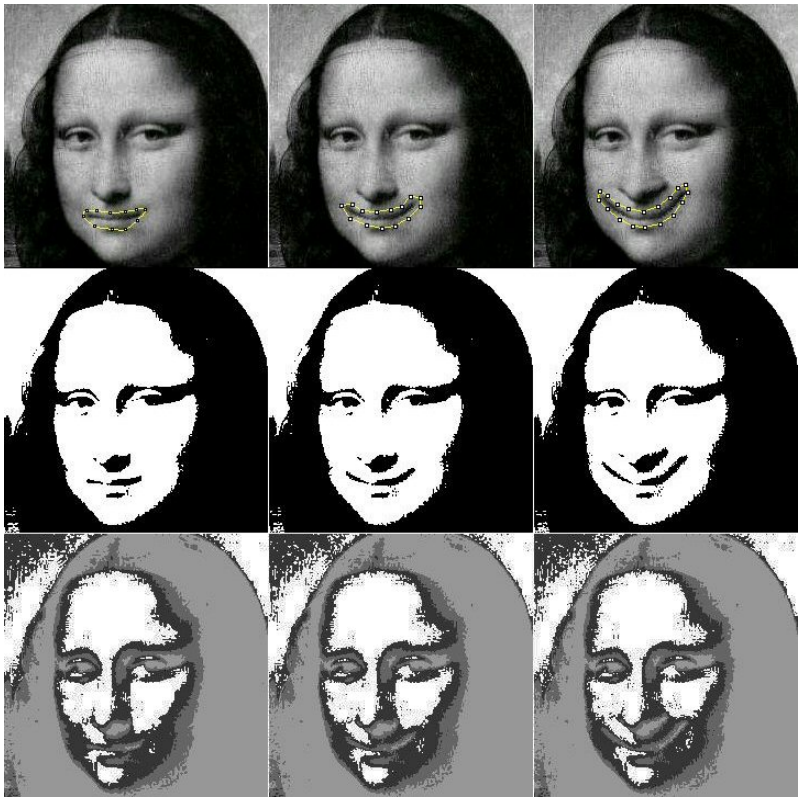
RESEARCH ASSIGNMENT

SEGMENTATION AND TRACKING OF DEFORMABLE OBJECTS IN VIDEO



Dennis Hendriksen - 9325091
d.hendriksen@student.tudelft.nl

August 11, 2005

TABLE OF CONTENTS

1 INTRODUCTION

Segmentation and tracking of objects in video is of great importance for video-based encoding, observation and retrieval. Applications are found in a wide range of applied sciences; tracking hooligans in a soccer stadium or wildlife animals, segmenting air bubbles in MRI image sequences or analyzing facial expressions.

The difference between segmentation of a single image and segmentation of a sequence of images is that the latter can benefit from temporal knowledge. Frames from the past as well as the future provide information helpful in segmenting the current fame, especially when the recording medium records exactly the same location over time. Object tracking is one way to gain useful information in this matter.

> The key question of this research assignment is: "What is the best segmentation technique for segmentation of deformable objects in video taking into account noise, (changes in) illumination and color?"

Real-world video segmentation problems are often complex. Noise, changes in illumination, occlusions, multiple look-a-like objects and cluttered backgrounds make segmentation a challenging task often requiring a task-specific approach where knowledge about the environment is known. Not to mention the fact that the segmentation often needs to be performed online, real-time at speeds averaging 30 frames a second.

This research assignment gives a survey of state-of-the-art segmentation and tracking techniques showing that often both go hand in hand. The specifics of deformable object tracking in the field of image segmentation are discussed first. The major part of this research assignment focuses on segmentation of static images, with special attention for the watershed transform and explains how they can be applied to the segmentation of image sequences. Chapter three also focuses on motion estimation and tracking of objects in image sequences discussing discrete as well as probabilistic methods. Chapter four is a comparison of all methods discussed and followed by experimental findings and conclusions.

2 DEFORMABLE OBJECT TRACKING

The best segmentation results are achieved by integrating as much knowledge about the to be segmented object in the segmentation process. Knowledge about the medium generating the image, object properties and even weather conditions such as the position of the sun etc.

As mentioned in the introduction image sequences offer additional knowledge above single images. The analysis of past and future frames gives information which can be used directly or indirectly in the segmentation and tracking process. Take for example a noisy image sequency with one ball moving randomly at a known speed through a noisy changing background. Suppose that the ball is located in the left part of an image. When during segmentation a supposed ball is detected at the right side of the image the segmentation engine knows the supposed ball is in fact not a ball.

Looking again at the ball example it is easy to see that only part of the image needs to be segmented in order to detect the ball. Tracking is then the process of updating the ball position in a frame based on new information from the current and possibly future frames.

Object properties include texture, reflection properties, deformability, speed and endless more. Here we focus on deformable objects. Take for example a mouse, which is a highly deformable object. Observing a mouse results in knowledge about its behavior, maximum speed and interaction with other objects and animals. Again all this knowledge can be integrated to reach better segmentation results. From the speed, camera position, camera angle and frame rate the maximum displacement in pixels in the image can be calculated. The mouse in frame $n+1$ is located at a maximum distance from its current point which can be used by the segmentation engine, similar to the ball example.

State of the art segmentation and tracking techniques, discussed in this research assignment, integrate in some way or another knowledge such as discussed above. As will be seen these are the techniques or combination of techniques resulting in the best segmentation and tracking of deformable objects in video.

3 STATE-OF-THE-ART SEGMENTATION TECHNIQUES

There exist hundreds and hundreds of methods to segment an image and again another hundred variants of each method improving image sequence segmentation results. These methods can be divided in threshold techniques and boundary-based (also called edge-based), region-based and motion-based methods.

Threshold techniques are based on the thresholds which are usually selected from the image histogram. Edge-based methods try to find the places of rapid transition from one to the other region of different brightness or color value. Edge detection techniques use filters such as the Sobel, Prewitt, Gaussian, Laplacian or LoG (Laplacian of Gaussian) filter. Region based segmentation methods are based on finding the regions directly, here we discuss active contours, region growing, region splitting and merging and the watershed transform with special attention to the latter. The use of multiscale approaches in these algorithms is mentioned shortly.

Motion-based segmentation work on image sequences and either segment objects directly or need combining with a segmentation technique for single images. Background subtraction, optical flow and the particle filter are discussed at the end of the chapter. Special attention is paid to the particle filter, because of its promising results.

Finally two approaches requiring training are discussed, the active appearance and active shape model.

3.1 ACTIVE CONTOURS

"Low level" feature detection processes are effective segmentation methods up to a point, but cannot be expected to retrieve entire geometric structures. Furthermore these processes are often applied to the whole image and make no use of prior information about objects. Active contours (also referred to as snakes) are curves defined in the image domain moving under influence of forces as to fit to for example object boundaries. They are applied locally and make effective use of specific prior information about objects which makes them interesting for segmentation of deformable objects.

First the basic concepts of active contours are introduced after which implementations using active rays, B-splines and level-sets are discussed.

3.1.1 Basic concepts

Active contours were first introduced in 1987 by Kass et al. [1] gaining popularity ever since. An active contour is an energy-minimizing curve guided by external forces pulling the curve towards object boundaries and internal forces discouraging bending and stretching to preserve smoothness of the contour. Active contours can be divided in parametric and geodesic (non-parametric) active contours [2]; this section focuses on the former because of its computational efficiency.

A parametric curve defined in image space is defined as

$$v(s) = (x(s), y(s)), s \in [0,1]$$

where x and y are splines. Traditional snakes minimize the energy E in

$$E = E_{int} + E_{ext} + E_{constraint}$$

The internal energy, preserving smoothness as said before, is the sum of the elastic and bending energy.

$$E_{elastic} = \frac{1}{2}\int_s \alpha(s)\left|v'(s)\right|^2 ds$$

$$E_{bending} = \frac{1}{2}\int_s \beta(s)\left|v''(s)\right|^2 ds$$

$$E_{internal} = E_{elastic} + E_{bending} = \int_s \frac{1}{2}\left(\alpha(s)\left|v'(s)\right|^2 + \beta(s)\left|v''(s)\right|^2\right) ds$$

The elastic energy discourages stretching of the active contour where $\alpha(s)$ is a weight controlling the elastic energy along different parts of the contour. Often $\alpha(s)$ is chosen to be constant. The bending energy is the sum of squared curvature (thus minimal for a circle) where $\beta(s)$ is a weight similar to $\alpha(s)$.

The external energy is derived from the image. Choose a function for the image energy, for example

$$E_{image} = -\left|\nabla I(x, y)\right|^2$$

The external energy is then defined as

$$E_{external} = \int_s E_{image} ds$$

Using variational calculus and applying the Euler-Lagrange differential equation results in the following equilibrium equation

$$(\alpha v'(s) - \beta v''(s)) - \nabla F_{external} = 0$$

where $F_{external}$ is the external force.

For application on images a discrete approximation is required. A practical and often used method is representing v by a sequence of samples spaced at intervals of length h, and approximate the first and second derivative using the finite element method

$$v'(s_i) = \frac{v(s_i) - v(s_{i-1})}{h}$$

$$v''(s_i) = \frac{v(s_{i+1}) - 2v(s_i) + v(s_{i-1})}{h^2}$$

Then solve the resulting equations for all samples, which can be done in time $O(n)$. The resulting curve representation is then a polynomial with the samples as nodes.

3.1.2 Advantages & Disadvantages

Table 1 shows advantages and disadvantages of the traditional snake when applied to segmentation of deformable objects in video.

Table 1

Advantages	Disadvantages
a) Images only processed locally	e) Problems with weak object contours and strong background edges near object
b) No training needed	f) Boundary concavities not processed into [3]
c) Real-time processing	g) No use of temporal information
d) Results in object contours	h) Small capture range: Initial contour most be close to true boundary
	i) No ordering of contour elements

Performance of an active contour in the case of e) can be increased by incorporating prior knowledge about the shape of the object as in [4], using the distance between the active contour and a contour of reference as a measure. Another way is updating the active contour in an image sequence frame by frame, which also decreases the chance of boundary concavities that need to be processed into. The latter is also prevented using an adjusted external energy function called Gradient Vector Flow described in [3]. This function is computed as a diffusion of the gradient vectors of an edge map of an image. Another advantage of Gradient Vector Flow is its increased capture range compared to the traditional snake.

3.1.3 Active ray

The contour point search of traditional snakes is performed in the 2D image plane. Active rays [5] reduce the search to a 1D search and overcome other shortcomings of traditional snakes.

The general idea behind active rays goes as follows: From a point inside the object, preferably the centre of gravity, rays are shot crossing the object boundary. The points at these intersections form the contour points, which linked together, form the object contour.

The internal energy $E_{internal}$ is defined as

$$E_{internal} = \frac{\alpha(\varphi)\left|\lambda'(\varphi)\right|^2 + \beta(\varphi)\left|\lambda''(\varphi)\right|^2}{2}$$

where $\lambda'(\varphi)$ is a gradient search along a line from the centre of gravity with angle φ.

Besides the computational speed advantage the active ray approach results in ordered contour elements opposed to the traditional snake. An often encountered phenomenom with the traditional snake is the shrinking of the contour in the case of

missing external energies [5], for example with weak object contours. The active ray approach does not suffer from this disadvantage.

The active ray approach can be extended to include a prediction step and knowledge between neighbouring contour elements. Another approach using a 1D line search is given in [6] where the line search takes place from each contour point instead of the centre of gravity (and in a probabilistic setting).

3.1.4 Active Contours using B-splines

In section 3.1.1 the discrete approximation of an active contour was represented as a polynomial. A better and also computational efficient representation of an active contour is as a B-spline [7]. The B-spline maintains a degree of smoothness, allowing a reduction in computation because fewer nodes are required.

Disadvantage of the B-spline approach is its sensitivity to noise. In [8] an approach using smoothing B-splines is proposed yielding better segmentation results in natural sequences at the same computational costs.

3.1.5 Active Contours using level-sets

More control points result in general in better active contours at the cost of performance. Unfortunately the situation gets difficult when the control points cross over themselves. Another disadvantage of parametric active contours occurs when objects split up. Active contours using level-sets don't suffer from these disadvantages.

The central idea of level-sets is representing a contour as a function representing a 3D shape (for example a cone or sphere) where a height of zero is the original contour. Then, according to the motion equation of the contour the flow of the surface is calculated and the *zero level set* is the contour propogation. This is illustrated in Fig. 1.

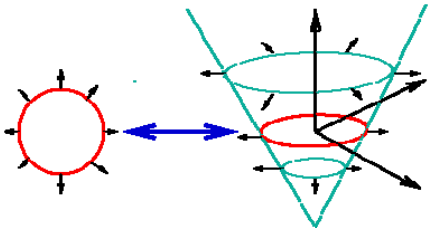


Fig. 1 Contour and level-set representation visualized

Disadvantage of level-set based active contours compared to parametric active contours is that they are computational more expensive.

3.2 ADAPTIVE THRESHOLDING

Histogram thresholding classifies a pixel as foreground pixel when $f(x, y) > T$. The threshold T can be chosen manually or automatically. Thresholding results can be

improved by dividing the image in regions and applying the thresholding to regions with a different threshold for each region. This is called basic adaptive thresholding [9].

3.3 REGION GROWING

Region growing is an easy to understand procedure. It groups pixels or subregions into larger regions based on specified criteria. The basic approach is to start from manual or automatic selected points, the seed points, and decide for each neighbor if it should join the region. Example growing criteria could be a minimum and or maximum intensity value, gradient difference or criteria based on texture. For each neighbor that does follow exactly the same approach. The region growing process stops when all neighbors have been processed, after a specified amount of iterations or based on shape criteria (for example a maximum distance from the seed point).

Region growing performs better than edge detection in noisy images and outperforms histogram methods in almost all situations.

3.4 REGION SPLITTING AND MERGING

Where region growing is a bottom-up approach, region splitting is a top-down approach. An image is subdivided in a set of arbitrary, disjoined regions and then merged and/or split according to specified criteria. Some examples of those criteria have been mentioned in the previous section.

A usual approach is storing the original image as root of a quadtree. Split the image in four rectangular equal sized regions and store these as children of the root. For each tree node decide if adjacent children are split or joined, for example based on the difference between the mean intensity values of the regions.

3.5 WATERSHED TRANSFORM

Edge detection techniques seldom result in connected edges. A post processing step such as edge linking is required to obtain closed contours corresponding to object boundaries. Region growing methods provide connected regions and often result in good segmentations, but rely heavily on the chosen seeds. The morphological approach to image processing combines the advantages of region processing and edge detection is called the watershed transform. This approach often leads to more stable segmentation results including connected edges.

This section discusses the basic concepts of the watershed transform and its application to segmentation of video. Watershed definitions are followed by an overview of the advantages and disadvantages of this segmentation technique. Finally a range of techniques and state-of-the-art algorithms are presented tackling these disadvantages.

3.5.1 Basic Concepts

Imagine a grayscale image as a mountain landscape where the intensities represent the height (Fig. 2b). A raindrop falling on the landscape can:

a) fall onto a regional minima
b) fall onto a location where it will flow with certainty to a regional minima
c) fall onto a location where it would be equally likely to flow to two or more regional minima

All points satisfying point a) and b) belong to a catchment basin. The watersheds (sometimes in literature referred to as watershed lines, dams or divide lines) consist of all points in c). The problem with this metaphore is that it is unclear what happens with raindrops falling on plateaus.

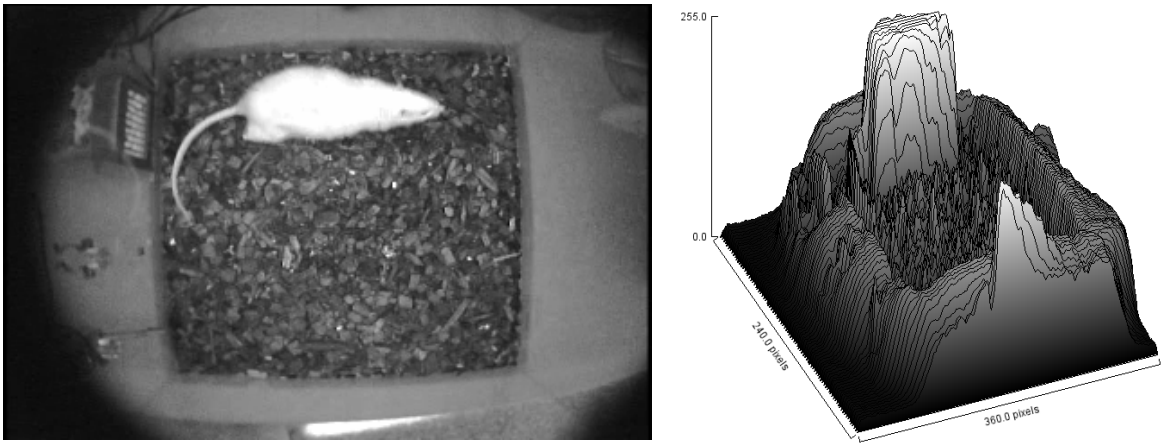


Fig. 2 [a b] An image and its representation as a topographical surface

Another approach to look at watersheds is in terms of flooding simulations. Suppose that a hole is pierced through each regional minimum and the landscape is uniformly flooded from below. Each time a catchment basin is about to flood a dam is build or its height increased. Eventually only the tops of the dams are visible above the water line. Now the dams constitute the watersheds and the lakes the catchment basins.

The objective of a watershed segmentation algorithm is to detect the watershed lines. Intuitively the watershed lines should represent the location of pixels which best separate the dark objects (regional minima), in terms of grey level difference (contrast). Thus in practice such an algorithm is applied to the (morphological) gradient of an image such that the watershed lines form object contours. Some implementations compute the watershed lines, others compute basins without touch. Classical watershed algorithms can be used to segment objects in video by applying the algorithm to each successive frame ignoring temporal information.

An extensive overview of watershed algorithms and the most common watershed definitions is given in [10].

3.5.2 Definitions

This section presents the two most common definitions of watershed transform. First watershed by simulated immersion as given by Vincent and Soille [11] followed by the definition by topographical distance presented in [12].

Geodesic influence zone

Before presenting both watershed definitions first the geodesic influence zone is introduced.

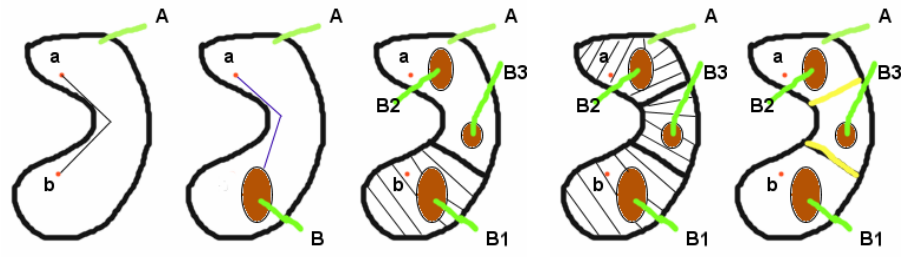


Fig. 3 [a b c d e]

The geodesic distance $d_A(a,b)$, as shown in Fig. 3a, is the minimum path length within A from a to b. The geodesic distance from a point to a subset of A (Fig. 3b) is defined as the minimum path length from a to the nearest point in B

$$d_A(a,B) = \underset{b\in B}{MIN}(d_A(a,b))$$

Let subset B of A be partitioned in multiple connected components $B = \{B_1, B_2, B_3\}$ as illustrated in Fig. 3c. The geodesic influence zone of B_1 in A is then defined as

$$\mathrm{IZ_A(B_1)} = \left\{ \mathrm{p} \in \mathrm{A} \middle| \forall j \in \left[1..k\right] \setminus \{1\} : d_A(p,B_1) < d_A(p,B_j) \right\}$$

Fig. 3c shows the geodesic influence zone of B_1 in A as the area with dashed stripes. The geodesic influence zone of B in A, shown in Fig. 3d, is then defined as the union of all influence zones of B in A

$$IZ_A(B) = \bigcup_{i=1}^{k} IZ_A(B_i)$$

The yellow lines in Fig. 3e show the skeleton of influence zone of B in A defined as

$$SKIZ_A(B) = A \setminus IZ_A(B)$$

Watershed by Immersion

Let $f : D \to \mathbb{N}$ be a digital grey value image. $h_{\min}$ and $h_{\max}$ are the minimum and maximum values in the image. T_n represents the set of coordinates below plane n defined as

$$T_n = \left\{ p \in D \middle| f(p) < n \right\}$$

The recursion to derive the watershed lines from an image is then given as

$$\begin{cases} X_{h_{\min}} = \{ p \in D \mid f(p) = h_{\min} \} = T_{h_{\min}} \\ X_{h+1} = MIN_{h+1} \cup IZ_{T_{h+1}}(X_h), h \in [h_{\min}, h_{\max}) \end{cases}$$
$$Wshed(f) = D \setminus X_{h_{\max}}$$

Watershed by topographical distance

Let again f be a digital grey value image and f^* the lower complete image of f so that each pixel which is not in a minimum has a neighbor of lower grey value. The topographical distance $T_f(p,q)$ between two points p and q is defined as the minimum of all topographical distances along all paths between p and q. The topographical distance along a path can be calculated in many ways. Here the presentation according to [12] is given. The cost of traveling from pixel p to q is defined as

$$\text{cost}(p,q) = \begin{cases} LS(p) \cdot d(p,q) \text{ if} f(p) > f(q) \\ LS(q) \cdot d(p,q) \text{ if} f(p) < f(q) \\ \frac{1}{2}(LS(p) + LS(q)) \cdot d(p,q) \text{ if} f(p) = f(q) \end{cases}$$

with the distance defined as earlier and the lower slope LS defined as

$$LS(p) = \underset{q \in N_G(p) \cup \{p\}}{MAX} \left(\frac{f(p) - f(q)}{d(p,q)} \right)$$

where $N_G(p)$ is the set of neighbors of p. For pixels whose neighbours are all of higher value, the lower slope is defined to be zero. The watershed by topographical distance is then defined as

$$CB(m_i) = \left\{ p \in D \mid \forall j \in I \setminus \{i\} : f^*(m_i) + T_{f^*}(p, m_i) < f^*(m_i) + T_{f^*}(p, m_j) \right\}$$
$$Wshed(f) = D \cap \left(\bigcup_{i \in I} CB(m_i) \right)^C$$

Immersion vs. Topographical distance

In general watershed algorithms based on immersion result execute faster than algorithms based on topographical distance. Both definitions are equivalent, but in practice implementations by immersion such as Vincent & Soille [11] do not adhere to this definition completely. In [13] it is noted that watershed by immersion does not lead to an optimal result while watershed by topographical distance does. An example is shown in figure x where the results of watershed applied to Fig. 4a are shown.

Meyer's watershed contours in Fig. 4b and Vincent & Soille's contours in Fig. 4c do not preserve the contour at altitude 20 while topographical watershed in Fig. 4d does.

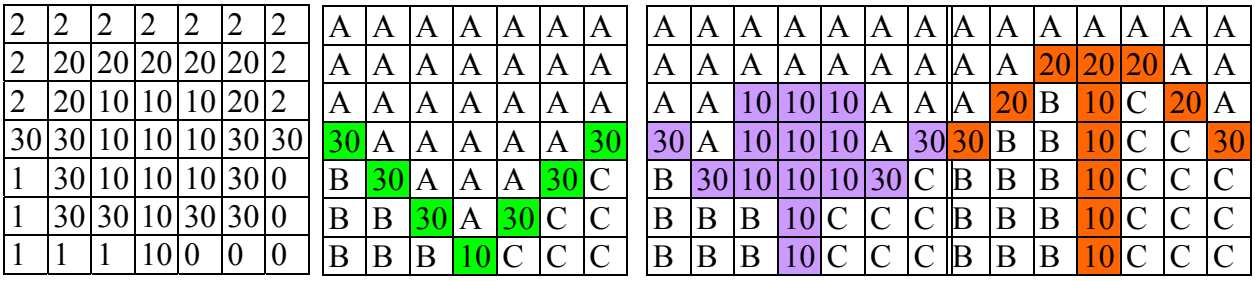


Fig. 4 [a b c d] Different Watershed algorithms. From left to right the original image, Meyer's watershed, Vincent & Soille watershed and topographical watershed

Algorithms based on the definition by immersion are deterministic, while algorithms based on topographical distance depend on the order in which the pixels are processed [10].

3.5.3 Advantages & Disadvantages

Table 2 shows the advantages and disadvantages of (classical) watershed algorithms to segment deformable objects in video.

Table 2

Advantages	Disadvantages
a) Continuous boundary segmentation [9]	i) Oversegmentation when applied to original image (often gradient of image needed) [9, 10]
b) Stable segmentation results [9]	j) Real-time processing (PAL: 25 frames per second) not possible [14]
c) Use of a priori knowledge possible, [9] But difficult [15]	k) No use of temporal information
d) Accurate boundaries, [14] Using topographic watershed [13]	l) No control over shape of catchment basins [15]
e) Good and simple integration in hybrid schemes [13, 15]	

Oversegmentation (Fig. 5b) can be prevented before, during or after the watershed transform. Preprocess with a Gaussian (faster) or adaptive edge preserving (slower) filter to smooth the image before applying the watershed transform. Use marker-based watershed segmentation (discussed further on in this section) and/or introduce a post processing step to merge labeled regions. A combination of these techniques is possible.

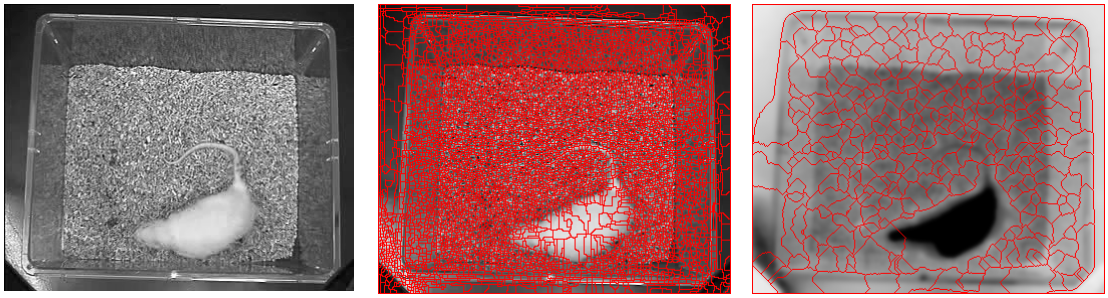


Fig. 5 [a b c] example of oversegmentation and a solution to this problem using markers

The time-complexity of fast watershed algorithms is linear in the number of pixels in the input image [10]. Because of this applying watershed on the region of interest instead of the whole image makes real-time processing possible. Another way to reduce computation time is applying watershed only each x^{th} frame and use a less computational expensive segmentation technique on the intermediate frames.

Temporal information can be used integrated in the watershed algorithm or together with watershed in a hybrid solution. Predictive watershed [14] is an example of the first. Hybrid solutions combine watershed with a motion estimation technique such as optical flow or particle filtering.

Watershed has no knowledge or control over the shape of the resulting contour. As discussed previously energy-driven segmentation techniques such as active contours have control over the shape. In [15] an algorithm is presented which combines energy-driven segmentation with watershed preventing "wrong limps" in object contours. Another possibility to control contour shape is a smart selecting of the markers in marker based watershed [9].

3.5.4 Algorithms

In the watershed transform connectivity (usually N_4 or N_8) affects almost all stages of this process. Using an 8-connected neighborhood is more robust against (random Gaussian) noise [16].

Vincent-Soille algorithm

The best known and fastest implementation of the classical watershed algorithm is the Vincent-Soille algorithm [11] based on immersion. The algorithm consists of two steps:

1. Sorting the pixels with respect to increasing grey values for direct access to pixels of a certain grey level
2. A flooding step proceeding level by level starting from the minima

Result after applying the algorithm to a grey scale image is a labeled image. Figure x shows that the Vincent-Soille algorithm results in watershed areas rather than watershed lines. In fact only topological watershed preserves features such as those shown in Fig. 4 [13].

Watersnake

Drawback of classical watershed is that it has no knowledge on boundary smoothness or shape. Background pixels with an intensity equal to foreground pixels are included in the same region. Fig. 6a shows an example.

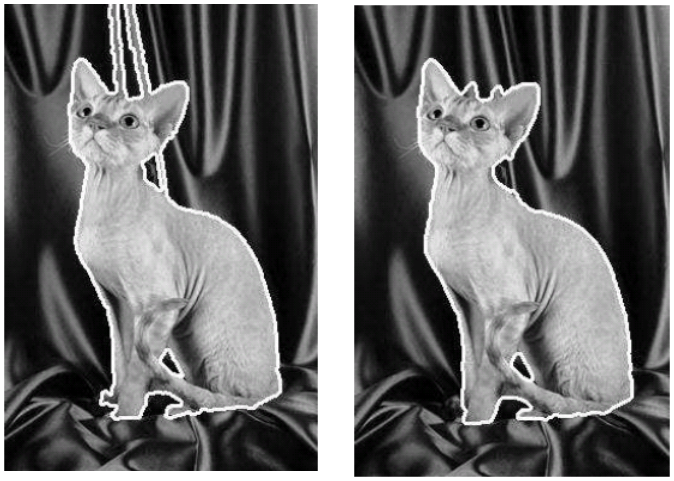


Fig. 6 [a b] Watersnake in action

To tackle these drawbacks in [15] watershed is considered as a minimization problem. Integration of watershed with energy-driven contours, as discussed before in chapter 2.1, combines the advantages of both techniques and is called the watersnake. An example of watersnake applied to an image is shown in Fig. 6b.

The energy yields a tradeoff between accurate region description and contour smoothness. Wrong limbs are prevented by using the contour length in the energy function.

Although the watersnake often results in excellent segmentation it takes effort to find the parameters to select for the energy function. Furthermore watersnake is computational expensive making it with today's hardware hard to use real-time for segmentation of objects in video.

Predictive watershed

A watershed-based technique that can be used real-time for segmentation of video is predictive watershed [14]. Instead of applying watershed on all pixels of each successive frame it is only applied on those areas with a significant change.

The proposed algorithm (P-Watershed) involves the following steps:

1) Apply the Vincent-Soille algorithm to the gradient of the first frame
2) For each successive frame use the difference between the current and previous frame to form an updating area mask (UAM)
3) Divide the UAM in blocks and for each block decide if the content has changed (based on a threshold)
4) Apply the Vincent-Soille algorithm to the updating area (UA)

Each successive frame a small error is introduced leading to error propagation. The segmentation quality of P-Watershed can be improved by applying the Vincent-Soille algorithm on every x^{th} frame and using the technique just described on the intermediate frames. This intra-inter watershed scheme is called IP-Watershed.

3.5.5 Markers

Oversegmentation is the result of processing an image with too many regional minimums. A transformation decreasing the amount of regional minima will prevent oversegmentation. Such a transformation identifying (marking) foreground and background locations is called marker-based or marker-controlled watershed. Two approaches exist: Preprocessing the image to select the regional minima [9] or use a sorted queue of minima in the watershed algorithm of Meyer [10]. Fig. 5c shows an example where the foreground object was marked in a preprocessing phase.

Fig. 7 [abcde] marker-based watershed. b) shows the watershed of the original image in a), c) the foreground marker which, added to the original image results in d). e) is the labeled image after watershed

Another example, in which tissue is segmented, is shown in Fig. 7. To prevent oversegmentation a foreground and background marker is calculated and then superimposed on the original image. The foreground marker is derived from the original image using the image processing operations "opening-by-reconstruction" and "closing by reconstruction"[1] followed by removal of small 'blobs'. Segmentation of Fig. 7d with only one regional minimum at each desired location results in Fig. 7e without any oversegmentation.

3.6 MULTISCALE SEGMENTATION

Multiscale segmentation is not a segmentation technique in itself but an approach to represent an image at different scales using a scale-space. Instead of applying a segmentation technique on the original image, it is applied on each scale.

Integration of multiscale segmentation in the segmentation techniques discussed in this chapter is possible. The key questions are which scales to choose, and how to fuse the results after segmentation in the final image.

[1] reconstruction-based opening and closing are more effective than standard opening and closing at removing small blemishes without affecting the overall shapes of the objects

3.7 BACKGROUND SUBTRACTION

The main goal of background subtraction is given a frame sequence to detect all foreground objects. The approach is defined as

$$\left| frame_i - background_i \right| > T$$

A background image often is not fixed but subject to illumination changes, motion changes and changes in the background geometry. Techniques such as frame difference or taking the average or median the last n frames are fast but have high memory requirements and lack accuracy.

A fast technique with low memory requirements is the running average calculating the background like

$$B_{i+1} = \alpha F_i + (1-\alpha) B_i$$

Objects that stop moving will in time be marked as background. The formula above can be adjusted by identifying a pixel as foreground pixel when it has been a foreground pixel for the last N frames.

Based on this technique is the running Gaussian average where each pixel has its own Gaussian distribution described by the mean and variance which are updated over time. A pixel is labeled as a foreground pixel when the difference between the actual and mean value is smaller than a threshold given by a constant multiplied with sigma. This method is slower but more accurate. Unfortunately it does not work for multimodal backgrounds, then a multi-mixture Gaussian approach is prefered.

An often used background subtraction technique called Kernel density estimators defines the background PDF as the histogram of the n most recent used pixel values, each smoothed with a Gaussian kernel. If

$$PDF(x) > T$$

with T a threshold than x is classified as a background pixel. Kernel density estimators are slower than averaging techniques and have high memory requirements but provide accurate and stable results. A good overview is given in [17].

3.8 OPTICAL FLOW

Optical flow is used to describe a dense motion field with vectors at each pixel. Image displacement is notated as *(u(x,y), v(x,y))* where u and v are vectors; see Fig. 8 for an example. Applications range from motion estimation and tracking to image alignment[2]. Define the image intensity through time as $I(x, y, t)$. Assume that the point moves from (*x, y)* to (*x + dx, y+ dy)* after *dt* time and the intensity of the pixel unchanged, then

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

Taylor expansion of the right side results in

$$I(x + dx, y + dy, t + dt) = \frac{\partial I}{\partial x} dx + \frac{\partial I}{\partial y} dy + \frac{\partial I}{\partial t} dt + e$$

[2] Image alignment consist of moving, and possible deforming, a template to minimize the difference between the template and an image.

which is the addition of the partial derivatives with respect to *x*, *y* and *t* and where *e* stand for higher order terms. Dividing the equation in the previous section by *dt* and in the limit that *dt* tends to zero we can write

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$$

where

$$u = \frac{dx}{dt}, v = \frac{dy}{dt}$$

are the speeds in the x and y direction. The equation above is called the *optical flow constraint equation* which needs to be solved. This equation has two unknowns; further constraints can be incorporated to simplify solving of this equation.

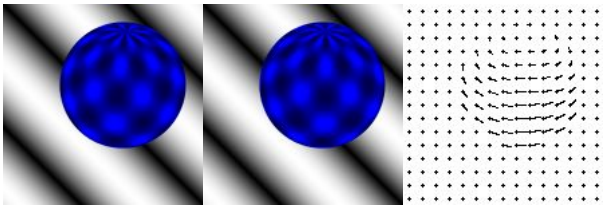


Fig. 8 [a b c] An image at time *t* and t+1 and its optical flow field

Optical flow techniques can be divided in differential techniques, region-based matching, energy-based methods and phase-based techniques. The first algorithm calculating optical flow, almost 25 years ago, by Horn [18] is still amongst the algorithms offering the best performance and discussed in more detail here . Good reviews and comparisons of optical flow techniques are given in [19, 20].

3.8.1 Horn-Schunck algorithm

The Horn-Schunck algorithm for computing Optical Flow adds another constraint, the smoothness constraint, to the equation formulated in the previous section resulting in an efficient algorithm.

A key observation in object tracking is that the pixels of a moving object move dependently of each other, else there would be little hope of calculating velocities using optical flow. Except near the edge of a moving object, the motion that is observed in adjacent pixels will be quite similar. A measure of how much the optical flow deviates from this smoothly varying ideal can be evaluated by calculating the following integral

$$S = \iint \left(E_x u + E_y v + E_t\right)^2 + \alpha^2 \left(\left(\frac{\partial u}{dx}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{dx}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \right) dxdy$$

minimizing, using least-squares, the rate of change of image brightness (first term) and the magnitude of the gradients of the optical flow velocity (second term). α is a constant controlling the weight of the smoothness constraint, which is there because the first term is mostly not identical to zero because of quantization errors and noise.
An iterative discrete solution for the integral can be derived by decoupling a system of two coupled differential equations obtained via calculus of variations. Langrarian multipliers can be used to combine both the optical flow constraint and the smoothness constraint.

Due to the smoothness constraint this algorithm has problems with occluded edges [18].

3.8.2 Optical Flow in practice

An example of Optical Flow in a multiple-object segmentation and tracking system is [21], where laboratory animals in video are tracked real-time. Optical flow is applied only to a region of interest around the tracked animal(s) and an active contour approach called active ray is used to derive the contour(s). This practical example shows that optical flow on its own is not enough to segment object by tracking when the object(s) in question are deformable.

3.9 PARTICLE FILTER

Particle filtering (also known as sequential Monte Carlo approach, bootstrap filtering, and the condensation algorithm) is a probabilistic modeling technique which can be used for object tracking. Doucet et al. [22] give a good overview about the history of particle filter research. To understand particle filtering one should be familiar with probabilistic modeling which will now be introduced shortly.

3.9.1 Probabilistic modeling

Instead of calculating the state of a system or an approximation of the state one can also calculate a degree of belief about the state of a system. Modeling a system is such a way is called probabilistic modeling. In order to analyze and make inferences about such a system two models are required: The system model describing the evolution of states and a model relating noisy measurements to states. With these models in probabilistic form the Bayesian approach is ideally suited.

With probabilistic modeling the posterior state is calculated from the prior state of a system according to Bayes' rule

$$p_t(x_t \mid Z_{1:t}) = \frac{p_t(Z_t \mid x_t)\, p_{t-1}(x_t \mid Z_{1:t-1})}{p_t(Z_t)}$$

Where $p_t(x_t \mid Z_{1:t})$ is the posterior probability distribution, $p_t(Z_t \mid x_t)$ the observation density (also referred to as the likelihood function for x) and $p_{t-1}(x_t \mid Z_{1:t-1})$ the prior probability distribution. x_t is the (multidimensional) target state vector, Z_t is the

measurement/observation at time t. This prior can be rewritten, as to include the term $p_t(x_t \mid x_{t-1})$, so that the equation is

$$p_t(x_t \mid Z_{1:t}) = \frac{p_t(Z_t \mid x_t)\int_{x_{t-1}} p_t(x_t \mid x_{t-1}) p_{t-1}(x_{t-1} \mid Z_{1:t-1}) dx_{t-1}}{p_t(Z_t)}$$

where $p_t(x_t \mid x_{t-1})$ is called the dynamics. This Bayesian recursion is the equation which a stochastical filter should implement.

If the data are non-linear and non-Gaussian, it is possible to derive an exact analytical expression to compute the posterior distributions using a recursive approach called the Kalman filter [22]. Grid-based methods provide an optimal algorithm when the state-space is discrete-valued with a finite number of states [22]. If the data are modeled as a partially observed Markov chain it is also possible to obtain an analytical solution using the Hidden Markov Model (HMM) [22] widely used in the speech processing community.

Unfortunately many real-world problems are complex and have data that is non-linear and non-Gaussian. This is where particle filtering comes into play which is discussed in more detail in the following section.

3.9.2 Basic concepts

Particle filtering is a technique for implementing the recursive Bayesian state estimation. The key idea is to represent the posterior using a weighted time-stamped finite sample-set of (random) particles called a particle set. A particle [20] is defined by the state x (for example a position (x,y)) and weight π. An example of a particle set representing a Gaussian probability distribution is given in figure Fig. 9a.

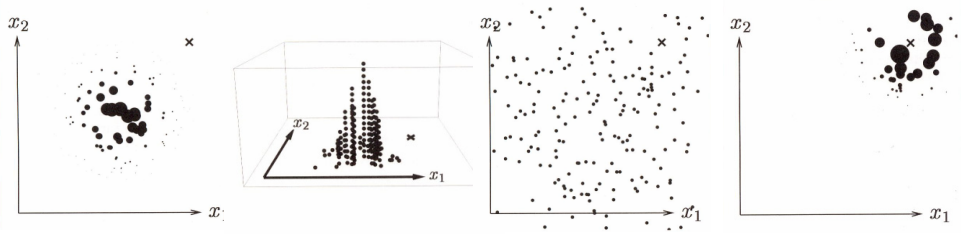


Fig. 9 [abcd] a) shows the sample distribution, which is resampled as shown in b). c) is the result of applying dynamics to c) and d) is reweighting c) by the likelihood of x

Many particle sets can represent a probability distribution; the most efficient representation is when all particles have an equal weight [6]. A weighted particle set can be converted to an unweighted particle set (with the same amount of particles) representing the same probability distribution by resampling. Samples with low importance weights are eliminated and those with high importance weights multiplied. More particles mean better result but more computational power. A good indication of the particle set size can be derived experimentally by looking at stabilization of results.

Most particle filters are implemented using the following recursion:

1. At time *t* resample the prior (which is the posterior of *t-1*)
2. Apply dynamics on the resulting particle set
3. Calculate the posterior at *t* by reweighing 2) with the likelihood of x

The dynamics $p_t(x_t \mid x_{t-1})$ is the probabilistic modeling of prediction the next position of target object x. Often the dynamics are modeled as first- or second-order auto-regressive processes [7] shifting the distribution and adding Gaussian noise. Finally reweighing the result with the likelihood of x (for example a Gaussian centered on the position of the target) results in the posterior.

The strength of particle filters becomes clear when tracking in higher dimensions, for example when tracking object contours. Clever particle filters are in *O(n)*, while for example the complexity of the HMM filter is in $O(n^2)$.

3.9.3 Random Sampling methods

The key feature of an efficient representation of a probability distribution is that (most of) the particles have equal weights. Different random sampling methods have been proposed. One could random resample, but better is to use a deterministic resample scheme. An example of a deterministic resample scheme is

$$x_i^{(n)\,\prime} = x_{j^*(i)}^{(n)}$$
$$\pi_i^{(n)\,\prime} = 1/n$$

where *j*(i)* is the smallest *j* such that the cumulative weight $c_j^{(n)} \geq i/n$. Using deterministic resampling particles with a higher weight are selected more often.

The combined resampling update scheme by Gordon, Salmon & Smith [23] is often used although not optimal in variance. A deterministic resampling scheme that is optimal in variance is proposed in the appendix of Kitagawa [24].

3.9.4 Algorithms for object tracking

Particle filters are used in a wide range of applied sciences. After reading the previous sections it is clear how to apply particle filters for tracking of (deformable) objects in video. A particle has an *x* and *y* component and the probability distribution $p(x_t)$ gives the probabilities of the location of the object. The easiest approach is tracking one object point (for example the center of gravity) following an approach analog to Fig. 9. Other approaches track multiple object points or object templates.

The condensation algorithm [6, 7] integrates an active contour approach with particle filtering. An object contour can be described by a B-spline, specified by a small number of "control points". Instead of tracking one point a whole vector of points (the control points) can be tracked. The probability distribution of x_t is now the 'fit' of the spline.

Bramble (Bayesian Multiple-Blob Tracker) is a Bayesian filter for tracking multiple objects using a particle filter similar to the condensation algorithm [25].

3.10 ACTIVE APPEARANCE & SHAPE MODEL

Statistical models of the shape and appearance of image structures can be matched to images using the Active Shape Model (ASM) and the Active Appearance Model (AAM). This section described a general model to describe the appearance of an object. The process of matching such a model to data using ASM and AAM is discussed hereafter, followed by a comparison of both models. Finally it is shown how these models can be used for segmentation of deformable objects in video.

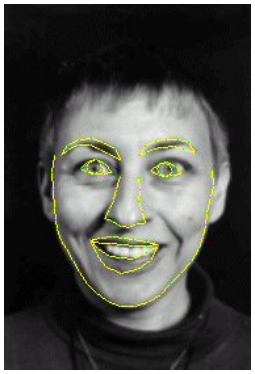


Fig. 10 An Active Shape Model locating a face

3.10.1 Appearance Models

An appearance model can consist of shape and texture location and variability derived from a training set. This training set consists of marked images where the marks are key points in the image. An appearance model of data for example is derived from a training set where the points are on the outline of face and on the edge of the most important features such as eyes, nose, lips and brows.

The statistical model is then generated from the training set as follows: Vector *v* and vector *g* represent the shape and texture of an object and a control parameter *c* such that

$$x = \bar{x} + Q_s c$$
$$g = \bar{g} + Q_g c$$

With is the addition of the mean shape/texture and the multiplication of matrices derived from the training set times the control parameter.

3.10.2 Model Matching

The ASM matches a set of points to a new image using a model similar to the one before without the texture component. Each step of the matching process is an iterative one improving the fit of points from the model to the data. The region of the image around the current model point is examined to find the best nearby match, for example using edge detection. This search is done for each modelpoint. Afterward the model itself can be updated based on the results from the latest match.

AAM seeks to minimize the difference between a new image and one produced using the appearance model using shape as well as texture.

3.10.3 Comparison

There are three key differences between the ASM and AAM [26]:

1. The ASM only uses models of the image texture in small regions about each landmark point, whereas the AAM uses a model of the appearance of the whole of the region.
2. The ASM searches around the current position, whereas the AAM only samples the image in the current position
3. The ASM essentially seeks to minimise the distance between model points and the corresponding points found in the image, whereas the AAM seeks to minimise the difference between the synthesized model image and the target image

Compared to each other ASM is faster and achieves more accurate feature point location but AAM gives a better match to texture.

Applying ASM or AAM to segmentation of deformable objects in video is computational expensive and cannot be done real-time on nowadays computers [26]. In case of very deformable objects a good approach is an atlas of appearance models with a search for the best model before fitting the model to the data.

4 COMPARISON

	Adaptive Thresholding	Active Contours	Watershed Transform	Optical Flow	Particle Filter	AAM /ASM
Noise	Sensitive to noise	Sensitive to noise; requires preprocessing or noise model	Preproc. step to remove noise	'Noise-term' in constraint	Good tracking performance in noisy environment	More feature points = less sensitive
Illumination changes	Sensitive to changes in illumination	Sensitivity depends on the chosen energy functions	Remove illumination changes in a preproc. step	Problems with changes in illumination	Sensitivity depends on observation likelihood	AAM more sensitive than ASM
Color	Yes	Yes	Yes	Yes	Yes	Yes
Multiple objects	Postprocessing step required to group pixels and label objects	Yes	Yes	Yes	Suitable for tracking multiple objects	Yes, when number of objects is known
Real-time processing	Yes, very fast	Most implementations run in real-time	Only the most efficiënt watershed algorithms run in real-time	Yes	Yes	No
Occlusion	No awareness of objects, so does not handle occlusions	Yes, but initial contour should approximate the object form	Does not handle occluded objects	Can handle occluded objects	Handles occluded objects well	Does not handle occluded objects
Moving camera	Yes	Yes	Yes, each segmentation is independent of previous segmentations	No	No, unless camera motion is taken into account in the dynamics and observation likelihood	Yes, but has difficulty with rotated objects
Multiple dimensions	Yes	Yes, but real-time performance not possible in higher dimensions	1D or 2D (resulting in 2D or 3D landscapes)	Does not perform real-time in higher dimensions	Good performance in higher dimensions	2D or 3D
Various		Initial location must be close to object. Handles 'gaps' in the contour				Appearance model needs training before application

5 EXPERIMENTAL RESULTS

Research for this research assignment included handson experience with most segmentation and tracking techniques to get a better understanding of what's happening at the lowest level of source code. Several image processing environments where used for this purpose. Some existing algorithms were tested, others adjusted as to fit in a framework and some coded from scratch. This section shows some of the bunch of results.

5.1 PARTICLE FILTER A-LA DENNIS

Fig. 11 shows a screenshot from a particle filter tracking an object which actual position is the mouse cursor. The main result of this experiment, besides handon experience, is knowledge about the importance and sensitivity of the observation likelihood (here a 1st order autoregressive function).

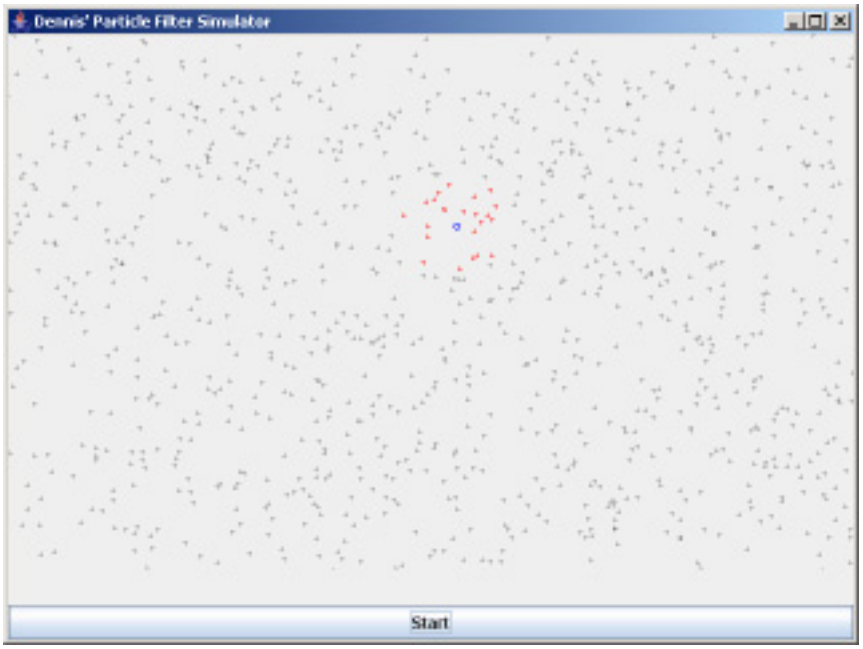


Fig. 11 Particle Filter tracking the mouse pointer (particles: red, mouse: blue)

The simulator consists of Particle objects (represented by old and new coordinates and a weight) and a ParticleFilter with the main functionality to obtain observations, select and resample particles, apply dynamics and reweight with likelihood.

5.2 RUNNING AVERAGE

Using the ImageJ[3] image processing environment background subtraction techniques were taken into practice. The running average algorithm , as discussed in section 3.7, was implemented and tested with several parameters for the weights which control the rate of change. The algorithm was tested on a movie with a moving caged rat resulting in a movie of the changing background image[4]. One frame is shown below.

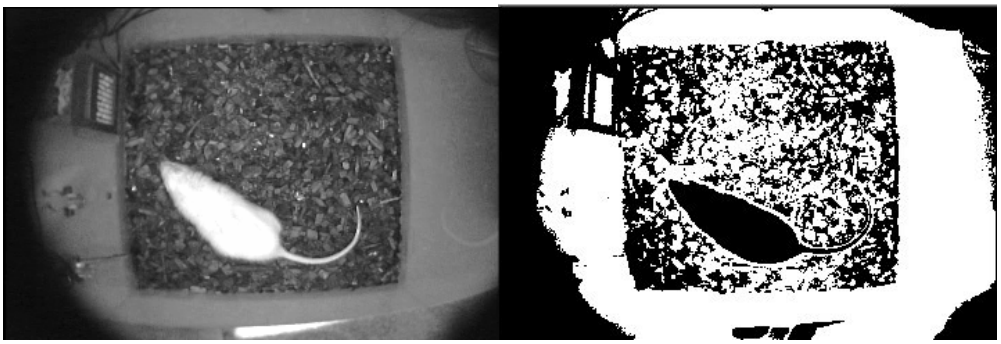


Fig. 12 [a,b] Original movie frame and the same frame with the running average subtracted

The most interesting result is that the background keeps adapting when the object stops moving (for example when the rat goes to sleep) and gradually the object dissapears from the image subtracted with the background. Masking the foreground pixels prevented these drawback. Gradual illumination changes and noisy images resulted in bad segmentation results for this approach. Afterwards a per pixel Gaussian distributed was stored and a threshold was based on the variance and not only on the mean. Probably integration with a masked foreground would result in even better results.

Experiments with a "mixture of Gaussians" technique where the background is modeled by several Gaussian distributions where started but not completed due to time constraints. Result was a knowledge gain on curve fitting and statistical techniques.

[3] http://rsb.info.nih.gov/ij/
[4] Movie on request available from the author

6 CONCLUSION

There exists no universal algorithm for segmentation of deformable objects in video thus the answer to the question "what is the best segmentation technique for segmentation of deformable objects in video taking into account noise, (changes in) illumination and color?" does not exist. Some algorithms deal better with noise then others, but are compututional expensive. Some handle segmentation of occluded objects, while other algorithms only perform well with a static background.

Clearly some segmentation techniques are favored above others, just look at the comparison in section 4, and some combinations of segmentation & tracking techniques seem to perform better. Although a simple answer to the research question does not exist, guidelines for selecting the best segmentation technique can be given.

Knowledge-driven segmentation approaches are more robust then "low level" segmentation techniques such as (adaptive) thresholding and often result in closed object contour. When real-time performance is required a parametric active contour approach is favored to segment deformable objects. If offline segmentation is an option the marker-based watershed transform also gives robust, accurate results depending on the pre-processing.

Robustness and effectiveness of knowledge-driven segmentation is improved greatly by incorporating temporal knowledge using a tracking technique, instead of frame-per-frame segmentation. A probablistic approach, modeling the belief in the state of the world, is favored. Amongst them are the particle filter and the Kalman filter where the first is prefered in a non-linear, non-gaussian system. Both don't need training before they can be used. Background subtraction requires less computational power and is a good candidate when the background is static.

When combining a knowlegde-driven segmentation technique with a tracking method for real-time segmentation one is often limited by computational power on nowadays computers. A tradeoff between robustness vs accuracy of contour is often needed. Especially when dealing with color instead of greyvalue video. Knowledge about image noise (SNR, Gaussian, non-Gaussian?), illumination changes (local, global, lightning source?) should be studied before deciding on the choice of algorithms.

BIBLIOGRAPHY

[1] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *International Journal of Computer Vision*, pp. 321-331, 1988.

[2] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic Active Contours," *International Journal of Computer Vision*, vol. 22, pp. 61-79, 1997.

[3] C. Xu and J. L. Prince, "Snakes, Shapes and Gradient Vector Flow," *IEEE Transactions on Image Processing*, vol. 7, pp. 359-368, 1998.

[4] M. Gastaud, M. Barlaud, and G. Aubert, "Combining Shape Prior and Statistical Features for Active Contour Segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, pp. 726-733, 2004.

[5] J. Denzler and H. Niemann, "Active Rays: Polar-transformed Active Contours for Real-Time Contour Tracking," *Real-Time Imaging*, vol. 5, pp. 203-213, 1999.

[6] J. MacCormick, *Stochastic Algorithms for Visual Tracking*: Springer, 2002.

[7] A. Blake and M. Isard, *Active Contours; the Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visuak Tracking of Shapes in Motion*, 2nd ed: Springer, 1998.

[8] F. Precioso and M. Barlaud, "Smoothing B-spline active contour for fast and robust image and video segmentation," *IEEE*, pp. 137-140, 2003.

[9] R. C. Gonzalez and R. E. Woods, "Digital Image Processing," 2nd ed. New Jersey: Prentice Hall, 2002, pp. 567-642.

[10] J. B. T. M. Roerdink and A. Meijster, "The Watershed Transform: Definitions, Algorithms and Parallelization Strategies," *Fundamenta Informaticae*, vol. 41, pp. 187-228, 2001.

[11] L. Vincent and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, pp. 583-598, 1991.

[12] F. Meyer, "Topographic distance and watershed lines," *Signal Processing*, vol. 38, pp. 113-125, 1994.

[13] L. Najman and M. Couprie, "Watershed algorithms and contrast preservation," *Lecture Notes in Computer Science*, vol. 2886, pp. 62-71, 2003.

[14] S.-Y. Chien, Y.-W. Huang, and L.-G. Chen, "Predictive Watershed: A Fast Watershed Algorithm for Video Segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 453-461, 2003.

[15] H. T. Nguyen, M. Worring, and R. v. d. Boomgaard, "Watersnakes: Energy-Driven Watershed Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, pp. 330-342, 2003.

[16] F. B. Tek, A. G. Dempster, and I. Kale, "Noise sensitivity of watershed segmentation for different connectivity: experimental study," *IEE Electronic Letters*, vol. 40, 2004.

[17] A. McIvor, Q. Zang, and R. Klette, "The Background Subtraction Problem for Video Surveillance Systems," presented at Robot Vision: International Workshop, 2001.

[18] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *Artificial Intelligence*, vol. 17, pp. 185-203, 1981.

[19] M. Lucena, J. M. Fuertes, and N. P. d. l. Blanca, "Evaluation of three optical flow-based observation models for tracking," presented at 17th International Conference on Pattern Recognition, 2004.

[20] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of Optical Flow Techniques," *International Journal of Computer Vision*, vol. 12, pp. 43-77, 1994.
[21] Z. Kalafatic, S. Ribaric, and V. Stanisavljevic, "A System for Tracking Laboratory Animals Based on Optical Flow and Active Contours," *IEEE*, pp. 334-339, 2001.
[22] A. Doucet, N. d. Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag, 2001.
[23] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," presented at IEE Proceedings on Radar and Signal Processing, 1993.
[24] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of Computational & Graphical Statistics*, vol. 5, pp. 1-25, 1996.
[25] M. Isard and J. MacCormick, "BraMBLe: A Bayesian Multiple-Blob Tracker," presented at Proc. 8th Int. Conf. Computer Vision, 2001.
[26] T. F. Cootes, G. Edwards, and C. J. Taylor, "Comparing Active Shape Models with Active Appearance Models," presented at British Machine Vision Conference, 1999.