *elongation*. In the following, we will consider this definition:

$$Ratio\,of\ \ principal\,axes=\frac{r_2}{r_1} \quad (8)$$

where r_1 is the major axis and r_2 is the minor axis of the elliptical approximation. The possibles values are between 0 (for lines) and 1 (for circles).

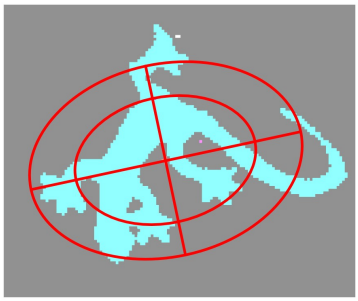


Figure 13: Ellipse approximation that fits the shape best (outer ellipse) and the ellipse scaled to have the same area as the shape (inner ellipse)

4.2.3.Maximum distance direction

In the reacTIVision framework, the fiducials have been designed to provide an angle of orientation that is a useful parameter of control. The above elliptical approximation gives us the major axis of the ellipse that can be used as the orientation of the shape, but the sense of this direction is not defined (for example, a vertical orientation may mean an angle of 90º or 270º). Moreover (as we will see in section 5.3), in the shapes that are approximated by a circle (when the major axis or the ellipse equals the minor axis), this angle is totally random. This means that we cannot use the elliptical approximation to find a stable angle for regular polygons (such as equilateral triangle, square, pentagon) and stars.

To complement the orientation of the elliptical approximation, we can search for the pixel of the shape that has the maximum distance from the centroid.

$$Maximum\ distance = argmax(\|p_i - \mu\|) \quad (9)$$

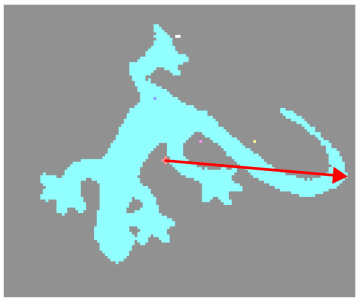


Figure 14: The point of maximum distance from the centroid can be used as an object orientation

This will give an angle that in the cases mentioned above will be more stable. For example, in the case of a square, the principal axis of the elliptical approximation has a random orientation, but the angle of the pixel in the maximum distance has only four possibilities. In shapes where there is not so much symmetry as in the regular polygons, this angle can have an stability comparable to the fiducials.

4.2.4.Compactness and circularity

The *compactness* measures how concentrated is the area of the object. The most compact two dimensional object is the circle, this is why the concept of compactness is closely related to circularity. Both concepts can be defined as the ratio of the squared perimeter and the area of the object [Peura

and Iivarinen 1997] [Zhang and Lu 2004]:

$$Compactness' = Circularity = \frac{Perimeter^2}{Area} \quad (10)$$

where the *perimeter* is the number of pixels of the contour of the shape. Using this definition, the circle is the shape with minimum value ($\frac{(2\pi r)^2}{(\pi r^2)} = 2\pi$) and other shapes will have a higher value. As it is desirable that the circle has the maximum circularity and compactness, the compactness can also be defined as [Peura and Iivarinen 1997]:

$$Compactness = \frac{2\sqrt{Area\,\pi}}{Perimeter} \quad (11)$$

and then the compactness of a circle will be 1 and all other shapes will have compactness between 0 and 1.

4.3.Polygon simplification

As we saw in section 4.1, we can see the contour of an object in the image as a polygon where every pixel in the contour corresponds to a vertex of the polygon and adjacent pixels in the contour share an edge of this polygon (see the first row in figure 8). Unfortunately, as we can see in figure 8, a small image can have hundreds of vertices in this polygon and it is very difficult to assess which of them are more important.

Our goal here is to simplify this polygon removing superfluous vertices. There are many algorithms for polygon approximation [Klette and Rosenfeld 2004]. We have chosen an algorithm called Discrete Contour Evolution for its simplicity and good results.

Discrete Contour Evolution [Latecki and Lakämper 1999a] is an iterative algorithm that at each iteration removes the least relevant vertex of the polygon. The relevance measure K of every vertex v at a given iteration depends of their two neighbors u and w according to:

$$K(v) = K(\beta, l_i, l_2) = \frac{\beta\, l_1 l_2}{l_1 + l_2} \quad (12)$$

where β is the *turn angle* at the vertex v, l_1 is the distance from u to v and l_2 is the distance from v to w (see figure 15):

$$l_1 = |\vec{uv}| \quad (13)$$

$$l_2 = |\vec{vw}| \quad (14)$$

$$\beta = \Pi - acos\left(\frac{\vec{uv}\cdot\vec{vw}}{|\vec{uv}||\vec{vw}|}\right) \quad (15)$$

For a motivation for the formula (12) see [Latecki and Lakämper 1999b].

As we see in figure 15, a large turn angle β means a higher relevance for the vertex (sharp vertices contribute more to the shape than straight lines) and, at the same time, long edges are more relevant than shorter ones.

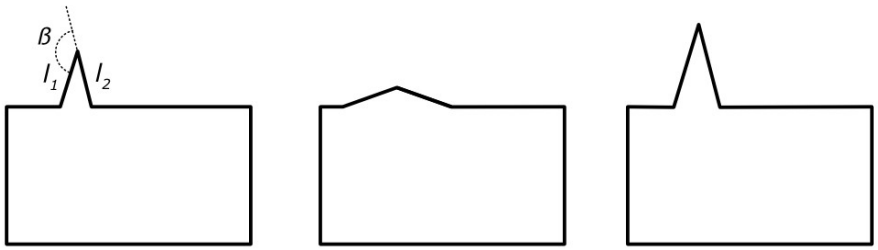


Figure 15: The relevance of a vertex depends on the turn angle and the length of the edges. The third case is the most relevant.

Even if the computation of the relevance measure K is local for every vertex (only two neighbors are taken into account), the algorithm performs a global simplification because only the vertex with minimum K in all the polygon is removed at every iteration. After the removal of the vertex v, the relevance measure of the neighbors $K(u)$ and $K(w)$ is updated.

The algorithm removes vertices until the minimum K is above some threshold. This threshold is the only parameter that the algorithm needs. In figure 16 we can see the original polygon and two simplifications using different thresholds. Notice that the results are as good as the ones in figure 8 but this algorithm is much more simple and allows to choose the degree of simplification.

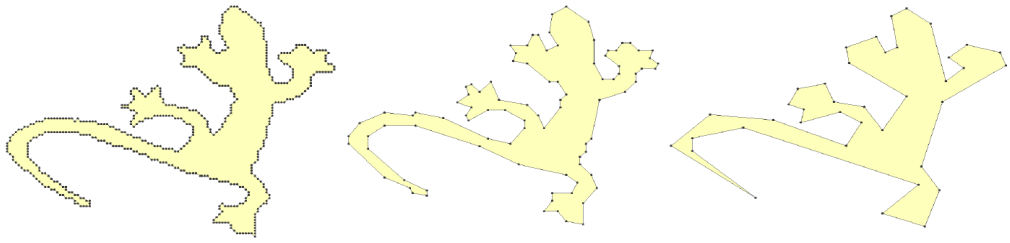


Figure 16: Original polygon and two simplifications using Discrete Contour Evolution.

The simplicity of this algorithm allows to simplify polygons with holes (that may be seen a polygon inside the polygon) and also polylines (polygons that are not closed). In the polyline case we just have to set the relevance measure of the endpoints to infinity (or a large number) in order not to remove them. As we will see in section 5.6, a modification of the algorithm can be used to simplify trees and graphs such as the skeleton.

4.4.Skeletons and Computational Geometry

The *skeleton* or *medial axis* of shape was first introduced by Blum in [Blum 1967] and it is an important descriptor that reduces the information but captures the essential structure of a shape. Since then, there have been many definitions and algorithms proposed to compute it [Smith 1987] [Leymarie and Levine 1992]. The common idea in all definitions for the skeleton is to find the main central line of the object. In general, the skeleton is a graph (not just a line): if the shape has branches, the skeleton will have branches as well, and if the shape has a hole, the skeleton will have a cycle.

The grassfire definition of the skeleton is the following: if we imagine a prairie with a given shape and we set fire to its contour, the front will propagate at a constant speed towards the center, then the points where two fire fronts meet would form the skeleton (see figure 17A).

The skeleton is also defined as the locus of the centers of the maximal disks. A maximal disk is a circle that is completely contained in the shape but it is not contained by any other maximal disk. In

practice, a maximal disk is any circle that is tangent to two points of the contour and its radius is proportional to the time it would take for the two hypothetical grassfire fronts to meet (see figure 17B).

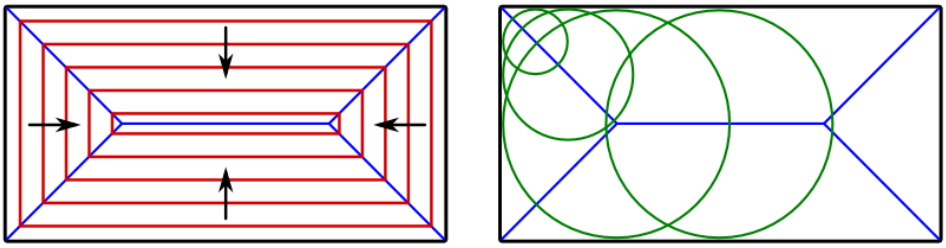


Figure 17: Skeleton of a rectangular shape: A) where two fire fronts meet in a prairie grassfire. B) locus of the centers of maximal disks

In figure 18 we can see some geometrical shapes and their skeleton. Notice that a hole in the shape results in a cycle in the skeleton and that different shapes (star and pentagon) may have the same skeleton. Notice also that the skeleton of a circle is a point, but the skeleton of the dodecagon (which is very similar to the circle) is a star. This has an important effect in a digital image, because the digitalization introduces many vertices that will result in skeleton branches. For example, a digitalized circle will have a star shaped skeleton instead of just a point.

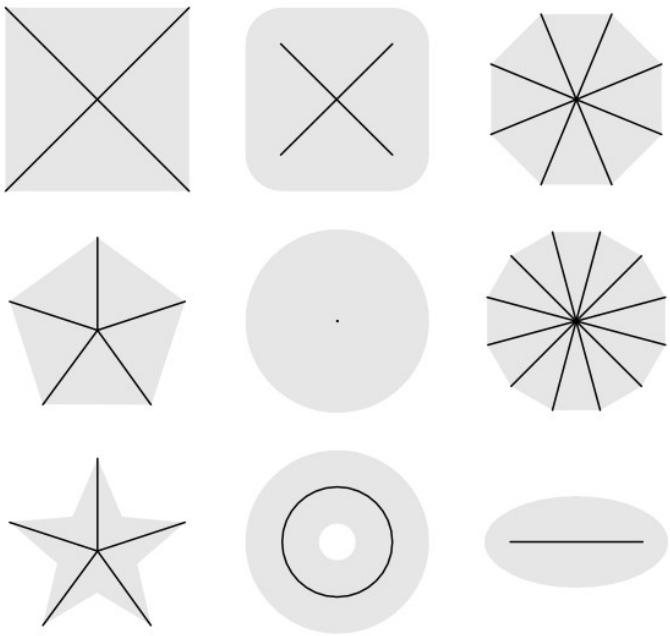


Figure 18: Skeletons of some geometrical shapes

One possible approach to finding the skeleton of a shape is using Computational Geometry techniques. Computational Geometry deals with objects with real valued coordinates (points, lines, polygons). In contrast, as we will see later, Digital Geometry deals with pixels in integer coordinates. The main advantage of Digital Geometry is its accuracy, but it requires more complex data structures to compute it. Now we will review some Computational Geometry concepts that are related to the skeleton.

For more detailed definitions and algorithms for computing the concepts reviewed in the next

sections, refer to a good Computational Geometry textbook such as [de Berg et al. 2000].

4.4.1.Voronoi Diagram and Medial Axis

The Voronoi Diagram of a set of points is the division of the plane in non overlapping convex polygons, called Voronoi regions, that are assigned to each of the points (see figure 19). The Voronoi Region of a point is the area of the plane that is nearer to this point than to any other point. For example, a Voronoi Diagram would be the answer to the question "Which is the nearest public phone (supermarket, pharmacy) for any point of the city?" (in this diagram every phone would be a point in the diagram and its zone of influence would be the Voronoi region. Voronoi Diagrams have applications in many fields, including biology, wireless network capacity, robot guiding and simulation of growth of crystals.

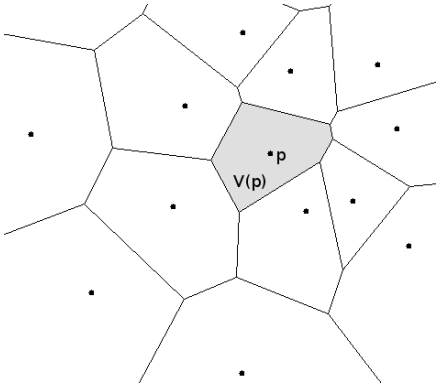


Figure 19: Voronoi Diagram of a set of points

In figure 20 we can see that the skeleton of a shape can be derived from the Voronoi Diagram of its contour pixels. The discretization of the shape introduces many branches in the skeleton, similarly as in the dodecagon that approximates a circle in figure 18. Some criteria must to be defined to prune the non-significant branches of this skeleton.

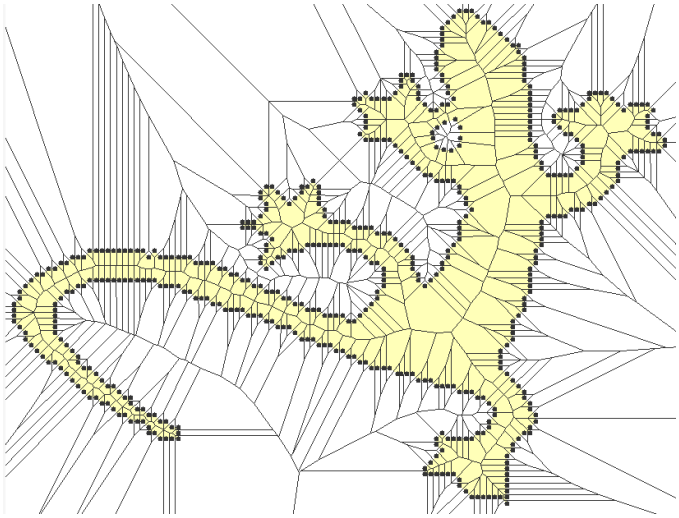


Figure 20: Voronoi Diagram where the points are the contour pixels of a shape. The skeleton can be derived from it but many branches have to be pruned

The Medial Axis of a polygon is the generalization of the Voronoi diagram to a set of connected lines. Every Voronoi region in this case is the area that is nearer to an edge or a vertex of the polygon than to any other edge or vertex. In this case, the Vornoi regions are not just polygons because their edges

may be parabolas. If we consider a polygonal shape, this Medial Axis corresponds to its skeleton (see figure 21).

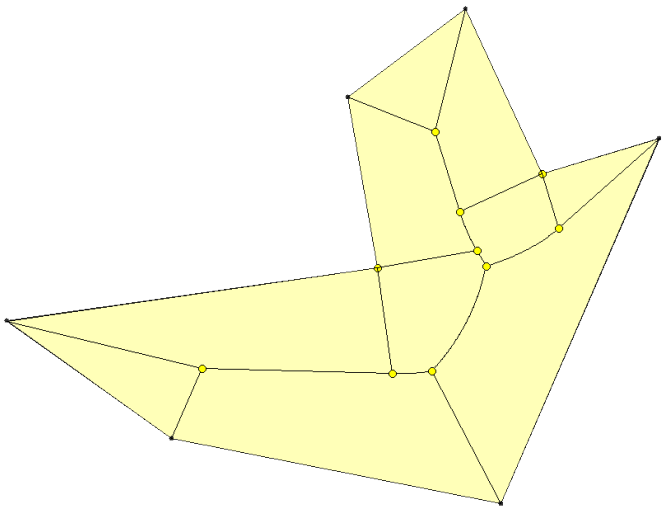


Figure 21: Medial Axis of a polygon

4.4.2.Constrained Delaunay Triangulation

A triangulation of a plane is the division of it in non-overlapping triangles that cover all the area and have their vertices on some predefined points. The Delaunay Triangulation is the triangulation that maximizes the minimum angle of its triangles (then the triangles tend to have angles near 60º because the equilateral triangle is the triangle that maximizes the minimum angle). The Delaunay Triangulation can be easily computed from the Voronoi Diagram because two points that have adjacent Voronoi cells, share an edge in the Voronoi Diagram. This duality can be seen in figure 22.

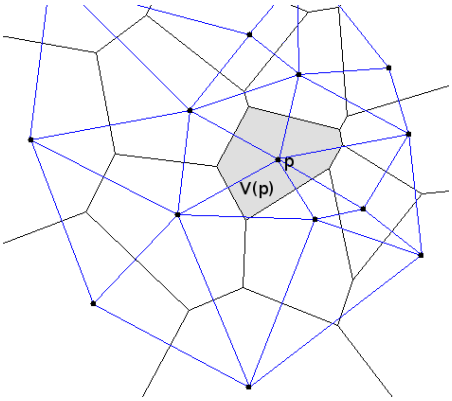


Figure 22: Voronoi Diagram and the corresponding Delaunay Triangulation

A Constrained Delaunay Triangulation is a triangulation where we force some of the edges, even if the resulting triangulation does not maximize the minimum angle. A Conforming Delaunay Triangulation is a Constrained Delaunay Triangulation where we allow the insertion of additional points (called Steiner points) so that the triangulation maximizes the minimum angle. The Constrained Delaunay Triangulation divides a polygon in triangles that can be used as an intermediate step to divide the polygon in convex regions (convex decomposition of a polygon). An approach to the skeletonization of a shape [Zou and Yan 2001] [Morrison and Zou 2007] decomposes a polygon in triangles with the the Constrained Delaunay Triangulation and then connects them by the midpoints of their edges (see figure 23).

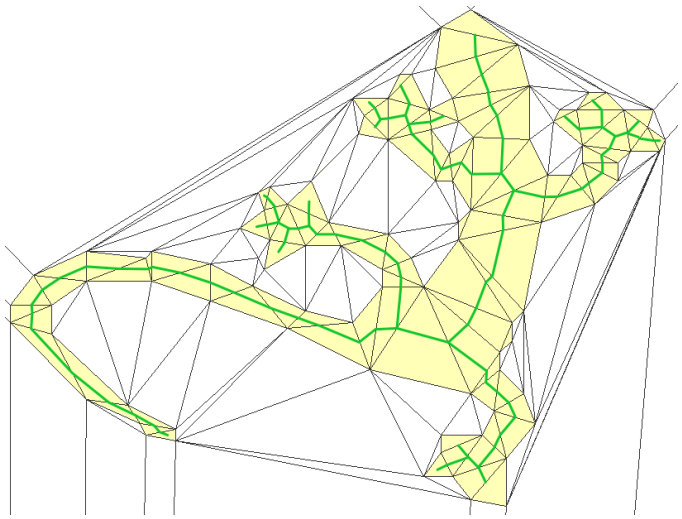


Figure 23: Constrained Delaunay Triangulation of a polygon and skeleton based on it

For a C implementation to compute Delaunay Triangulations, Constrained Delaunay Triangulations and Conforming Delaunay Triangulation see Triangle [Shewchuk 1996][9].

4.5.Skeletons in Digital Geometry

In digital images the skeleton can be represented as a binary image that encodes for every point if it belongs or not to the skeleton.

One of the main drawbacks of the skeleton is that is very sensitive to the noise in the contour as any small protuberance in the shape will create a branch in the skeleton. This results in very different skeleton topologies for similar shapes and it makes skeleton comparison a very difficult task. For example, compare the skeleton of the circle and the skeleton of the dodecagon in figure 18. Moreover, the spatial quantization of the digitalization (conversion to pixels), introduces many irrelevant vertices that will result in new branches of the skeleton.

To solve this problem, many algorithms have been proposed in the literature, that mostly can be grouped in two classes: contour smoothing and skeleton pruning. Contour smoothing simplifies the contour, removing the small protrusions caused by noise and thus many non-significant branches of the skeleton will not appear. Contour smoothing has the disadvantage that can displace the position of the actual skeleton and the degree of smoothing must be tuned so that important features of the shape are not removed. On the other hand skeleton pruning techniques try to remove the non-significant branches once the skeleton has been computed. Many criteria can be defined to decide which branches of the skeleton have to be pruned, most of them take into account the length of the branch, the length of the segment of the contour associated with it, or the lost area in the reconstructed shape if the branch is pruned.

As we will see, the Distance Transform is an important step to compute the skeleton and the MAT. The Distance Transform is a grayscale image that encodes for every pixel the distance to the nearest contour point (figure 24). The Distance Transform can be computed in many metrics, such as Euclidean, Manhattan or Chamfer distances.

[9]Triangle is freely available for non-commercial use at http://www.cs.cmu.edu/~quake/triangle.html

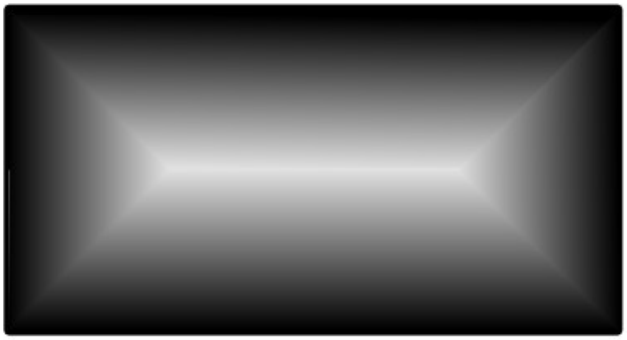


Figure 24: Distance Transform of a rectangle

Higher values in the distance transform correspond to central points of the shape. In the figure 25 we can see a shape and its Distance Transform. The skeleton corresponds to the "ridges" of the Distance Transform. These "ridges" can be seen very well if we represent the Distance Transform in a three dimensional view using heights (figure 25): the "ridge" is at the maximum of distance from two sides of the object.

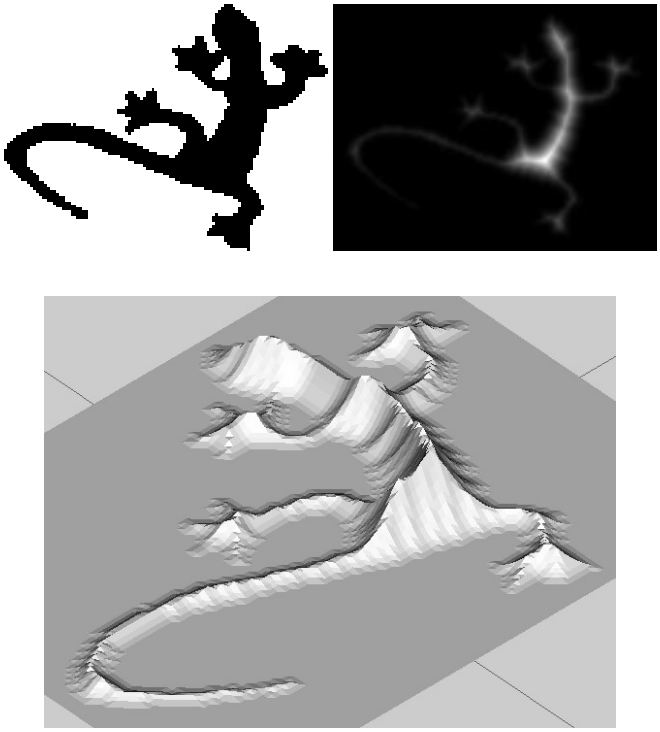


Figure 25: A shape, its Distance Transform represented using gray values and the same Distance Transform represented using heights. The skeleton of the shape is composed by the ridges in the Distance Transform

For a recent review of algorithms to compute the exact Euclidean Distance Transform with implementations of many of them see [Fabbri et al. 2008][10]. According to this survey, the algorithms from [Meijster et al. 2002] and [Maurer et al. 2003] are fast and easy to implement. Moreover, both

[10]Implementation of Distance Transform algorithms available at http://animal.sourceforge.net/ Animal (An Imaging Library)

algorithms can be used in arbitrary dimensions and they can be implemented in parallel architectures.

The Medial Axis Transform (MAT) is an skeleton that not only contains the position of its points (the center of the maximal disks), but also their distance to the contour (the radius of the maximal disks). In figure 26 we can see the MAT of a rectangle: the lines are the skeleton and their gray value encode the radius of the corresponding maximal disk at this point. The MAT (figure 26) can also be seen as the Distance Transform (figure 24) restricted to the skeleton points (figure 17).

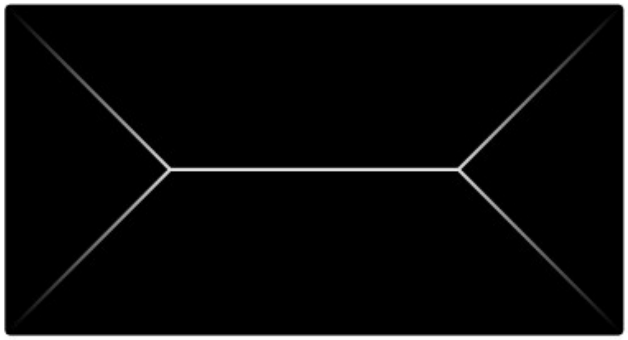


Figure 26: Medial Axis Transform of a rectangle

It is important to notice that the MAT preserves all the information of the shape and from the MAT the original shape can be exactly reconstructed (a simple, but not optimal, way to reconstruct the shape would be by drawing circles centered at the skeleton points with the radius equal to the value of the pixel).

As well as in the Computational Geometry case, here we can define a Discrete Voronoi Diagram that for every pixel tells us which is the nearest contour pixel of the shape. In this case, the Voronoi region is a set of pixels and boundaries of these regions are the skeleton of the shape.

In figure 28 we can see all the concepts we have just reviewed: the Distance Transform, the Discrete Voronoi Diagram and the Medial Axis Transform.

It is worth to mention that more sophisticated representations that can be derived from the skeleton can simplify greatly the problem of shape matching. *Shock graphs* [Sebastian et al. 2004] converts the graph of the skeleton (or the MAT) to a tree where nodes are shocks in the graph of the skeleton. The skeleton is divided in monotonic parts, i.e. parts of the skeleton where the corresponding radius of the shape either increases or decreases. These segments are called *shocks* and they are classified in four types: *protusions* (fingers of a hand, for example) that correspond to endpoints in the skeleton, *necks* that correspond to local minimum radius in the MAT, *bends* that correspond to inner parts or the core of the skeleton and *seeds* that are skeletons composed by a single point (like in circles). These shocks form a graph, just like the skeleton, but this graph is then represented as a tree that can be used for shape matching.

4.6.The skeleton computed using the Image Foresting Transform

The Image Foresting Transform (IFT) [Falcao et al. 2004] is an algorithm that looks at the image as a graph: every pixel of the image corresponds to a node in the graph and neighboring points in the image have an edge in this graph. In relation to what image operation we are interested in, we set different costs to the edges of the graph and we set some seed pixels each of them with a handicap. Then to compute the image operation that we are interested in, we just have to find the shortest path from every pixel of the image to a seed pixel. This shortest path is computed using a modification of the

Dijskstra's shortest path algorithm.

Dijkstra's shortest path algorithm [Dijkstra 1959] computes the shortest paths (and the corresponding costs) from all the nodes of a graph to a destination of the graph. The result of the Dijkstra's algorithm is a tree, where the destination is the root. In figure 27, there is an example of a graph with the cost of its edges, the cost from every node to the destination (the central pixel) and the shortest path (black arrows). In the case of the IFT, this destination of the paths is called a *seed* and in this case there may be more than one seed, and then the result is a forest (a set of trees); this is why the algorithm is called *foresting*.

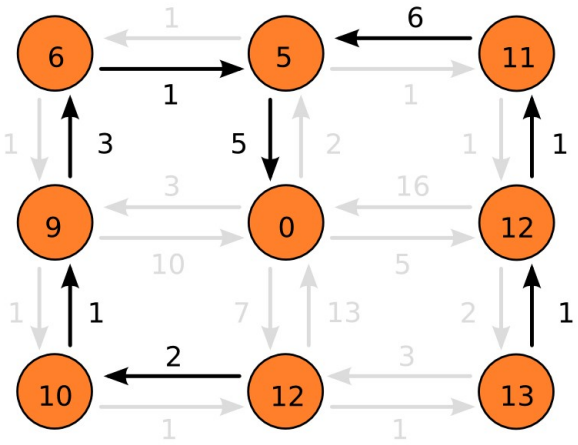


Figure 27: Small graph and the shortest path from all nodes to the central one after running Dijkstra's shortest path algorithm

The main output of the algorithm is the edges that form the shortest path. This information can be encoded as an image (a matrix of the size of the original image) that says which is the cheaper neighbor for each pixel of the image. This image is called the *predecessors image* because when we finish the algorithm, following the predecessors of any pixel we will arrive to the cheapest seed.

Another output of the algorithm is the *cost image*, that is an image that encodes the minimum cost from every pixel to a seed.

The algorithm starts storing the *seeds* (that are pixels), along with their *handicap* (the initial cost) in an ordered queue (the queue orders the pixels according to their cost). This handicap is also stored in the cost image, that will have an infinity value for all other pixels as we don't know yet the cost from them to a seed. Similarly, the predecessors image is initialized to NULL to all the pixels but the seeds, that will point to themselves.

After this initialization, the algorithm is very similar to Dijkstra's shortest path algorithm. It extracts the cheapest pixel from the queue. As this is the cheapest pixel of all, we are sure that its cost is the minimum possible cost (any other path to it would pass through more expensive pixels). Then, for every neighbor of this pixel we compute the cost to arrive to them passing through this pixel (adding the cost of the direct path between the neighbor and the pixel to the cost of the pixel). If this neighbor was already in the queue and the new cost is lower than the old cost, we update the predecessors image (now the neighbor points to the pixel) and we update the cost in the queue. If the neighbor it is not in the queue, we add it along with the cost we just computed. After updating the neighbors (if necessary) or adding them to the queue, we extract the next pixel of lower cost of the queue and repeat all the process.

As the algorithm evolves, all the pixels of the image are inserted in the queue. Once a pixel is extracted from the queue, it will never be inserted again because all the remaining pixels in the queue

have higher costs and thus it is not possible that a cheaper path to the pixel that we have just extracted appears. Then every pixel in inserted in the queue exactly once. As we are extracting pixels from the queue, eventually the queue will be empty, and the algorithm will end.

The IFT can be used for operations that need to find the minimum path using some geodesic distance. For example, the watershed transform groups pixels around local minima and is used for image segmentation. In this case the seeds of the IFT are these local minima and the cost function is the difference of intensities between pixels. Another example is contour tracking, that finds a path between two pixels according to some geodesic criteria. The IFT has been implemented in FPGAs reporting speedups of 5600 upon the software implementation [Cappabianco et al. 2007].

In the case of the computation of Discrete Voronoi Diagram [Falcao et al. 2002], the contour pixels of the shape are the seeds of the IFT and the Euclidean distance is the cost function of the algorithm. The predecessors image (the output of the algorithm) corresponds to the Discrete Voronoi Diagram because it tells which is the nearest contour point for every pixel of the image. On the other hand, the cost image corresponds to the Distance Transform as it tells which is the distance from every point to the nearest contour point. Finally, the skeleton is the boundary between Voronoi regions in the Discrete Voronoi Diagram, and the MAT is this skeleton plus the information of the cost. In figure 28 we can see an example of all these concepts: the Distance Transform, the Discrete Voronoi Diagram and the Medial Axis.

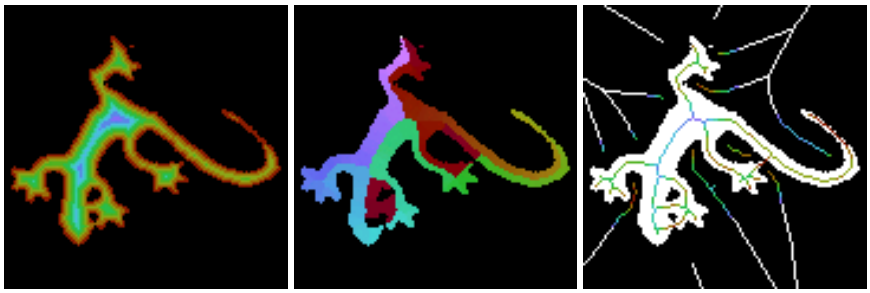


Figure 28: Distance Transform, Discrete Voronoi Diagram and Medial Axis Transform computed using the Image Foresting Transform

In fact, as we saw in figure 20, the skeleton of an image using the Voronoi Diagram (or the Discrete Voronoi Diagram) has many branches that have to be pruned according to some criterion. The criterion used in [Falcao et al. 2002] is that only skeleton pixels caused by contour pixels that are far from each other, are kept when simplifying the skeleton. In other words, only skeleton points associated with a long perimeter are kept.

In figure 29 we see an example of some polygons: a skeleton point is caused by two contour points. The distance along the contour between the two contour points is used to asses the relevance of the skeleton point. Note that the skeleton point of the star is more relevant than the one of the regular polygon. Moreover, regular polygons with more sides (that are more similar to a circle) have skeleton points with a small relevance. Then the skeleton can be simplified removing the less significant points. What is more, the resulting skeleton is always connected, because the extreme points of a branch of the skeleton always have a smaller associated perimeter than the points staying closer to the center.

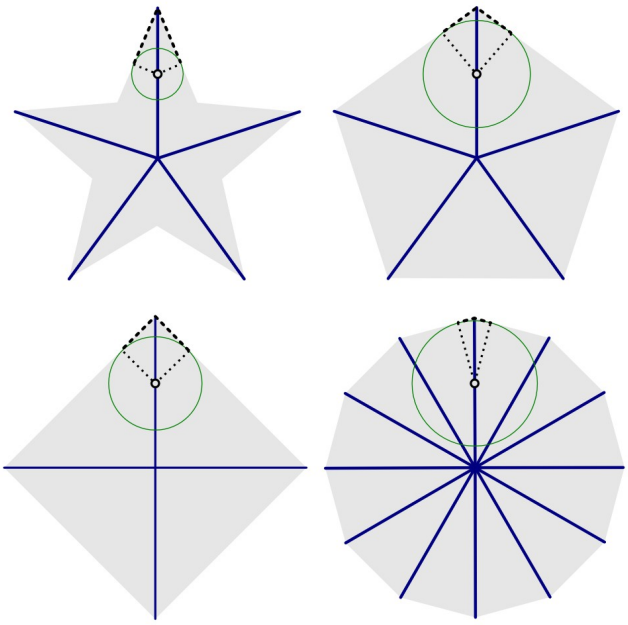


Figure 29: Perimeter associated to a point of the skeleton (thick dotted line). The length of this segment is used to asses the relevance of the skeleton point.

This relevance of the skeleton points can be easily computed from the Voronoi regions, because every skeleton pixel is in the boundary of at least two Voronoi regions and, at the same time, every Voronoi region corresponds to a contour pixel. As a result, from every skeleton pixel we can associate two contour pixels. If we number the contour pixels in clockwise order, for example, the computation of the perimeter associated to every skeleton point is straightforward. In figure 28 we can see an example: every contour pixel is encoded with a color in its Discrete Voronoi Diagram and the points of the skeleton are only those where the separation of colors is high (which means that the associated perimeter is high).

5.Implementation details and results

In this chapter we will present the results derived from the implementation of the existing algorithms described in chapter 4. We will present some details that have been important in the implementation, we will evaluate information obtained from the algorithms and we will show how it can be used in the reacTIVision system.

5.1.Contour extraction

In reacTIVision objects are white, the background is black, and, in the current version, they have 4-neighborbood connection. Among all the possible options we have, we choose to traverse the contour clockwise (this does not affect the final result), allowing diagonals (so the contour is smoother) and with the points of the contour centered at background pixels (this makes the object thicker). The algorithm to find it is simple: find the top-left corner and, starting in the north direction, traverse the contour turning to the right if possible (including the diagonals), until you reach the beginning.

In figure 30 there is an example of the extraction of contour from an object (the gray pixels represent neighboring objects that are not connected to this object).

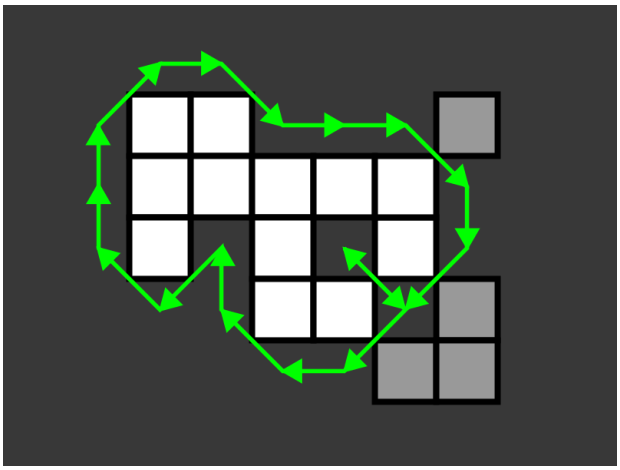


Figure 30: In reacTIVision objects are white, with 4-neighborhood connection. The points of our contour are centered at the background pixels, ordered clockwise and allowing diagonals.

One important detail is where we store this path and how long it can be. In figure 11 and also in figure 30 we can see the effect of choosing the position of the contour points. If the points of the contour are centered in the object pixels, the background pixels or the corners of the pixels, the contour may traverse two times the same point. This means that we cannot store the contour in a matrix of the same size as the image (a matrix of pointers to the next pixel of the contour, for example) because a position in this matrix could be written twice and a portion of the contour would be lost. This error was found in the implementation of the IFT that we will use in section 5.5. On the other hand, if we center the contour points at the sides of the image pixels (the last case in figure 11), there are twice as possible locations as pixels in the image. In any case, the contour can be (in the worst case) twice as long as the number of pixels in the image. In our case we choose to store the contour positions in an array (that allows to store the same position twice).

5.2.Simple descriptors for classification

In section 4.2 we described how to compute some simple and intuitive shape descriptors that will be useful for both control and classification. Now we will see if the ratio of principal axes and the compactness can be used as a simple classifier.

Intuitively we may think that the ratio of principal axes (elongation) may be useful to discriminate long and straight objects such as pencils from round or square objects. On the other hand, the compactness (that relates perimeter and area) will help us to separate simple objects (circles and regular polygons) from complex ones (with holes and protrusions that increase the perimeter).

In order to test this intuition, we computed these two descriptors for the MPEG-7 Shapes B database [11] which includes 1400 shapes and 70 classes. In figure 31 we see that the results match the intuition above: the circle is the most compact object and at the same time its ratio of principal axes equals 1. Elongated objects are in the other extreme in the ratio of principal axes descriptor and objects with many protrusions and holes have a low compactness. Moreover we see that all the shapes are well

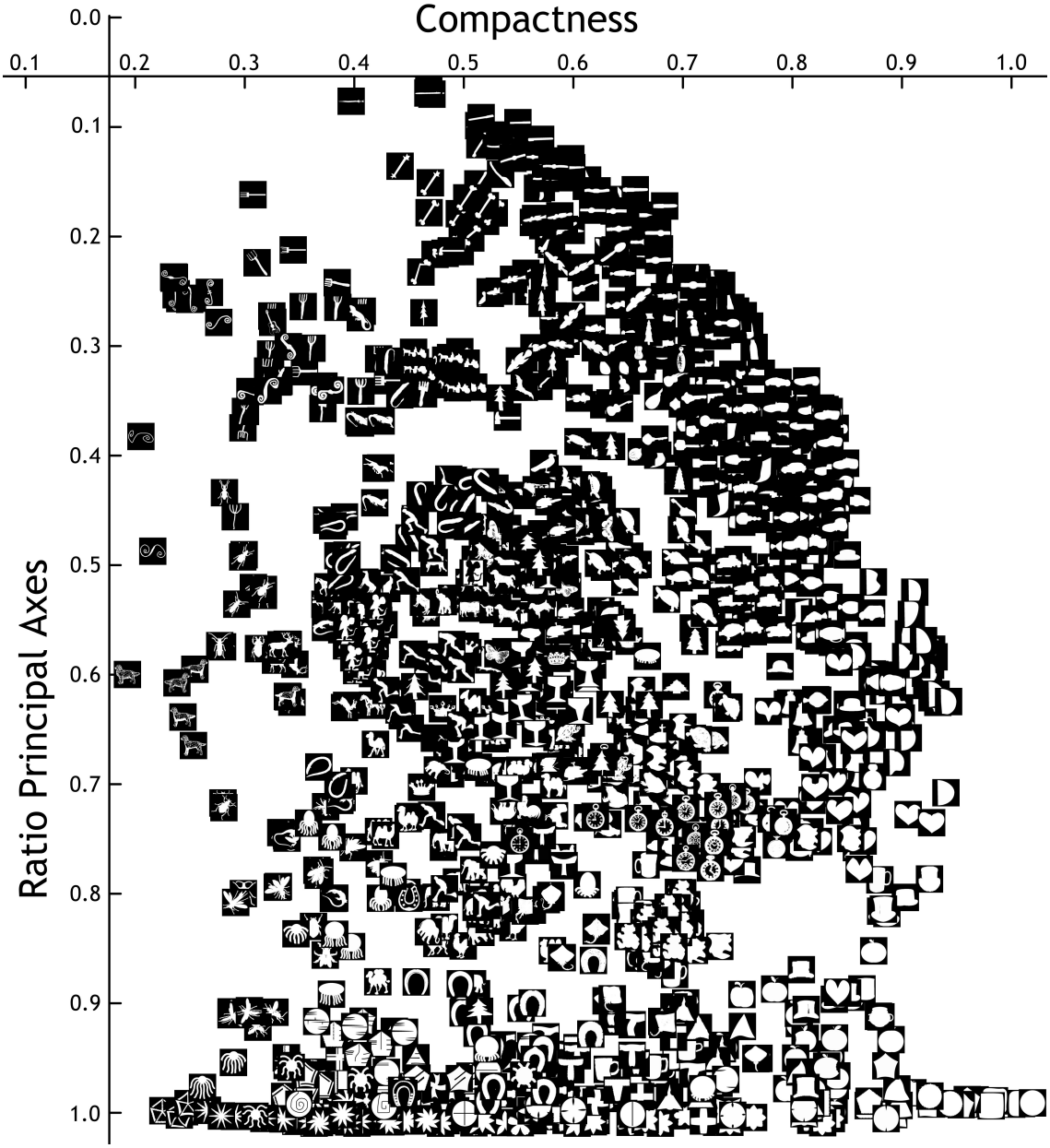


Figure 31: Ratio of principal axes versus compactness in the MPEG-7 Shape B database

distributed in all the area as the correlation coefficient of these two descriptors for this dataset is -0.03.

[11]The MPEG-7 Shape B database was downloaded from the web page of Professor Latecki: http://www.cis.temple.edu/~latecki/research.html

In figure 32 we see the same plot with other shapes. Here we can see that if we change the elongation of an object, it also affects at its compactness: an ellipse is more elongated than a circle and it is also less compact (the same happens with the square and the rectangles). Then, the compactness and the ratio of principal axes have some correlation.

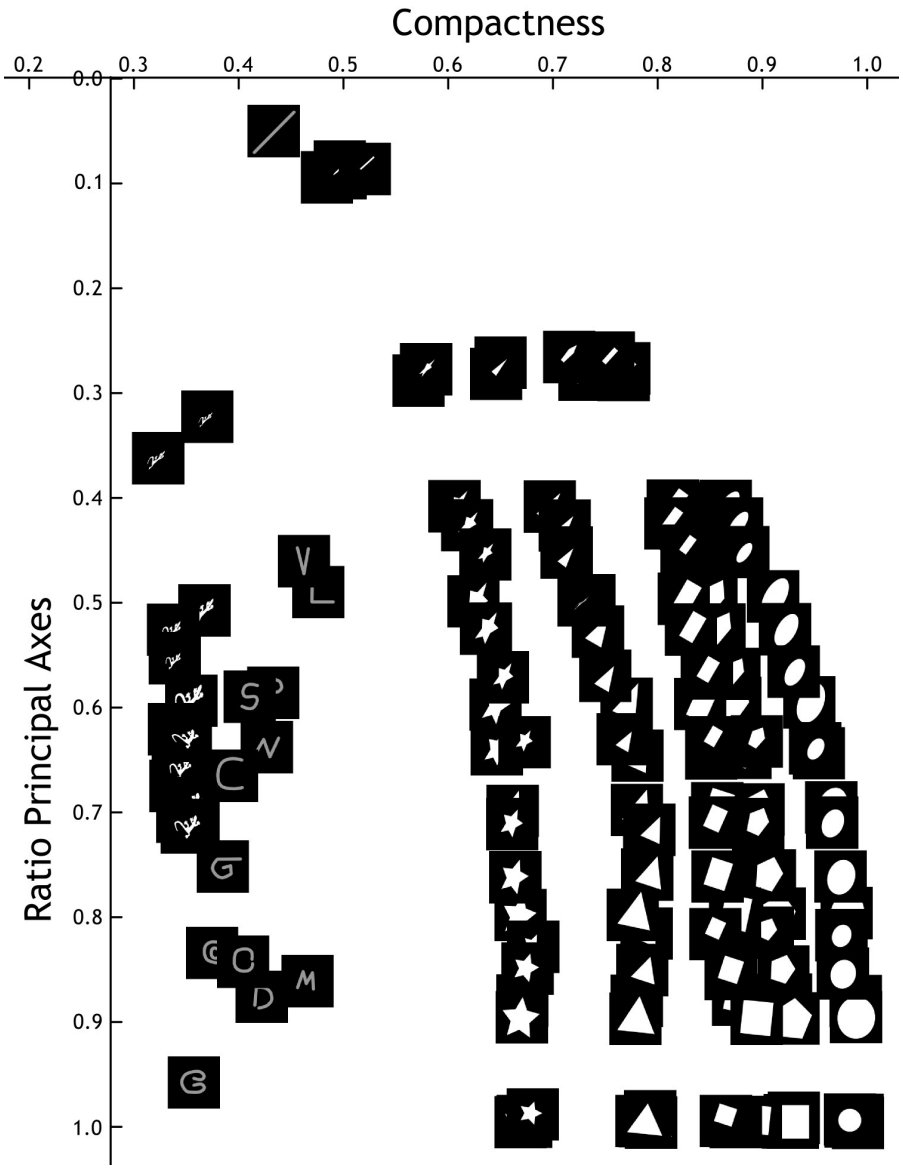


Figure 32: The Compactness and the Ratio of principal axes have some correlation: an elongated shape has always low compactness.

If we want to remove this correlation, we can define an "elliptical compactness" as the relation between the area an the perimeter that an ellipse with the same principal axes would have.

According to the Ramanujan's approximation [Ramanujan 1914], the perimeter of an ellipse can be approximated by:

$$Ellipse\,Perimeter \approx \Pi\left(3\left(r_1+r_2\right)-\sqrt{\left(3r_1+r_2\right)\left(r_1+3r_2\right)}\right) \quad (16)$$

and the area of the ellipse is

$$Ellipse\,Area = \Pi r_1 r_2 \quad (17)$$

Then, if we want the elliptical compactness of an ellipse to be 1, we define it as:

$$Ellip.Compact = \frac{\Pi(3(r_1+r_2)-\sqrt{(3r_1+r_2)(r_1+3r_2)})}{\sqrt{\Pi r_1 r_2}} \frac{\sqrt{Area}}{Perimeter} \quad (18)$$

In figure 33 we can see the same dataset as in figure 32 but using the elliptical compactness. Now the similar shapes are organized in vertical lines according to their shape and the elliptical compactness can be used independently from the ratio of principal axes to separate the shapes: in this case, ellipses have elliptical compactness around 1.0, pentagons 0.95, rectangles 0.9, triangles 0.8, stars 0.7 and the gecko 0.4.

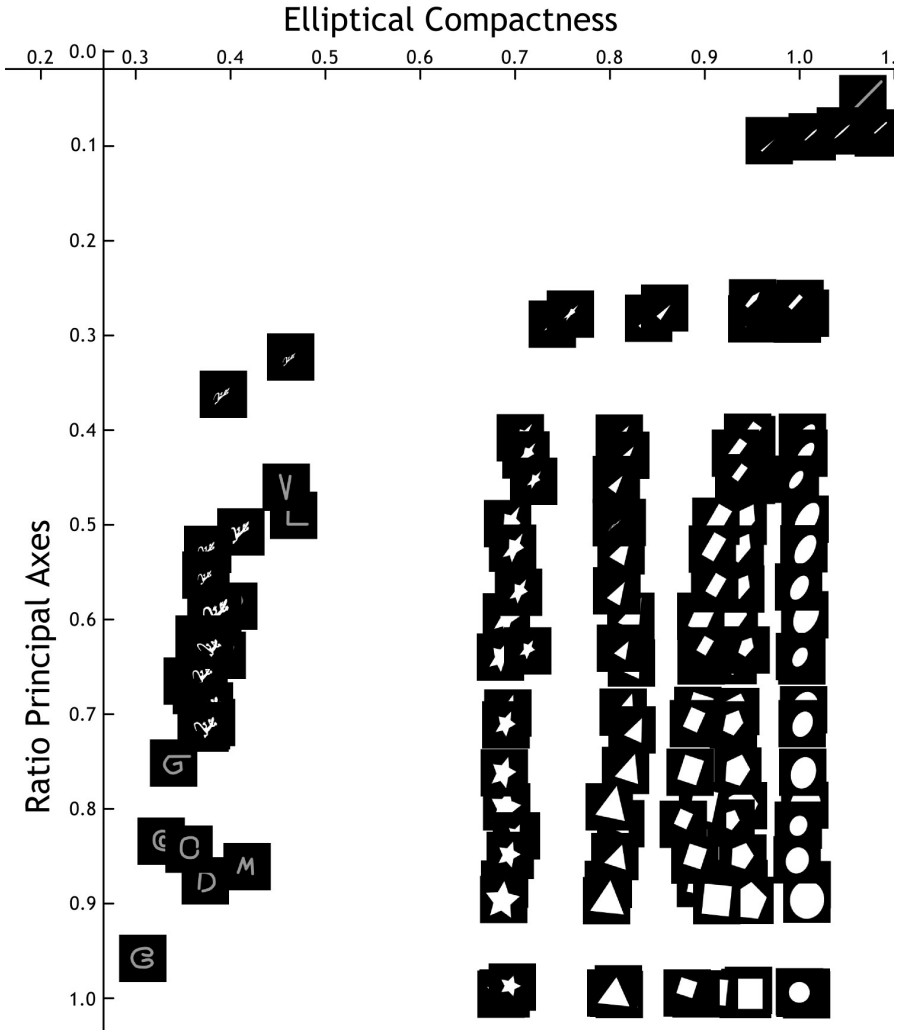


Figure 33: The Elliptical Compactness is less correlated to the ratio of principal axes than the (circular) Compactness

Looking at the two datasets that we have analyzed, we can see that it is possible to construct a simple shape classifier using the ratio of principal axes and the compactness (or elliptical compactness). In figure 34 there is a diagram of this classifier. We can see that "line" objects that may correspond to pens or pencils have a ratio of principal axes smaller than 0.15. On the other hand regular polygons have a ratio of principal axes greater than 0.8 and compactness greater than 0.9. The circles are at the extreme in this corner (both values very close to 1), but it is easy to confuse a square with a circle with a noisy contour (the noise increases the perimeter and thus reduces the compactness).

The majority of the shapes fall in the category of "deformable objects". In figure 31 we can observe

that even if many similar objects are grouped together in the graph, other perceptually similar objects may be very distant in the graph. For example, we can find C-shaped objects (like a horseshoe or a snake) mostly everywhere in this graph because even if the shape is similar, a C may be "rounder" or "longer" (which entails a different ratio of principal axes) and thicker or thinner (which affects its compactness). If we need a finer subdivision we should use more descriptors such as *convexity*, *circular variance* and *elliptic variance.*

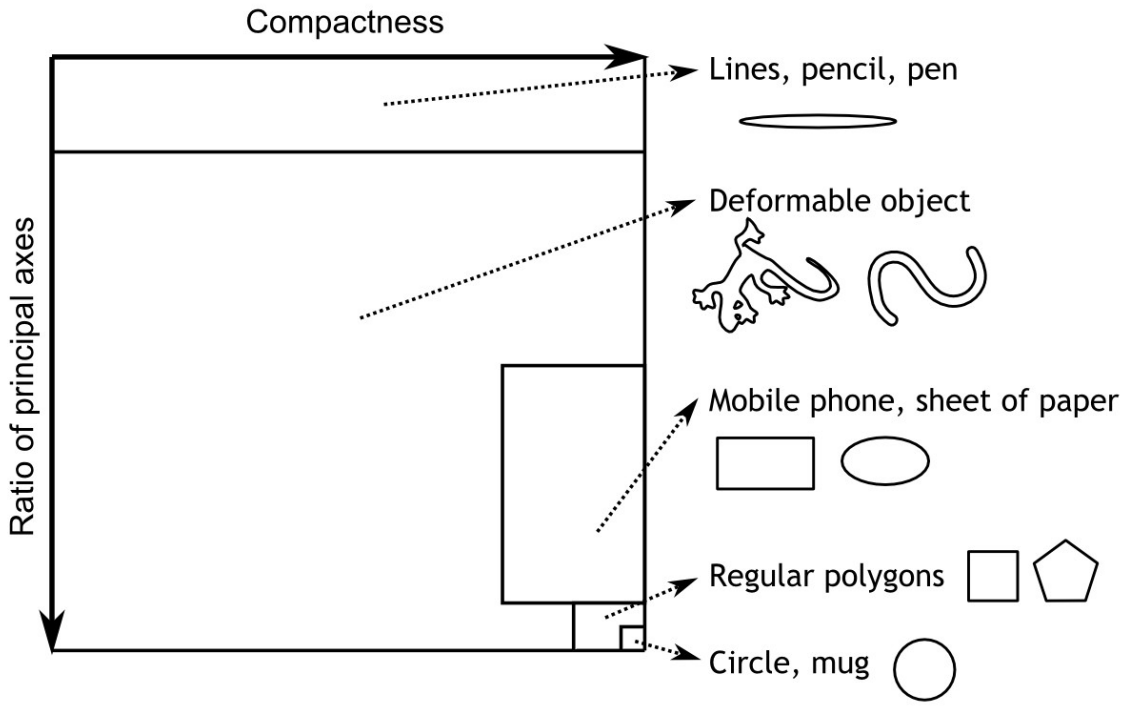


Figure 34: Shape classification using two simple descriptors: compactness and ratio of principal axes. More descriptors or combinations of them can be used to refine it.

5.3.Simple descriptors for control

With these simple global descriptors we have an approximation to the information that the fiducials give us: position, angle and recognition. Unfortunately, the angle of the principal axis is accurate only in elongated shapes (see for example the squares in figure 35) and we can know only the direction, not the sense. The pixel at the maximum distance from the centroid can provide an accurate angle in many cases but may have several possible positions in objects with symmetry. In order to have a more stable angle of orientation, we should use a combination of the two angles, track them along time and maybe add some knowledge about the shape. For example, if we know that the shape is a square, we have four possible vectors of maximum direction (one for every vertex) and tracking them we can deduce a stable angle of orientation.

At the same time there is not a unique recognition, but only a rough classification. Nevertheless, for some applications with a low number of objects or where the recognition is not crucial, this information is enough and we can avoid the the use of fiducials. For other applications where some objects need an accurate description, a rough initial classification is important to select which are the objects of interest. For example, using the classification in figure 34 we can apply the polygon simplification to the regular polygons and the skeleton analysis to the deformable objects.

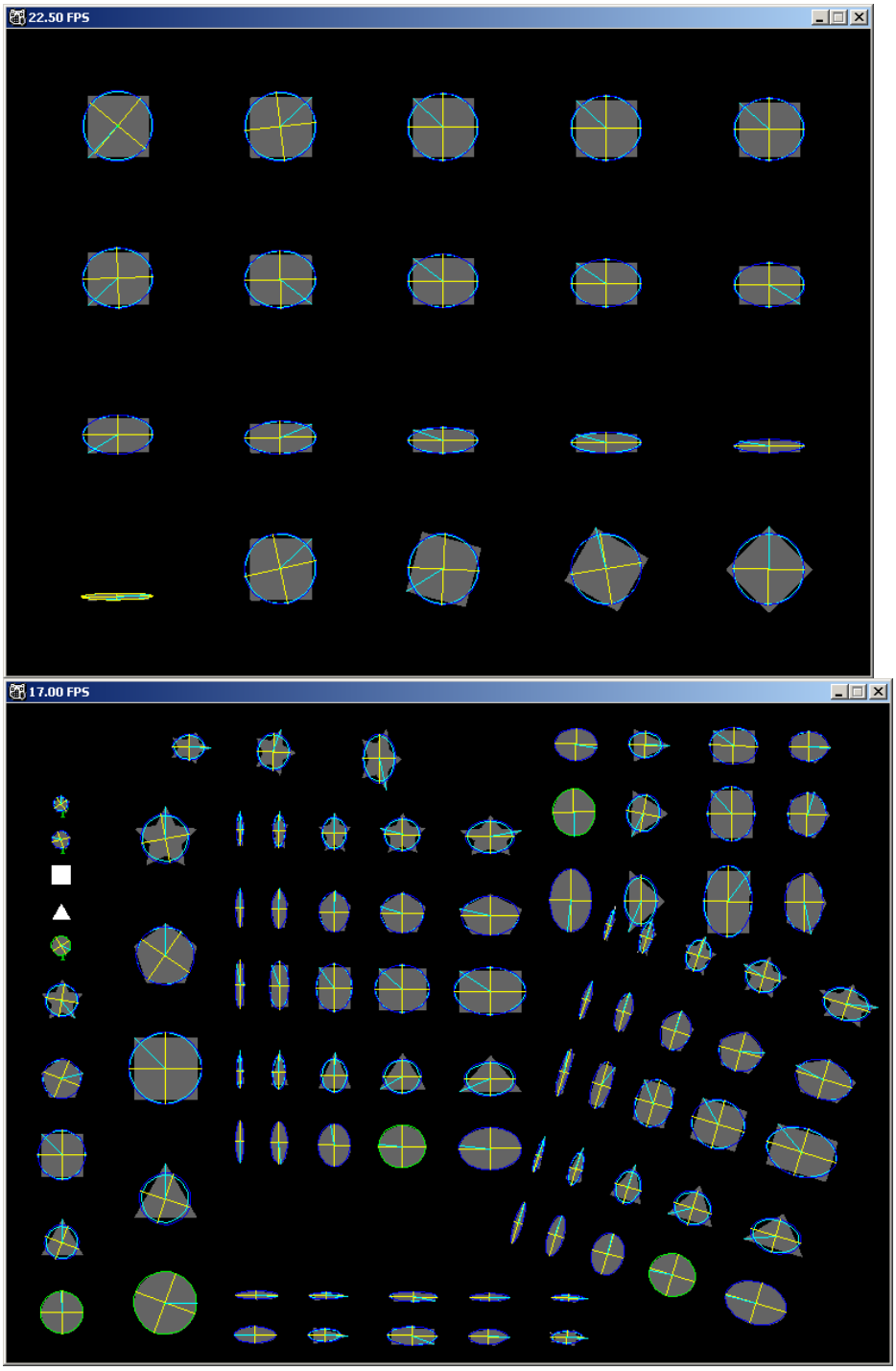


Figure 35: The principal axis can be used to find the orientation of a rectangle, but they are not stable for the squares (see last row). In this case, the maximum direction is more meaningful.

5.4.Polygon simplification

In figure 36 there is an image with some simple shapes approximated by polygons. If the value of the relevance measure is set too low, many non significant vertices appear, and if it is too high, the shape of the objects changes considerably (the stars become pentagons and the circles become

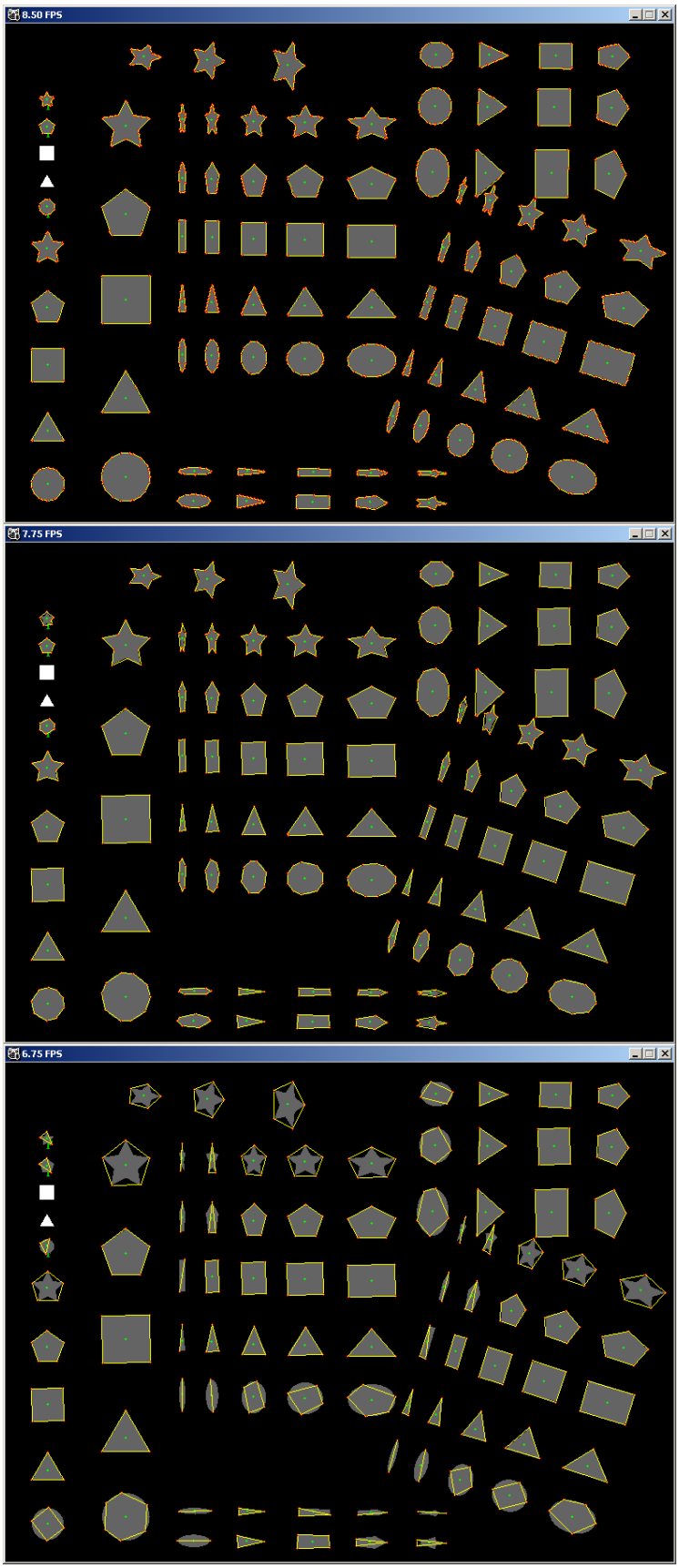


Figure 36: Simplifying polygons with different values for the relevance measure K: 0.73, 2.20 and 9.41

squares). In any case, the simplification works as desired, conserving the vertices of the polygons and distributing well the vertices in round objects. Moreover, a medium value for K (near 2.2) is a good choice for all the shapes and sizes at the same time.

In figure 37 we can see another example with some letters. In the current version, the interior holes are not traced even if it would be easy to implement.

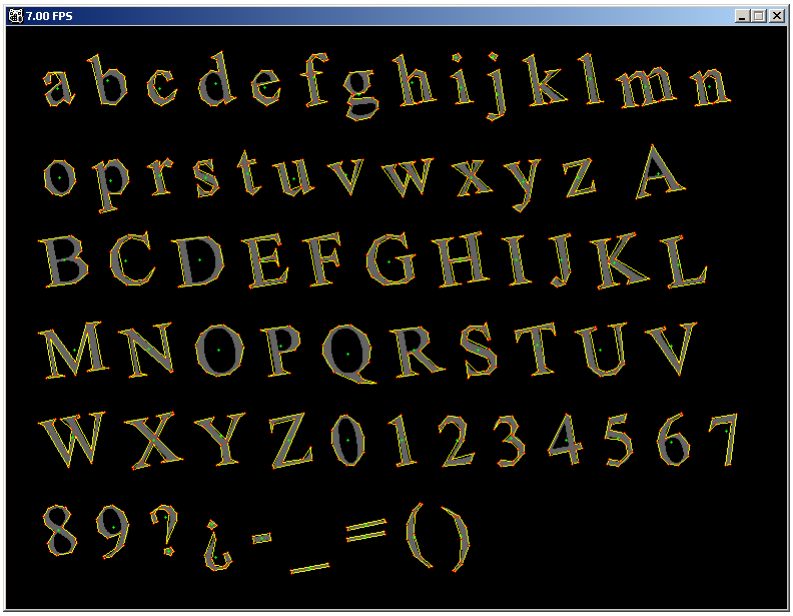


Figure 37: Polygon simplification of Times New Roman characters rotated 10 degrees (K=2.20)

5.5.Skeleton using the Image Foresting Transform

The current skeletonization method uses the Image Foresting Transform to compute the Euclidean Distance Transform and it uses the contour length associated to every skeleton pixel to prune non-significant branches (just as described in section 4.6). In this case we used an Open Source (GPL license) implementation[12] by Alexandre Xavier Falcão which is one of the authors of the IFT algorithm [Falcao et al. 2004]. Unfortunately, the contour extraction available in this library fails in some cases as mentioned in section 5.1, and we substituted it by our implementation.

An interesting feature of the IFT is that it is possible not to evaluate all the points in the image by setting their cost to infinity at some point. In our case, we are interested only in the skeleton inside the shape and we do not use the EDT of the external points. By setting the cost of the external pixels to infinity, we avoid their computation. This improvement is important in thin objects because the number of pixels inside the shape is much smaller than the number of pixels outside.

It important to choose if we allow holes in the shape. We know that a hole in the shape (even if it is only a pixel) will cause a cycle in the skeleton, changing it completely (see for example the circles in figure 18). In order to avoid cycles in the skeleton caused by the noise, we should remove them in a preprocessing step. In our case, we opted to disallow completely holes in the shapes (and cycles in the skeleton) because the texture of many objects (or things printed on them) would result in non-existent holes in the shape. To implement it with the IFT, we have to add only the external contour to the initial seeds and allow the propagation of the EDT in the internal holes.

In figure 38 there is a complete example of the computation of the Medial Axis Transform. The cost

[12]The implementation of the Image Foresting Transform by A. X. Falcão is available at http://www.ic.unicamp.br/~afalcao/downloads.html

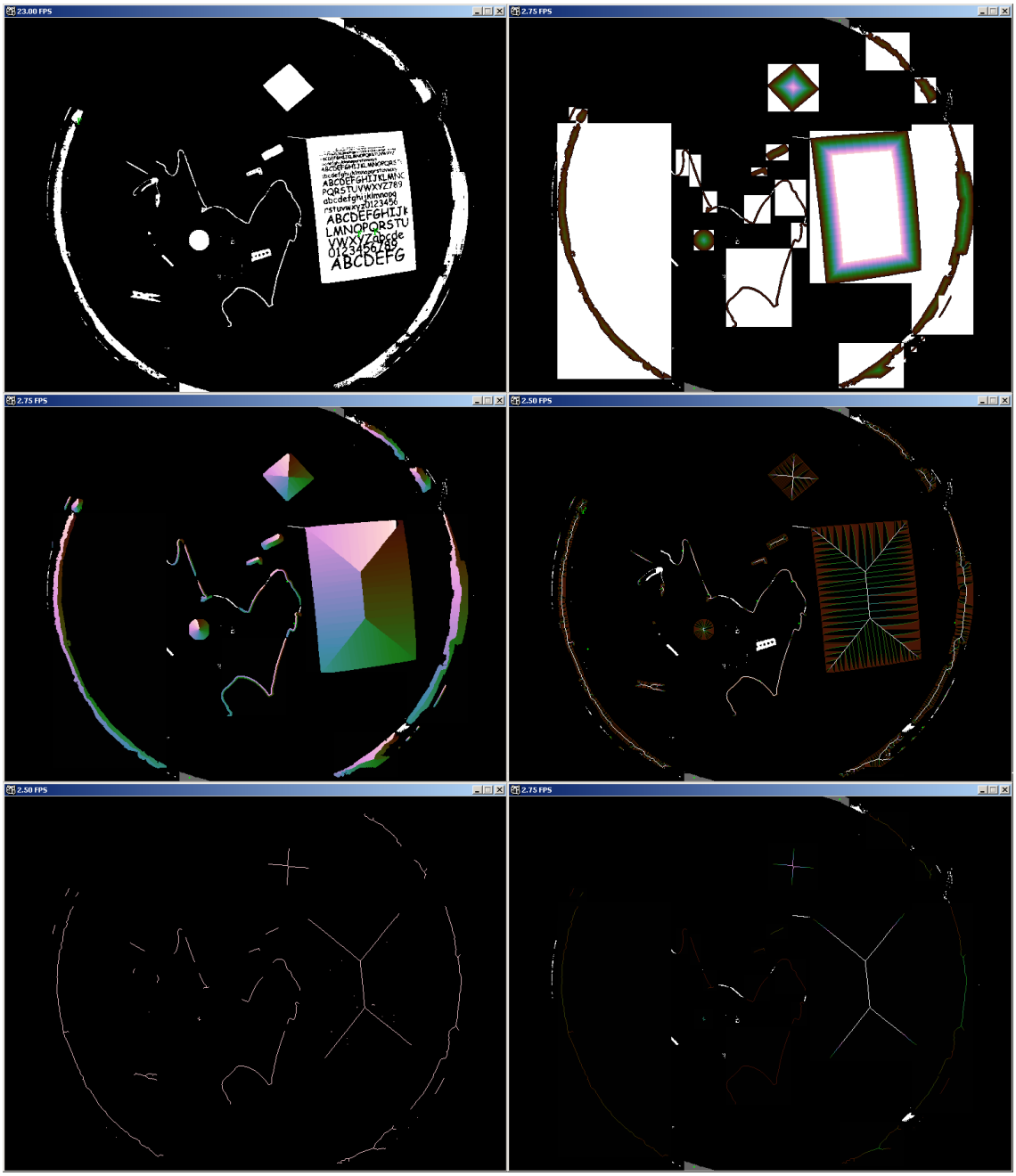


Figure 38: MAT computation: A: Original image, B: Euclidean Distance Transform, C: Voronoi Regions of contour pixels, D: Skeleton weighted by the perimeter associated to every point, E: thresholded Skeleton, F: Medial Axis Transform

function of the IFT (B) corresponds to the Euclidean Distance Transform and its predecessors image (C) corresponds to the Voronoi regions of the contour points. The divisions between Voronoi regions correspond to the skeleton, and in figure D, the color encodes how long is the perimeter associated to every skeleton point. The simplified skeleton (E) is obtained by thresholding it, and finally the Medial Axis Transform (F) combines the information of the skeleton with the Euclidean Distance Transform.

5.6.MAT simplification

The Medial Axis Transform (MAT) obtained with the previous algorithm (or any other pixel based algorithm) is composed by many pixels, because every branch is a line formed by a succession of

adjacent pixels. For example the MAT in figure 28 has 325 pixels. Since our intention is to send this information through the TUIO protocol, it is interesting to reduce the number of pixels sent just as we did when we simplified the contour pixels to a polygon.

The idea here is to simplify the pixels of every branch of the MAT approximating them to straight segments. Just as in the polygon simplification, we need to select the most significant pixels, but in this case we have the radius information that may be important in many cases (necks in the shape, for example). As we have the radius information in every point, the MAT can be seen as a set of three-dimensional lines (x, y and radius) connected in some intersection points (see figure 39).

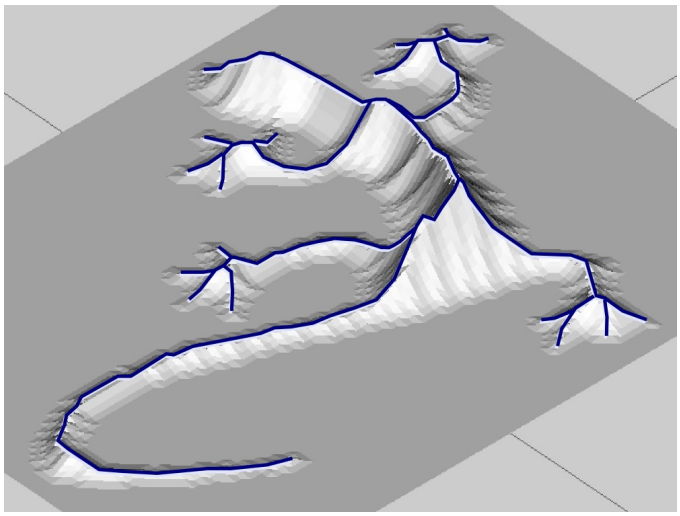


Figure 39: The Medial Axis Transform can be seen as a graph in a 3D space (x, y and radius)

To simplify this lines we can use the same approach as in the Discrete Contour Evolution [Latecki and Lakämper 1999a] that we described in section 4.3. There are two basic differences: first, we only simplify polylines instead of polygons, so the endpoints cannot be simplified, and second the segments

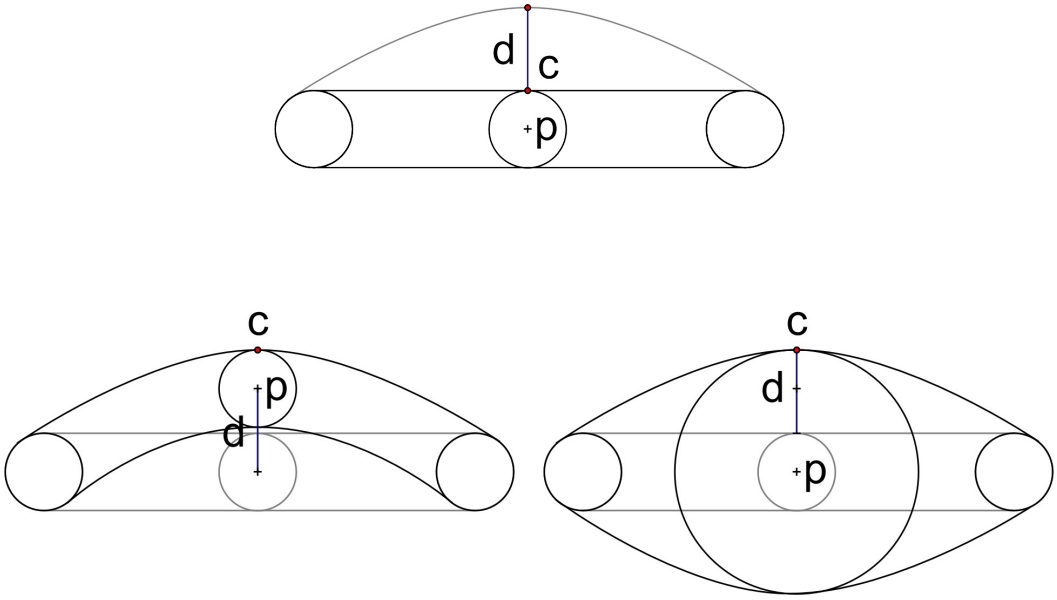


Figure 40: A displacement d of a contour pixel c may be caused by a displacement d of the skeleton point p or an increment d of the associated radius

are three-dimensional.

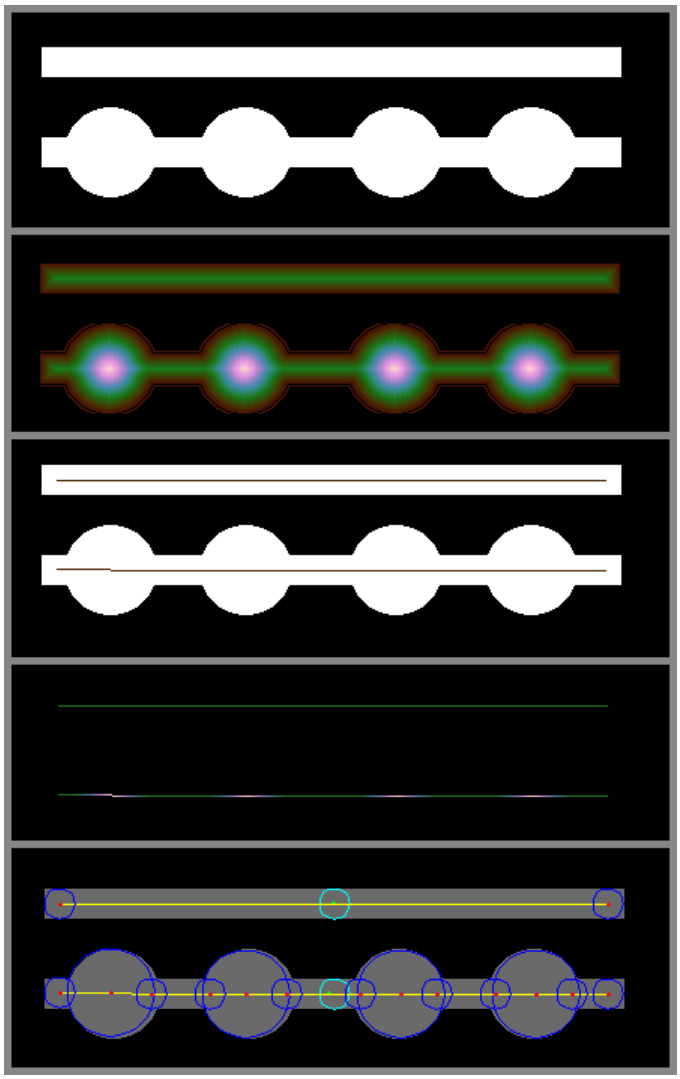


Figure 41: Computing the MAT simplification of two shapes. Original shape, Distance Transform, skeleton, MAT and MAT simplification with our generalization of the Discrete Contour Evolution algorithm to three dimensional polylines.

In this case, we compute the relevance of every point of the MAT with the same formulas (12)-(15) of the Discrete Contour Evolution, but in this case the points are three-dimensional $(x, y, radius)$. We choose to treat all three dimensions the same way because, even if the position (x and y) and the radius encode different information, both affect the contour of the shape by the same degree. In figure 40 we can see that moving a skeleton point a distance *d*, displaces a contour point the same distance as increasing the radius the same distance *d*.

In figure 41 we can see the complete process of MAT simplification. Observe that even if the two shapes are very different, their skeleton is the same (a straight line), but the MAT shows the differences between them as it includes the radius. Taking into account this radius when simplifying the MAT, the most relevant points are retained and the shape can be reconstructed.

In figure 42 we can see another example. Note that there are more points where there is more curvature and also where there are important changes in the radius of the shape.

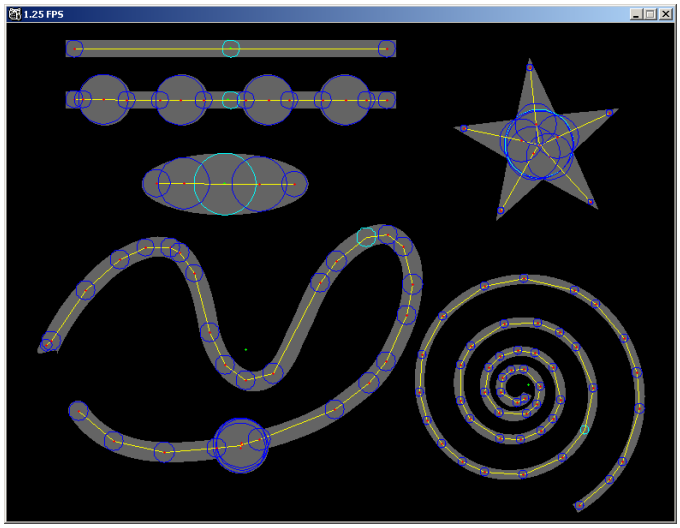


Figure 42: MAT simplification of several shapes.

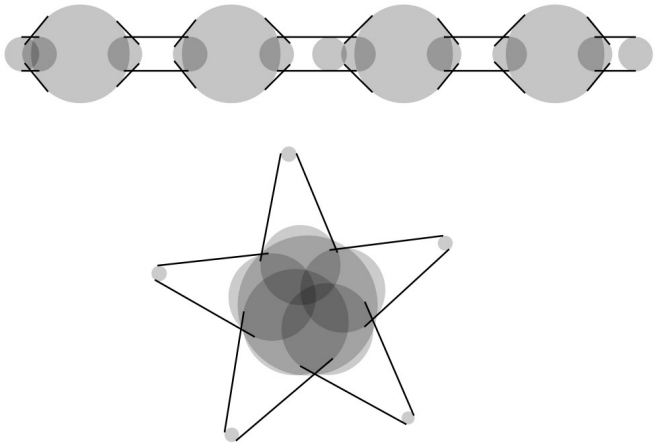


Figure 43: An approximation of the original shape can be reconstructed by drawing the tangent lines of consecutive circles in the MAT simplification. These objects are from figure 42

In figure 44 we can see the effect of the two simplifications we use. The skeleton simplification uses the perimeter associated to every skeleton point and it shortens and removes branches. On the other hand, the MAT simplification uses the position and radius of the points to simplify the branches to straight lines. Both simplifications affect the final result and they have to be set accordingly to achieve a uniform simplification. For example, in the last image of the first row the tail is too much simplified and other parts have too many points.

Finally the original shape can be reconstructed by drawing the tangent lines of every consecutive pair of MAT circles as shown in figure 43.

5.7.Polygon vs MAT

Both the polygon and the MAT simplifications discussed above are compact representations of the shape that can be used to reconstruct it.

The polygon approximation allows a straightforward representation of the object, but the MAT approximation needs more computation as it needs to compute the tangent lines of the circles.

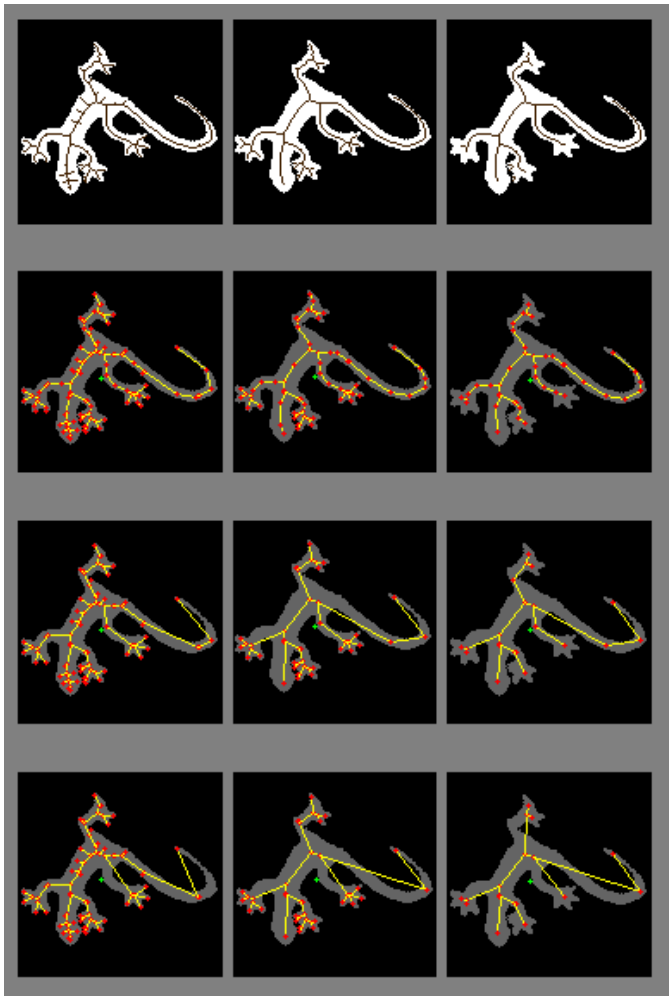


Figure 44: Skeleton simplification (first row) and MAT simplification (other rows). Each column has a skeleton simplification threshold and each row has a different MAT simplification threshold. The skeleton simplification removes branches (according to the perimeter associated to every point) and the MAT simplification converts them to straight lines. Both thresholds affect the final result.

Some shapes can be compressed better using one of the two approaches. For example, a circle needs many points in the polygon simplification, but it is exactly reconstructed with one point of the MAT simplification. Linear objects will need twice as many points in the polygon simplification than in the MAT simplification, but remember that every point of the MAT has tree dimensions instead of two. Regular polygons need one point for every vertex in the polygonal simplification and one branch in the MAT simplification.

At the same time, both approaches allow to recognize objects. Regular polygons can be recognized by counting the vertices in the polygon simplification or the branches (endpoints) in the MAT simplification. Nevertheless, the MAT simplification has more information about the structure of the object (protusions, necks). For example, most objects in figure 42 are composed by a single branch in their MAT and it is straightforward to tell that they are in fact lines, but it would be very difficult to recognize them using the polygonal representation.

Then, both representations have their strengths and complement each other. The polygon simplification is in general better for user feedback (drawings) and the MAT simplification is better for object description and recognition.

5.8.Temporal analysis

Until now we have analyzed single images, but the evolution of the different parameters along the temporal axis is as much as important in the reacTIVision system. If we want to use a parameter for control it should not change randomly because of the noise in the image.

As we are extracting many parameters from the image (many contour and skeleton points) it is difficult to track them along time to show here numerical statistics of their variance. We will present here a qualitative analysis with a rough numerical approximation of the variance of each parameter.

We will analyze sequences of two objects: the square in figure 46 and hands in figure 45.

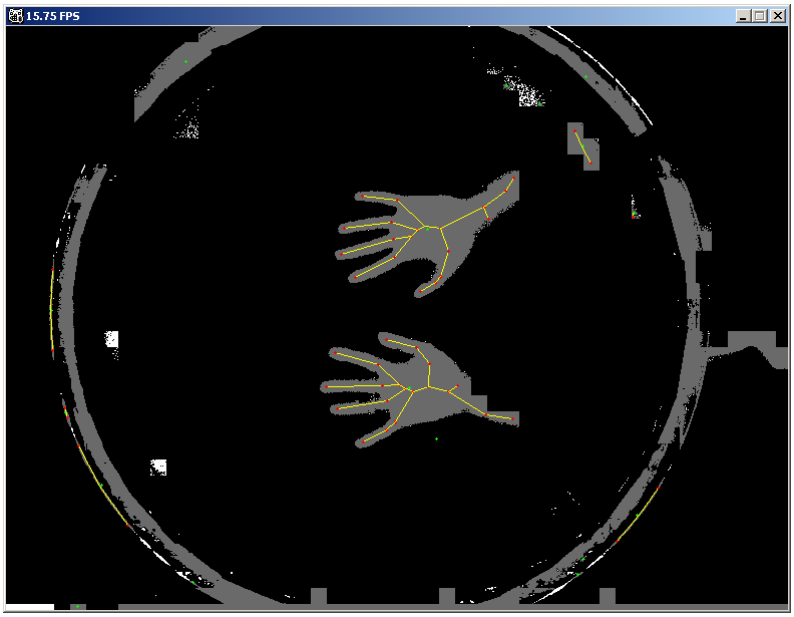


Figure 45: A frame in a sequence with hands

Because of the construction of the tabletop (with a semitransparent glass), the objects that are not completely flat (like a hand) will have blurry contours and, because of this, the binarized image has a lot of noise. Moreover the arm disappears at some point (because it is separated from the surface) and the tiled thresholder of reacTIVision introduces square blocks (the tiles) in this area. On the other hand, flat objects such as fiducials or other polygons do not have this problems.

In the case of the hands (figure 45), the centroid may change about 5 pixels from one frame to the next because of the changes in the number of blocks in the arm area. On the other hand, the angle of the principal axis is extremely stable because the elliptical approximation of the shape is quite elongated and it is not affected by these blocks in the arm. The maximum distance direction usually points to the arm and is not as much as stable as the principal axis because the arm is the noisiest part of the hand. The vertices of the polygon approximation are not very stable because in a hand there are not well defined corners, and a curve is approximated by some points and their exact position depends mostly on the noise. The endpoints of the skeleton (the fingers of the hand) are very accurate and change one pixel or none from one frame to the next. Other points of the skeleton change positions and appear and disappear randomly (similarly as points in the polygon approximation). The crossing points of the skeleton (the palm of the hand) are also quite accurate. The crossing caused by the thumb, as the thumb is big, it is not affected by the noise, but the crossings of other fingers may change two pixels from one frame to the next. Finally, there are many branches that appear in the zone of the arm due to the noise and they should be pruned in a post processing step.

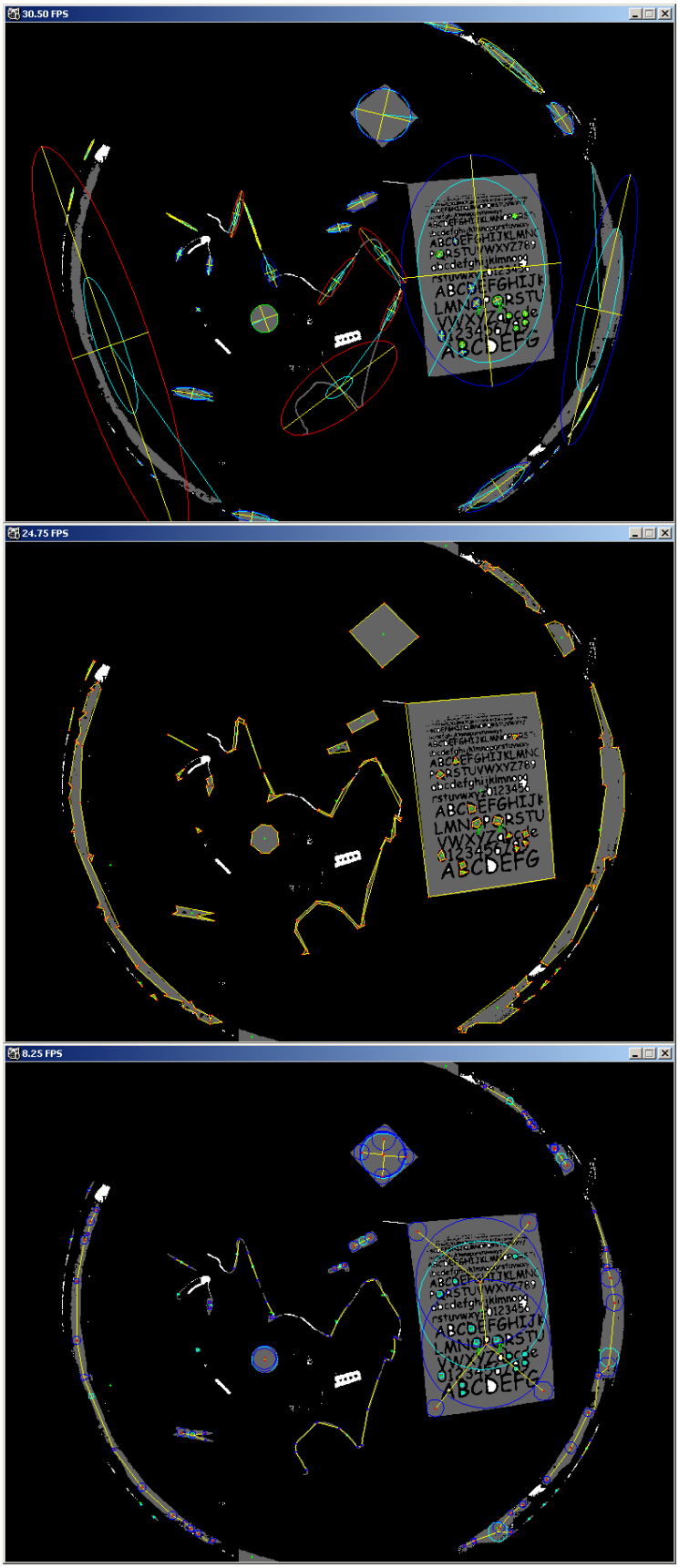


Figure 46: Information obtained with all the described algorithms: Simple descriptors (principal axes, maximum direction, area and compactness), Polygon simplification and Medial Axis Transform simplification

The square in the sequence of figure 46 has a very stable centroid that does not change its position. On the other hand the principal axis is not very stable (changing 2 pixels) because the shape is not elongated and the direction of the maximum distance changes randomly the position to any of the vertices. The vertices in the polygonal approximation and the endpoints of the skeleton are very accurate and displace only one pixel between one frame to the next.

In conclusion, the most stable parameters are the centroid, the pricipal axis (for elongated shapes), the endpoints of the skeleton (for all the shapes), the crossings of the skeleton caused by big parts of the object (the thumb of the hand), and the vertices of the polygon approximation if the object has some well defined corners. The other parameters (like the polygon approximation of round objects) may change randomly and should be used mostly for user feedback and not for an accurate control.

5.9.Computation speed

In table 3 we can see the computation speed of each of the algorithms. We can see that all of them except the MAT simplification can be computed at more than 25 fps and the simplification of the MAT can be computed at 8.5 fps. These results include the computation of all the objects in the image, which in many cases is not necessary.

As we can classify roughly the objects based on their area and simple descriptors (as discussed in section 5.2), we can select a few objects to compute their MAT simplification. Since we can compute more that 130 medium sized objects per second (the gecko image), we can compute for example the skeleton of four objects and the overall computation will be faster than 30 fps.

Other improvements or simplifications are possible, such as resizing the image as a preprocessing step. For example if we reduce the width and the height to half its size, we can compute the skeleton of all the objects in the image at more than 30 fps.

Algorithm	reactable image (780x582)	Gecko image (116x90)
Fiducials and fingers	39.5	>800
Simple descriptors	29.0	>500
Polygon simplification	25.0	320
Simplified MAT	8.5	137

Table 3: Computation speed (in frames per second) of each of the algorithms described using a Pentium IV M 2GHs under Windows XP. For the qualitative results of the reatable image see figure 46.

5.10.TUIO protocol

For completeness we will describe how these parameters can be sent to an application in order to control it.

The TUIO [Kaltenbrunner et al. 2005] protocol is based on Open Sound Control (OSC) [Wright et al. 2003]. OSC is targeted to musical control and TUIO is targeted to Tangible User Interfaces (TUI) and tabletops in particular. TUIO is used between reacTIVision and the applications that listen to its events.

These are the message types in the TUIO protocol:

```
/tuio/[profileName] set sessionID [parameterList]
/tuio/[profileName] alive [list of active sessionIDs]
/tuio/[profileName] fseq int32
```

`fseq` messages maintain a frame sequence number, `alive` messages return a list of current objects in the image and `set` messages include parameters for a particular object. The profileName indicates the type of information of object is sent. For example, the information about fiducials is sent using the `2Dobj` profile and the information for cursors (fingers) is sent using the `2Dcur` profile. The parameters in these cases include x, y, angle, and speeds and accelerations.

In order to send the information obtained with the new algorithms discussed here, we should extend the messages. Both skeleton and polygon simplification need a variable number of parameters and should be implemented in new profiles. On the other hand, the simple shape descriptors have a fixed number of parameters that are similar to the fiducials (position and angles) and the `2Dobj` profile could be used to maintain compatibility with existing applications. Nevertheless, a new profile is also needed if we want to send information about axes, maximum distance, area and perimeter (the compactness and elongation can be computed in the client side).

The proposed messages are as follows:

```
/tuio/2Ddesc set sessionID x y angleAxis1 lengthAxis1 lengthAxis2 angleMaximum lengthMaximum area perimeter
/tuio/2Dshape set sessionID x1 y1 x2 y2 ... xn yn
/tuio/2Dskel set sessionID B x11 y11 r11 x12 y12 r12 ... x1n y1n r1n B x21 y21 r21 x22 y22 r22 ... x2n y2n r2n B x31 y31 ...
```

The `2Ddesc` profile includes the information about the simple descriptors, the `2Dshape` profile includes the list of positions of the vertices in the polygon approximation and the `2Dskel` profile is a list of branches (separated by the B character) each of them containing a list of points with position and radius. In the case of the skeleton, the client could reconstruct the actual tree of branches comparing the positions (x, y) of the endpoints of all the branches (this branches should be ordered from the root to the leaves to reduce the possible comparisons).

5.11.Ideas for future work

There are many possible improvements in the algorithms presented in this research. Here we present a list of ideas:

- We could track the different points in the contour and the skeleton along time, labeling each of them with a unique ID to make it possible to treat different parts of the objects independently. For example, we could track the fingers of a hand independently and control different parameters of the application with each of them. With this point tracking, it would be possible to analyze numerically the temporal variance of the parameters as a complement to the results presented in section 5.8.

- Here we have used the Discrete Contour Evolution for polygon simplification (two dimensional vertices) and then we have extended it to MAT simplification (three dimensional vertices). If we track the different parameters along time, we can use another version of the DCE to detect gestures, for example.

- In order to speed up the computation of the skeleton, it is possible to implement the IFT in parallel architectures like FPGA with reported speedups of 5600 upon the software implementation [Cappabianco et al. 2007], but this hardware change may not be possible in all the systems that use reacTIVision.

- Other algorithms to compute the Distance Transform may be faster than the IFT and they can be also parallelized. Good candidates are the algorithms presented in [Meijster et al. 2002] and [Maurer et al. 2003].

- In order to provide a pleasant user feedback, the polygon could be converted to a Bézier curve (b-spline) smoothing some of its vertices. The reconstruction of the shape from the MAT simplification as

shown in figure 43 should be implemented as well at the client side.

- Using the MAT simplification, we can implement some of the ideas present in *Shock graphs* [Sebastian et al. 2004] (see section 4.5) for object recognition.

6.Conclusion

We have reviewed many shape descriptors and we have implemented three different techniques that look at the shape from different points of view and complement each other.

First, we used some simple global descriptors that allow us to extract useful parameters for control (position and orientation), that can also be used for user feedback (elliptical approximation of the object) and moreover they help us to classify the objects in simple categories.

Second, from the structural contour-based techniques, we implemented a polygonal approximation of the contour that is a compact representation of the shape that can be used for polygon recognition and user feedback.

Finally, from the structural region-based techniques, we proposed and implemented a simplification of the Medial Axis Transform (skeleton) that is better suited for deformable objects.

The computation of the simple descriptors and the polygon approximation is fast and performs in real-time analyzing all the objects of the image. The skeleton approximation is slower and for real-time interaction we should process only a few objects or reduce the image resolution as a preprocessing step.

References

M. Alexa, D. Cohen-Or and D. Levin (2000), "As-rigid-as-possible shape interpolation," *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 157-164, 2000.

B.P. Amavasai, F. Caparrelli, A. Selvan, M. Boissenin, JR Travis and S. Meikle (2005), "Machine vision methods for autonomous micro-robotic systems," *Kybernetes*, pp. 1421-1439, 2005.

I. Bartolini, P. Ciaccia and M. Patella (2005), "WARP: accurate retrieval of shapes using phase of Fourier descriptors and time warping distance," *Transactions on Pattern Analysis and Machine Intelligence*, pp. 142-145, 2005.

R. Bencina and M. Kaltenbrunner (2005), "The Design and Evolution of Fiducials for the reacTIVision System," *Proceedings of the 3rd International Conference on Generative Systems in the Electronic Arts* (3rd Iteration 2005), Melbourne (Australia), 2005.

R. Bencina, M. Kaltenbrunner and S. Jorda (2005), "Improved Topological Fiducial Tracking in the reacTIVision System," *Proc. of the IEEE Int. Workshop on Projector-Camera Systems* (PROCAMS 2005), 2005.

H. Benko, A.D. Wilson and P. Baudisch (2006), "Precise selection techniques for multi-touch screens," *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 1263-1272, 2006.

M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf (2000), "Computational Geometry: Algorithms and Applications", *Springer*, 2000.

M. Billinghurst, H. Kato and I. Poupyrev (2001), "The MagicBook-moving seamlessly between reality and virtuality," *Computer Graphics and Applications, IEEE*, pp. 6-8, 2001.

H. Blum (1967), "A transformation for extracting new descriptors of shape," in *Models for the Perception of Speech and Visual Form* (W. Walthen-Dunn, Ed.), pp. 362–380, MIT Press, Boston, 1967.

G. Bradski (2000), "The OpenCV Library," *Dr. Dobb's Journal* November 2000, Computer Security, 2000.

F.A.M. Cappabianco, G. Araujo and A.X. Falcao (2007), "The Image Forest Transform Architecture", *International Conference on Field-Programmable Technology*, ICFPT, pp.137-144, 2007.

J.H. Dangaria, S. Yang and P.J. Butler (2007), "Improved nanometer-scale particle tracking in optical microscopy using microfabricated fiduciary posts," *BioTechniques*, pp. 437-440, 2007.

E. W. Dijkstra (1959) "A note on two problems in connexion with graphs". *Numerische Mathematik*, 1:269-271, 1959.

R. Fabbri, L.D.F. Costa, J.C. Torelli and O.M. Bruno (2008), "2D Euclidean distance transform algorithms: A comparative survey," *ACM Computing Surveys*, vol 40 Art 2, 2008.

A.X. Falcao, L. Fontaura Costa, B.S. Cunha (2002), "Multiscale skeletons by image foresting transform and its application to neuromorphometry," *Pattern Recognition*, pp. 1571-1582, 2002.

A.X. Falcao, J. Stolfi and R.A. Lotufo (2004), "The image foresting transform: theory, algorithms, and applications", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 19-29, 2004.

P.F. Felzenszwalb (2005), "Representation and detection of deformable shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 208-220, 2005.

M. Fiala (2005), "ARTag, a Fiducial Marker System Using Digital Techniques," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

K. Finkenzeller (2003), "RFID Handbook: Fundamentals and Applications," in *Contactless Smart Cards and Identification*," 2003.

G.D. Hager and K. Toyama (1998), "X Vision: A portable substrate for real-time vision applications," *Computer Vision Image Understanding*, pp. 23-37, 1998.

J.Y. Han (2005), "Low-cost multi-touch sensing through frustrated total internal reflection," *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pp. 115-118, 2005.

P. Hutterer and B.H. Thomas (2007), "Groupware support in the windowing system," *Proceedings of the eight Australasian conference on User interface,* Volume 64, pp. 39-46, 2007.

S. Jordà, M. Kaltenbrunner, G. Geiger and R. Bencina (2005), "The reacTable*," *Proceedings of the International Computer Music Conference (ICMC 2005)*, Barcelona, Spain, 2005.

S. Jordà, G. Geiger, M. Alonso and M. Kaltenbrunner (2007), "The reacTable: Exploring the Synergy between Live Music Performance and Tabletop Tangible Interfaces," Proceedings of the 1st international conference on Tangible and Embedded Interaction, Baton Rouge, Lousiana, 2007.

S. Jordà (2008), "On stage: the Reactable and other Musical Tangibles go real," *International Journal of Arts and Technology*, in print, 2008.

M. Kaltenbrunner and R. Bencina (2007), "reacTIVision: a computer-vision framework for table-based tangible interaction," *Proceedings of the 1st international conference on Tangible and embedded interaction*, pp. 69-74, 2007.

M. Kaltenbrunner, T. Bovermann, R. Bencina and E. Costanza (2005), "TUIO: A protocol for table-top tangible user interfaces," *Proc. of the The 6th Int'l Workshop on Gesture in Human-Computer Interaction and Simulation*, 2005.

H. Kato and M. Billinghurst (1999), "Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System," *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, pp. 85-94, 1999.

R. Klette and A. Rosenfeld (2004), "Digital Geometry, Geometric Methods for Digital Picture Analysis, series in computer graphics and geometric modeling," *Morgan Kaufmann*, 2004.

L.J. Latecki and R. Lakämper (1999a), "Polygon Evolution by Vertex Deletion," *Proceedings of the Second International Conference on Scale-Space Theories in Computer Vision*, pp. 398-409, 1999.

L.J. Latecki and R. Lakämper (1999b), "Convexity Rule for Shape Decomposition Based on Discrete Contour Evolution," *Computer Vision and Image Understanding*, pp. 441-454, 1999.

L.J. Latecki and R. Lakämper (2000), "Shape Similarity Measure Based on Correspondence of Visual Parts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1185-1190, 2000.

F. Leymarie and M. D. Levine (1992), "Simulating the grassfire transform using an active contour model," *IEEE Transactions on Pattern Recognition and Machine Intelligence* 14(1), 56 75, 1992.

S. Loncaric (1998), "A survey of shape analysis techniques," *Pattern Recognition*, pp. 983-1001, 1998.

C.R. Maurer, R. Qi and V. Raghavan (2003), "A Linear Time Algorithm for Computing Exact Euclidean Distance Transforms of Binary Images in Arbitrary Dimensions", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 265-270, 2003.

A. Meijster, J.B.T.M. Roerdink and W.H. Hesselink (2002), "A General Algorithm for Computing Distance Transforms in Linear Time," *Mathematical Morphology and its Applications to Image and Signal Processing*, pp. 331-340, 2002.

P. Morrison and J.J. Zou (2007), "Triangle refinement in a constrained Delaunay triangulation skeleton," *Pattern Recognition*, pp.2754-2765, 2007.

L. Naimark, E. Foxlin and I.S. Inc (2002), "Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker," *Proceedings. International Symposium on Mixed and Augmented Reality, 2002*. ISMAR 2002., pp. 27-36, 2002.

J. Patten, H. Ishii, J. Hines and G. Pangaro (2001), "Sensetable: a wireless object tracking platform for tangible user interfaces," *ACM Press*, 2001.

U.S. Pawar, J. Pal and K. Toyama (2006), "Multiple Mice for Computers in Education in Developing Countries," *International Conference on Information and Communication Technologies and Development*, 2006. ICTD'06, pp. 64-71, 2006.

M. Peura and J. Iivarinen (1997), "Efficiency of simple descriptors," *Aspects of Visual Form*, pp. 443-451, 1997.

W. Piekarski, B. Avery, BH Thomas and P. Malbezin (2004), "Integrated head and hand tracking for indoor and outdoor augmented reality," *Virtual Reality, 2004. Proceedings. IEEE*, pp. 11-276, 2004.

S. Ramanujan (1914), "Modular equations and approximations to PI", Quarterly Journal of Mathematics, vol 45, pp. 350-372, 1914.

T.B. Sebastian, P.N. Klein, B.B. Kimia (2004), "Recognition of shapes by editing their shock graphs", *Transactions on Pattern Analysis and Machine Intelligence*, pp. 550-571, 2004.

D. Schmaistieg, A. Fuhrmann, G. Hesina, Z. Szalavari, L.M. Encarnacao, M. Gervautz and W. Purgathofer (2002), "The Studierstube augmented reality project," *Presence: Teleoperators and Virtual Environments*, pp. 33-54, 2002.

C. Shen, F.D. Vernier, C. Forlines and M. Ringel (2004), "DiamondSpin: an extensible toolkit for around-the-table interaction," *Proceedings of the 2004 conference on Human factors in computing systems*, pp. 167-174, 2004.

J.R. Shewchunk (1996), "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," *Lecture Notes In Computer Science*, pp. 2003-222, 1996.

F.A. Smit, A. van Rhijn and R. van Liere (2007), "GraphTracker: A Topology Projection Invariant Optical Tracker," *Computers & Graphics*, pp. 26-38, 2007.

R. W. Smith (1987), "Computer processing of line images: A survey", *Pattern Recognition* 20(1), 7 15, 1987.

B. Ullmer and H. Ishii (1997), "The metaDESK: models and prototypes for tangible user interfaces," *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pp. 223-232, 1997.

P. Wellner, "Interacting with paper on the DigitalDesk (1993)," *Communications of the ACM*, pp. 87-96, 1993.

A.D. Wilson (2005), "PlayAnywhere: a compact interactive tabletop projection-vision system," *Proceedings of the 18th annual ACM symposium on User interface software and technology*, pp. 83-92, 2005.

A.D. Wilson (2006), "Robust computer vision-based detection of pinching for one and two-handed gesture input," *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pp. 255-258, 2006.

A.D. Wilson and R. Sarin (2007), "BlueTable: connecting wireless mobile devices on interactive surfaces using vision-based handshaking," *Proceedings of Graphics Interface 2007*, pp. 119-125, 2007.

M. Wright, A. Freed, A. Momeni (2003) "OpenSound Control: State of the Art 2003," *Proceedings of the 3rd Conference on New Instruments for Musical Expression* (NIME 03), Montreal, Canada, 2003.

D. Zhang and G. Lu (2001), "A Comparative Study on Shape Retrieval Using Fourier Descriptors with Different Shape Signatures," *Proceedings of the International Conference on Multimedia and Distance Education*, pp. 1-9, 2001.

D. Zhang and G. Lu (2004), "Review of shape representation and description techniques," *Pattern Recognition*, pp. 1-19, 2004.

J.J. Zou and H. Yan (2001), "Skeletonization of ribbon-like shapes based on regularity and singularity analyses," IEEE Transactions on Systems, Man, and Cybernetics, Part B, pp. 401-407, 2001.