EVALUATING THE PERFORMANCE OF ACTIVE CONTOUR MODELS FOR REAL–TIME OBJECT TRACKING

J. Denzler, H. Niemann

Universität Erlangen–Nürnberg
Lehrstuhl für Mustererkennung (Informatik 5)
Martensstr. 3, D–91058 Erlangen
Germany
email: {denzler,niemann}@informatik.uni-erlangen.de

Abstract

In the past six years many algorithms and models for active contours (snakes) have been presented. Some of the work has been applied to static image analysis, some other to image sequence processing. Despite of the fact that snakes can be used for object tracking, no comparative study of the performance for real–time object tracking is known up to now.

In this paper we compare several active contour models presented earlier in the literature for object tracking: the "Greedy" algorithm, the dynamic programming approach, and the first work of Kass, based on the variational calculus. We discuss and compare the various active contour models with respect to the quality of contour extraction, the computation time and robustness. All evaluation is done using sequences, grabbed during closed–loop real–time experiments.

1 Introduction

Active contour models (snakes) have been proven to be a promising approach in many different fields in computer vision. They have been applied to image segmentation and to the analysis of static images and image sequences, even in real–time systems [2, 5, 10]. The inherent local processing of an image — nearby the snake elements — also makes active contour models suitable for the use in real–time active vision systems.

Many extensions of the original approach have been suggested, referring to the definition of the energy as well as to the energy minimization [1, 2, 3, 4]. But, the models have always been presented in a special area of applications. Thus comparisons between the different approaches or predictions of the behavior of one model in another area of application can hardly be drawn. Especially in the field of real–time object tracking in a closed loop of sensor and actor, different special points should be taken into account while estimating the suitability of an active contour model: the computation time for extracting the moving object's contour, the robustness of the contour extraction also in the case of weak object's contours, or changing illumination conditions; another aspect is the accuracy of contour extraction, also in the case of shrinking or growing contours due to changing views of the object. Finally, the possibility of an automatic initialization should be taken into consideration.

Up to now, no comparison of the different models and energy minimization schemes for real–time object tracking is known, which enables someone to choose the optimal active contour model. Thus, in our contribution we investigate the most important active contour models, i.e. the dynamic programming approach, the greedy algorithm and the original model based on the variational calculus, and draw qualitative comparisons referring to the suitability in real–time object tracking. We make use of the detected weak and strong points of each model to define extensions for the variational approach, which we show to be suited to fulfill the above demanded properties of an active contour for tracking in a real–time system.

The comparison is done in a real–time object tracking system [6, 7]. A qualitative judgement for the quality and robustness of the contour tracking is employed. No specialized hardware but only standard Unix workstations are used.

2 The Active Contour Models in the Comparison

An active contour can be described as a parametric function $\boldsymbol{v}(s) = (x(s), y(s))$, $s \in [0, 1]$, with $x(s) \in [0, x_{max}]$, $y(s) \in [0, y_{max}]$. Such an active contour has an energy E^* defined by

$$E^* = \int_0^1 [E_i(\boldsymbol{v}(s)) + E_f(\boldsymbol{v}(s)) + E_c(\boldsymbol{v}(s))]\, ds. \quad (1)$$

In most cases the internal energy E_i is given by

$$E_i(\boldsymbol{v}(s)) = \frac{1}{2}\left(\alpha(s)|\boldsymbol{v}_s(s)|^2 + \beta(s)|\boldsymbol{v}_{ss}(s)|^2\right), \qquad (2)$$

where $\boldsymbol{v}_s$ and $\boldsymbol{v}_{ss}$ are the first and second derivatives of $\boldsymbol{v}$ with respect to s. E_f describes the forces of the image on the snake and E_c summarizes all the other constraints of the snake, for example, connections of snake elements to image features (spring forces) or the limitation of the distance between the snake elements [9].

In our contribution we compare the following active contour models:

- Energy minimization with the variational calculus [9] (Abbreviation VARN — with matrix inversion after each minimization step, VAR — with matrix inversion only once for each new image)
- The dynamic programming approach [1] (DP)
- The "Greedy" algorithm [11] (GRDY)

Additionally, we present several improvements of the active contour model based on the variational Calculus (IMPR) with respect to real–time object tracking:

- For each snake element we fix it on its old position with a weak spring force.

$$E_c(\boldsymbol{v}(s)) = \sigma_1(\boldsymbol{v}(s) - \boldsymbol{v}_{old}(s))^2 \qquad (3)$$

  The vector $\boldsymbol{v}_{old}(s)$ is the position the snake has reached after the energy minimization in the previous image. This limits the possible movements of the elements during the energy minimization, because large movements are punished in the energy term, weighted with the parameter σ_1. We use the same value σ_1 for all snake elements.

- For decreasing the computation time we minimize the energy without inverting the matrix $(\boldsymbol{A} + \gamma\boldsymbol{I})$ [9] at each minimization step. We compute the inverse matrix only once at the beginning of the tracking. This corresponds to the calculation of the snake element's position out of a linear combination of its neighbors, where the influence of neighboring elements decreases with the distance to the actual element (see Figure 1). Because of the computation time reduction the number of iteration steps can be increased and we can choose a smaller value for the step size of the iterative minimization.

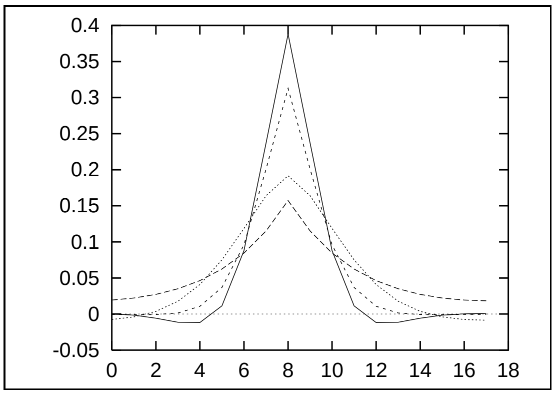

Figure 1: Influence of the snake elements on the position of the 8th element for different combinations of α and β using a constant matrix $\mathbf{A}$ (x–axis: snake element, y–axis: influence of the snake element's position).

- We use blowing forces known from the balloon model [3]. Because an active contour based on the variational calculus tends to shrink, we blow up the snake before minimizing the energy for the new image.

$$\boldsymbol{v}_{start}(s) = \boldsymbol{v}_{old}(s) + \sigma_2\boldsymbol{n}(s) \qquad (4)$$

  The vector $\boldsymbol{n}(s)$ is the normal unity vector on the contour point $\boldsymbol{v}_{old}(s)$, directing outside the contour. $\boldsymbol{v}_{start}(s)$ is the starting position of the snake for the new image. The same value σ_2 is used for all snake elements.

Our real–time experiments show that these improvements lead to a more robust object tracking compared to the original model, i.e. the number of times, the tracked object is lost can be reduced, and to a more accurate one, i.e. the quality of contour extraction is improved.

3 Experimental Environment

We integrate the models into the tracking system for real scenes described in [6, 7]. Within this closed–loop object tracking, we can evaluate the efficiency for real–time applications. A moving toy train is tracked by steering a robot's arm holding a camera. The system is divided up into two parts: the object detection and the object tracking using snakes. The control of the robot is done on a second machine, connected with the object tracking process by using a programming library for communication between workstations in a workstation cluster (*PVM*, [8]). No specialized hardware for computation is used. All algorithms are running on standard Unix workstations (HP-735), implemented in an object–oriented image processing environment. We judge the results in a qualitative manner. During the tracking the

method	speed	max # tracked	# lost	$p^{\boldsymbol{c}}_{20}$	$p^{\boldsymbol{c}}_{10}$	$p^{\boldsymbol{c}}_{5}$
VAR	0.8	426	1	93	43	13
	1.4	326	3	76	11	8
	1.8	205	16	73	33	8
	2.4	153	18	66	33	8
VARN	0.8	600	0	100	71	12
	1.4	337	5	97	84	18
	1.8	255	6	92	71	18
	2.4	105	20	66	40	7
GRDY	0.8	387	1	92	46	7
	1.4	483	2	100	63	36
	1.8	259	3	97	46	4
	2.4	119	19	77	50	8
IMPR	0.8	453	1	76	75	33
	1.4	453	1	97	70	29
	1.8	599	0	98	92	30
	2.4	313	5	96	75	15

Table 1: Results over 600 images for each model: the speed of the object (cm/sec), the max. number of consecutive images, for which the object could be tracked, the number of times the object has been lost, and the percentage in which the object is less than 20,10 and 5 pixels from the center of the image (128,128) (see equation (5)).

frame grabbing and robot motion is always done in such a way that the center of the snake coincides with the center of the digitized camera image. We use a simple light object in front of a dark background. By calculating the center of mass $\boldsymbol{c}_m(t)$ of the pixels with a value greater than a given threshold we get the real center of the object in image at time t. Comparing these coordinates with the center of the image which should be identical in the case of a precise tracking, we get a measurement for the tracking error:

$$p^{\boldsymbol{c}}_d = \frac{|\{t \mid |\boldsymbol{c}_m(t) - \boldsymbol{c}| \leq d\}|}{n} \qquad (5)$$

where $p^{\boldsymbol{c}}_d$ is the percentage of images, in which the center of mass of the object is less than d pixels from the center $\boldsymbol{c}$ of the image at time t, n being the number of images.

We have run several experiments for every snake model. The speed of the object has been varied between 0.8 cm/sec and 2.4 cm/sec. Due to a very moderate frame grabbing rate of our workstations this is equal to a displacement in the image between 4 and 8 pixels at a distance of 1.5–2.0 meters to the moving object. The energy field has been smoothed with a 3×3 mean filter. In the experiments each model was used to track 600 images at four different speeds (see Table 1). Since we are interested in real time tracking, we are forced to choose the number of iterations and the number of contour elements in a way such that the energy minimization can be done within the image frame rate (about 3 frames per second).

The chosen experimental environment is, of course well suited for other object tracking methods, for example line–based tracking algorithms. Since we are not interested in judging the method of active contours for real–time object tracking, but in comparing the different approaches within this framework, our experimental environment is well suited to get informations about the behavior of the models in a real–time closed–loop application, i.e. automatic initialization, robustness and accuracy. Nevertheless, we have taken another sequence (Figure 4) to verify our results, for which, for example a line–based tracking method, would fail.

4 Results

Table 1 shows the results of the algorithms. In Figure 2 representative images of the tracking with the proposed improvements (IMPR) are shown. The results for the dynamic programming approach cannot be found in Table 1 because even for the slowest speed of the object, the computation time for the energy minimization has been too large. One can see that VARN leads to a more accurate result than VAR. But the complexity of the algorithm increases, too. Thus, for faster moving objects, the stability of the tracking decreases. Using the proposed improvements (IMPR), i.e. no matrix inversion at all and spring forces with an initial blow up of the snake, we got the best result in our experimental environment. In Table 1 the stability (4th column) and

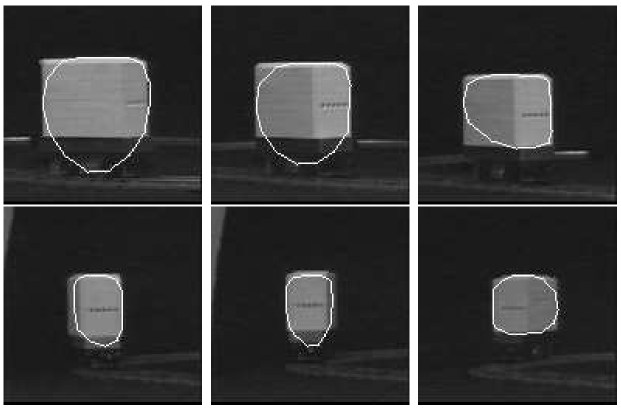

Figure 2: Results of the tracking with the proposed improvements (IMPR) at a speed of 1.8 cm/sec. Images 50, 100, 150, 200, 250 and 300 of an sequence of 600 images are shown, grabbed during a real–time experiment.

quality (column 5–7) of the models at different speeds can be compared. As one can see, all models show a better tracking quality at a speed of 1.4 cm/sec, compared to the result at a speed of 0.8 cm/sec. The reason

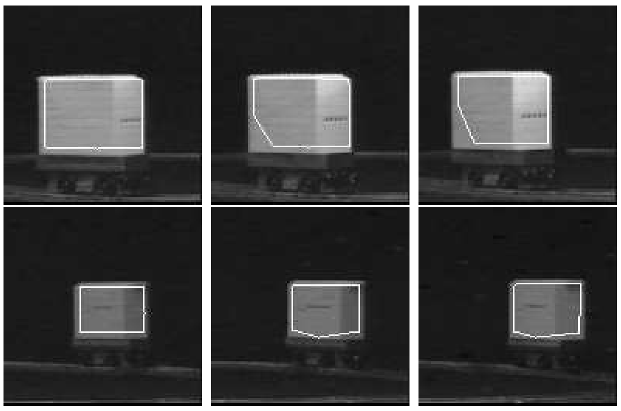
Figure 3: Results of the tracking with GRDY at a speed of 1.6 cm/sec. Images 150, 180, 210, 450, 480, and 510 of an sequence of 600 images are shown, grabbed during a real–time experiment.

METHOD	TIME (msec)
DP	800
VAR	110
VARN	490
GRDY	10
IMPR	80

Table 2: Computation time for one image on a HP-735. All results are taken from a snake with 16 elements.

is that the robot has a minimum speed which is too fast for tracking smoothly the moving object at a speed of 0.8 cm/sec. Thus the robot performs a lot of short movements which results in a less accurate tracking.

The quality of contour extraction also varies. GRDY extracts the contour of the object accurately (see Figure 3) because this method allows to form corners by automatically setting $\beta(s)$ of the internal energy to zero (see equation (2)), dependent on the image data. Using DP one can extract more complex contours, but as already mentioned, the computation time is too large. VAR and VARN are very sensitive to noise in the image, VARN tends to produce a more stiff contour. This is advantageous for tracking, if the contour of the object does not change. For an automatic initialization this behavior leads to more errors, if the initial contour has not the shape of the object's contour. This will be true in nearly all data driven initializations. If a model based initialization is available, which results in an accurate shape description of the contour, the stiffness might be advantageous.

The contour extracted with IMPR looks in most of the cases like an egg. Thus, the extracted contour cannot be used for a segmentation of the image, and one needs an additional step to extract the contour of the object accurately. For that, IMPR provides the region, in which the segmentation (for example line extraction) should be done.

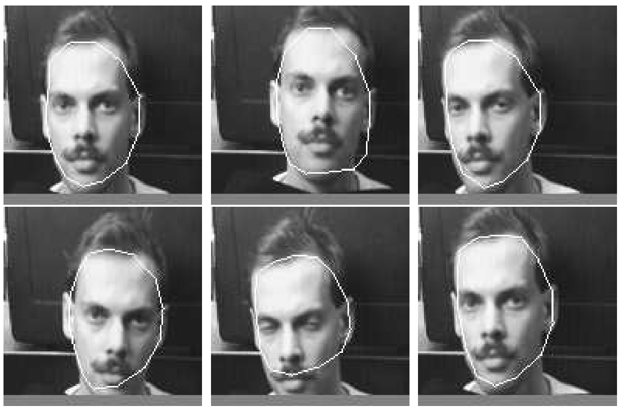
Figure 4: Results of the offline tracking with VARN for the head–sequence (images number 0,30,60,90,108,111). Images 108 and 111 show the effect of the blowing forces: Although partially loosing the contour, the active contour catches again the contour of the head.

In Table 2 the computation time for the different models can be seen. We have measured the time for extracting the contour of a moving object in one image using 16 contour elements. The time for image acquisition and preprocessing (edge extraction and smoothing) is not included. For DP and GRDY the convergence criterion described in [11] has been applied. For VAR, VARN and IMPR 400 iterations have been carried out.

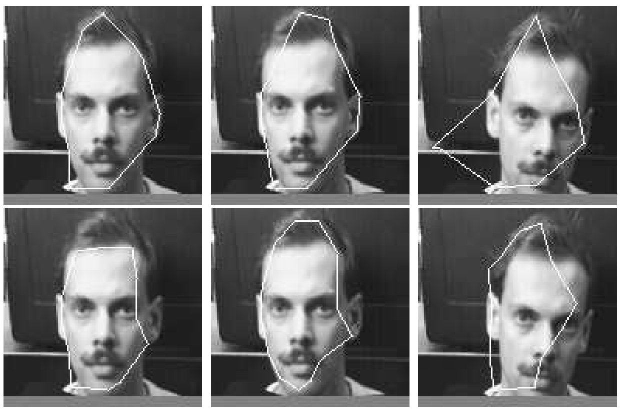
Figure 5: Results of the offline tracking with DP (first row) and GRDY (second row) for the head–sequence.

In Figure 4 (IMPR) and Figure 5 (DP and GRDY) the verifications of the real–time results on another image sequence processed offline can be seen. Especially in Figure 5 possible source of errors for GRDY and DP can be seen. First, for larger displacements (in these images 12 pixels) the search area for the energy minimization might be too small. By increasing the search area, the chance to extract a background edge near the object increases, too. Using approaches based on the variational calculus (VAR, VARN, IMPR), the maximal displacement depends on the smoothing of the external energy. Additionally, this does not influence the computation time of the energy minimization. Of course, by largely smoothing the external energy the edges of small objects can disappear, which is one disadvantage of the variational calculus approach. Secondly, single snake elements, which are in a homogeneous area (for example, the horizontal background edge in Figure 5, first row) with a large external energy compared with the rest of the image, remain in this area, until the internal energy grows sufficiently to force a movement of these elements. This depends on the parameter setting and is difficult to adjust.

5 Summary

In our work, different snake models have been compared with respect to real-time contour tracking. The time complexity of the dynamic programming approach is too large and thus this approach is not suited for tracking a contour in real–time without specialized hardware. Nevertheless, this approach might be applied to static image analysis. The "Greedy"–algorithm is computationally inexpensive and therefore well suited for real–time applications. The variational approach leads to a better result, if the matrix is inverted at each minimization step. But this needs too much computation time without specialized hardware. Thus tracking is only possible for slowly moving objects. The energy minimization with a matrix inversion for each new image is, of course, computationally less expensive but the results of the tracking can be improved only moderately for faster moving objects. By visually judging the result of the active contours in Figure 2 and Figure 3 one gets the impression that GRDY might be the best approach in this context. But the overall results of the experiments at different speed (Table 1) show that the best result in the context of data driven real–time object tracking with respect to stability and quality, could be achieved with our proposed improvements (IMPR). Even for the maximum speed of 2.4 cm/sec the tracking can be done very robustly compared to other models. But, as already mentioned, this method lacks the necessary accuracy for a segmentation of the object's contour. For this field of application DP and GRDY should be used.

We have not included a prediction step in our comparison, because in this case the quality of the tracking mostly depends on the quality of the prediction step and not on the applied active contour model.

References

[1] A. Amini, T. Weymouth, and R. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867, 1990.

[2] A. Blake and A. Yuille, editors. *Active Vision*. MIT Press, Cambridge, Massachusetts, London, England, 1992.

[3] L.D. Cohen and I. Cohen. Finite-element method for active contour models and balloons for 2–D and 3–D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147, 1993.

[4] R. Curwen and A. Blake. Dynamic contours: Real–time active splines. In A. Blake and A. Yuille, editors, *Active Vision*, pages 39–58. MIT Press, Cambridge, Massachusetts, London, England, 1992.

[5] C. Davatzikos and J.L. Prince. Adaptive active contour algorithm for extracting and mapping thick curves. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 524–529, Ney York City, 1993.

[6] J. Denzler and H. Niemann. Combination of simple vision modules for robust real–time motion tracking. *European Transactions on Telecommunications*, 5(3):275–286, 1995.

[7] J. Denzler and D.W.R. Paulus. Active motion detection and object tracking. In *First International Conference on Image Processing*, pages 635–639, Austin, Texas, USA, 1994.

[8] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. PVM 3.0 user's guide and reference manual. Technical Report ORNL/TM-12187, Engineering Physics and Mathemaics Division, Oak Ridge National Laboratory, Tennessee, 1993.

[9] M. Kass, A. Wittkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 2(3):321–331, 1988.

[10] F. Leymarie and M.D. Levine. Tracking deformable objects in the plane using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):617–634, 1993.

[11] D.J. Williams and M. Shah. A fast algorithm for active contours and curvature estimation. *Computer Vision, Graphics, and Image Processing*, 55(1):14–26, 1992.