Statistical Feature Modelling for Active Contours

Simon Rowe and Andrew Blake

Dept. of Engineering Science,
Oxford University,
Parks Road, Oxford OX1 3PJ UK

Abstract. *A method is proposed of robust feature-detection for visual tracking. Frequently strong background clutter competes with foreground features and may succeed in pulling a tracker off target. This effect may be avoided by modelling the appearance of the foreground object (the target). The model consists of probability density functions of intensity along curve normals—a form of statistical template. The model can then be located by the use of a dynamic programming algorithm—even in the presence of substantial image distortions. Practical tests with contour tracking show marked improvement over simple feature detection techniques.*

1 Introduction

This paper details recent work aimed at making curve matchers and trackers more robust. A major problem in achieving robust curve tracking is the distracting effect of background objects—clutter. Strong features in the background compete for the attention of the tracked curve and may eventually succeed in pulling it away from the foreground object. This effect is clearly visible in figure 1 Immunity to distraction can be enhanced both by modelling of the foreground and of the background. A foreground model may include a shape template, object dynamics [3, 10] and intensity profiles for certain object features [12, 4]. A background model may use simple differencing with a gray level reference image, or a more detailed statistical model [9]. However background modelling does not help with a moving background (other than a rotating camera[9]).

In this paper we propose a method to reduce the effect of clutter by using a model of the gray level intensity of the target. This model is built along *lines* in an image, typically normals to an estimated curve position.We concentrate on the problem of finding a *feature* along these lines in the image. If this feature is an intensity profile, the problem is one of template matching. One technique which has been very successfully applied to this problem is correlation [2], unfortunately this suffers from some problems related to none-linear changes in the image of the target. This paper proposes a more flexible way to find the best match for the template on the image line.

The gray level profile of an image line overlying a moving target will vary for many reasons, such as shadows, sub-sampling of its texture and rotation of the target perpendicular to the image plane. Since these effects interact it will be

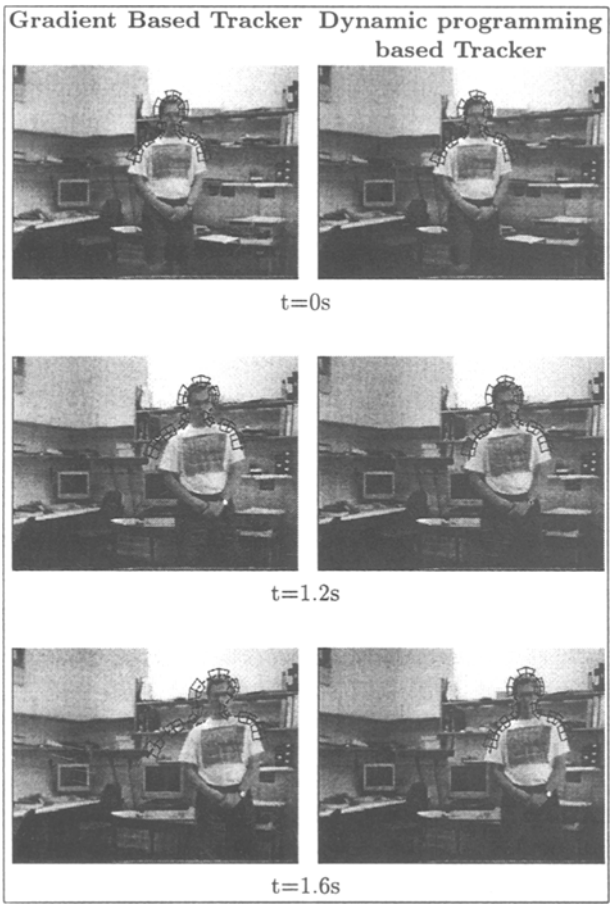


Fig. 1. A dynamic programming based tracker outperforms a gradient based one. The left column shows a tracking sequence obtained using a gradient based tracker – The tracker is distracted by the strong edge and fails to track the target. The right column shows the same sequence, but using the dynamic programming based feature search. The tracker continues to track the target past the strong distracting clutter.

almost impossible to predict the profile exactly. It may however be possible to build a statistical model incorporating the unknown elements of this variation, and use this model to locate the target—this is the approach used in this paper.

Woods, Taylor *et al* [11] use a Gaussian model for the intensity of pixels along an image line as their template. They first divide the image line into sections of constant intensity and then form statistics as to how well each section fits the constant intensity model. This intensity template is based on an analysis of hand picked regions in typical images. In the matching phase each image line

is first divided into sections of *constant* intensity. The correspondence between the image line and the template is then obtain by minimising a cost function (a weighted sum of differences between the image and the model) over all possible interpretations of the model (although how exactly this is done is unspecified). The model is shown to work well in matching image lines containing large regions of constant intensity (as are typical in their application), but its applicability to more general models is doubtful.

In our work we similarly use a statistical model for our template. However rather than making any (possibly invalid) assumptions about distributions, we use an experimentally determined probability density function (pdf) for the intensity function, stored as an array of histogram. We also learn the pdf for stretches between the search line and the template. The pdf's are learnt from *real* data obtained *automatically* by tracking the target on a live video stream, rather than being hand crafted from proto-typical images, or given *a priori*. Image search lines are then matched to the template using a dynamic programming algorithm. The use of a pdf enables the algorithm to be more or less strict about how tightly intensities along the image line need to match the template (depending on where along the template they are trying to match). Using the dynamic programming algorithm enables efficient calculation of the warp from the image line to the template—and takes into account both intensity and stretching effects. The new methods are tested using a tracker based on snakes deforming over time [8], represented by B-spline curves.

Dynamic Programming has been used previously with Active Contours but in a very different context. Whereas Amini et al. [1] applied dynamic programming to the problem of matching curves to models under certain constraints, here we use dynamic programming to identify features by matching templates to intensity data along certain lines.

The layout of the rest of this paper is as follows. Section 2 introduces the notation we use to describe the feature search algorithm. Section 3 then actually explains the algorithm. Sections 4 and 5 show how the probability density functions used by the algorithm may actually be determined in practice. Section6 gives a brief explanation of the active contour tracker used to obtain the results given throughout this paper. Finally section 7 draws some conclusions on the method.

2 Feature search using a statistical foreground model

Before introducing the template matching algorithm, we first introduce notation. We then describe the algorithm and the acquisition of the gray level statistical template. The image line (or search line) under consideration at time t is denoted I_t, and the intensity a distance r along it as $I_t(r)$. It is defined it from $\pm\rho_t$ (this length being a function of time). The template is a pdf of intensity. It exists at a discrete set of positions (spaced $\bar{\delta}$ apart) along a given normal to the active contour. We define the template to be a density function, $\bar{\phi}$, so that the

probability that intensity γ was generated by $\bar{\phi}$ will be denoted:

$$\bar{\phi}(\bar{r}, \gamma), \qquad -\bar{\rho} \leq \bar{r} \leq \bar{\rho} \tag{1}$$

where $\bar{r}$ is distance along the template and this is illustrated in figure 2. Details of how $\bar{\phi}$ may be estimated are given later in Section 4 .

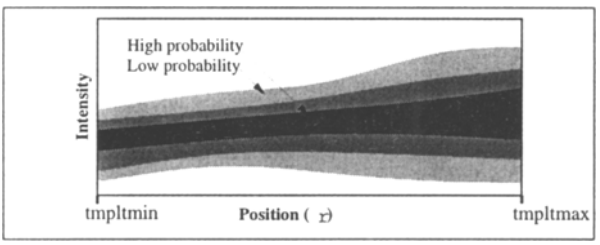


Fig. 2. The statistical template, $\bar{\phi}$. The template is a pdf for the intensity function along a prototypical search line. The darker colours in the figure indicate higher probability intensities. It is the task of the feature search under discussion here to find the warping which produces the closest fit of a search line to the template, given some constraints on the warps which are actually allowed.

The problem facing us when searching for a feature is to match a search line (I_t) to the template ($\bar{\phi}$).The search line is deformed relative to the template; to *undo* these deformations we need to find the warp which takes I_t and matches it, as far as possible, to $\bar{\phi}$. This is done by considering all possible warps of the search line and picking the one which matches the template with the highest probability. The warp function $f_t(r)$ is defined by $\bar{r} = f_t(r)$ and the warped intensity function is $\tilde{I}_t$ given by:

$$\tilde{I}_t(f_t(r)) = I_t(r) \tag{2}$$

where $f_t(\bar{r})$ is the mapping, assumed invertible, from the search line to the template. In general the ends of the search line will not map to the ends of the template. If fact only a sub-section of the search line may actually correspond to part of the template, with one or both ends of the search line lying off the end of the template. The beginning and end of the section of search line which maps onto the template will be denoted as ρ^{min} and ρ^{max} respectively. The positions on the template corresponding to the ends of the mapped section will be denoted at $\bar{\rho}^{min}$ and $\bar{\rho}^{max}$.

Since the I_t corresponds to the intensity pattern on physical objects viewed with a camera, there are constraints on the possible deformations which it can have undergone in the imaging process. These constraints can be expressed as the pdf, $\bar{\alpha}(\bar{r}, d)$, that a length δ (the discretisation interval), on the search line

will map to length d of the template. This pdf may vary with distance along the template. Mathematically, $\bar{\alpha}(\bar{r}, d)$ is given by:

$$\bar{\alpha}(\bar{r}, d) = \text{prob}\left((f_t(r) - f_t(r-\delta)) = d \mid \bar{r} = f_t(r)\right) \tag{3}$$

We assume $\bar{\alpha}(\bar{r}, d)$ and $\bar{\alpha}(\bar{r}+\bar{\delta}, d')$ are independent of each other, then the probability of a given mapping of the entire search line is given (discretely) by:

$$\begin{aligned} \text{prob}(f_t) = & \prod_{(-\rho_t/\delta) \le i \le (\rho_t/\delta)} \bar{\alpha}(\bar{r}, d) \\ \text{where} \quad & \bar{r} = f_t(i\delta) \\ & d = f_t(i\delta) - f_t((i-1)\delta). \end{aligned} \tag{4}$$

An example of $\bar{\alpha}(r, d)$, representing the case where the search line is a translated copy of the template, is:

$$\begin{aligned} \bar{\alpha}(a, b) &= 1, && \text{iff } b = \delta \\ &= 0, && \text{otherwise} \end{aligned} \tag{5}$$

Section 5 explains how $\bar{\alpha}(r, d)$ may be estimated in practice.

To find the *best* warp of the search line to the template we use the dynamic programming algorithm.

3 The dynamic programming algorithm

Dynamic programming works in a recursive fashion [5]. Considering each point r along the search line and $\bar{r}$ along the template in turn, the algorithm finds the probability that r corresponds to $\bar{r}$ by considering the set of possible partial mappings over $(-\rho_t, r-\delta)$. We denote the ***highest probability*** of r matching $\bar{r}$ by $P_t(\bar{r}, r)$.

We obtain $P_t(\bar{r}, r)$ by applying the following dynamic programming algorithm:

$$\begin{aligned} P_t(\bar{r}, -\rho_t) &= \bar{\phi}(\bar{r}, I_t(-\bar{\rho})), \quad -\bar{\rho} \le \bar{r} \le \bar{\rho} \\ P_t(\bar{r}, r) &= \max_{-\bar{\rho} \le \bar{k} \le \bar{\rho}} \bar{\phi}(\bar{r}, I_t(r)).\bar{\alpha}(\bar{r}, \bar{r}-\bar{k}).P_t(\bar{k}, r-\delta) \quad \begin{matrix} -\bar{\rho} < \bar{r} \le \bar{\rho} \\ -\rho_t < r \le \rho_t \end{matrix} \\ \mathcal{W}_t(\bar{r}, r) &= \arg \max_{-\bar{\rho} \le \bar{k} \le \bar{\rho}} (\bar{\alpha}(\bar{r}, \bar{r}-\bar{k}) P_t(\bar{k}, \bar{r}-\bar{\delta})) \end{aligned} \tag{6}$$

where the "policy function" $\mathcal{W}$ is defined for later use in recovering the most probable warp.

Before we can recover the warping, we need to find the most probable position on the template to which the end of the template mapped ($\bar{\rho}^{max}$). Since the search line may actually overlap the end of the template, we also need to find the position on the search line which maps to $\bar{\rho}^{max}$: we denoted this ρ^{max}. We find these positions by examining the probabilities $P_t(\bar{r}, r)$ of the warps which contain either the end of the search line or the end of the template (i.e. either

$r = \rho_t$ or $\bar{r} = \bar{\rho}$), and thus finding the warp with the highest probability. $\bar{\rho}^{max}$ and ρ^{max} are obtained using the equation:

$$(\bar{\rho}^{max}, \rho^{max}) = \arg\max_{(a,b)} \{ P_t(a,b) | ((a = \bar{\rho}) \wedge (-\rho_t \leq b \leq \rho_t)) \vee ((-\bar{\rho} \leq a \leq \bar{\rho}) \wedge (b = \rho_t)) \}$$

Once $\bar{\rho}^{max}$ and ρ^{max} have been found, $\tilde{I}_t(r)$ can be found by backtracking using the path implicitly stored in $\mathcal{W}_t(a,b)$:

$$\begin{aligned} \tilde{I}_t(r) &= I_t(f_t^{-1}(r)) \\ \text{where} \quad f_t^{-1}(r) &= \mathcal{W}_t(r+\delta, f_t^{-1}(r+\delta)) \\ f_t^{-1}(\bar{\rho}^{max}) &= \rho^{max} \end{aligned} \tag{7}$$

In order to use this algorithm estimates are needed for template intensity distribution $\bar{\phi}(r,\gamma)$ and warp probabilities $\bar{\alpha}(r,d)$. The following sections explain how these estimates may be obtained.

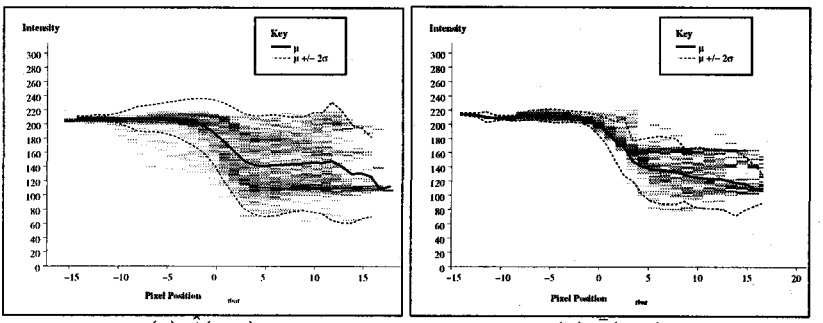
4 Estimating Intensity PDF's

We would like to estimate the match probabilities $\bar{\phi}(r,\gamma)$ from data obtained while tracking the target in a representative environment. This training data will include variations along the image line similar to those which we are likely to see once tracking using the intensity model. The training data can be collected by a bootstrap tracker (a tracker able to follow the target as it undergoes some limited motion[1]) tracking the target from time $t = 1$ to $t = T$. The training data will contain variations in I_t due to tracking error and scale changes in the object. This means that in order to obtain an reliable estimate for $\bar{\phi}(r,\gamma)$, each item of training data needs to be pre-warped, or transformed to remove these errors (see Figure 3). We perform this transformation by using the dynamic programming algorithm and a rough estimate $\hat{\phi}(r,\gamma)$ for $\bar{\phi}(r,\gamma)$. One method[2] for obtaining the estimate $\hat{\phi}(r,\gamma)$ is to align the training data based on the feature positions ν_t obtained using the bootstrap tracker. The estimated intensity pdf, $\hat{\phi}(r,\gamma)$, can then calculated using simple statistics.

Using $\hat{\phi}(r,\gamma)$ to estimate $\bar{\phi}(r,\gamma)$ We define $\hat{I}_t$ to be the warp of I_t using the dynamic programming algorithm and the estimates $\hat{\phi}(t,\hat{r})$ and $\hat{\alpha}(r,d)$ (see section 5). $\hat{I}_t(\hat{r})$ is the intensity at distance $\hat{r}$ along this warped search line and exists for $\hat{\rho}^{min} \leq \hat{r} \leq \hat{\rho}^{max}$. We can now obtain an estimate for $\bar{\phi}(r,\gamma)$ by

[1] Such a tracker may use image differencing, or a strong model of the targets motion to determine the target's position.

[2] This method assumes that only small changes in the scale of the target occurred during $1 \leq t \leq T$



(a) $\hat{\phi}(\bar{r},\gamma)$ (b) $\bar{\phi}(\bar{r},\gamma)$

Fig. 3. The effect of pre-warping on the distribution of the template. Dark regions on the graphs show regions of high probability. Overlaid on the templates are the mean (μ) and $\pm$ 2 standard deviations (σ). Graph (a) shows the estimate $\hat{\phi}(\bar{r},\gamma)$ obtained using the raw data. (b) shows the corresponding $\bar{\phi}(\bar{r},\gamma)$ obtained by pre-warping the training data using the estimate $\hat{\phi}(\bar{r},\gamma)$. Note how the template in (b) is much sharper and tighter than the estimate (a)—this is because the search lines in the *pre-warped* training dataset are better aligned with one another.

analysing the results of warping the training data to the rough template:

$$\bar{\phi}(r,\gamma) = \frac{1}{N_d(r)} \sum_{t=1}^{T} h\left(\hat{I}_t(r),\gamma,r\right)$$

$$\text{where} \quad h(\lambda,\gamma,r) = \begin{cases} 1, & \text{iff } (\lambda=\gamma) \wedge (\hat{\rho}^{min} \leq (r-\nu_t) \leq \hat{\rho}^{max}) \\ 0, & \text{otherwise} \end{cases}$$

$$N_d(r) = \sum_{t=0}^{T} g(t,r)$$

$$g(t,r) = \begin{cases} 1, & \text{iff } \hat{\rho}^{min} \leq r-\nu_t \leq \hat{\rho}^{max} \\ 0, & \text{otherwise} \end{cases}$$

Figure 3 shows the effect that the pre-warping has on the estimate of $\bar{\phi}(\bar{r},\gamma)$. Note that it is perfectly feasible to use the histogram based approach to storing pdf in this case. A 50 pixel long template will only need 32k of memory to store $\bar{\phi}(\bar{r},\gamma)$ in a histogram form (to an accuracy of $\frac{1}{256}$. If a similar approach was used to store the pdf of intensity in a 768 $\times$ 576 background model, it would require 108MB of memory!

5 Estimating stretching PDF's

We also need an estimate for $\bar{\alpha}(r,d)$. As with estimating $\bar{\phi}(r,\gamma)$ this can be done by analysing data collected with a bootstrap tracker from time $t=1$ to $t=T$. The function $f_t(u)$ proves to be particularly useful in estimating $\bar{\alpha}(r,d)$ – we can simply examine what length each δ section of search line mapped to on the

template and build a pdf from this. This pdf may vary along the length of the template.

$$\bar{\alpha}(\bar{r},d) = \frac{1}{N_D}\sum_{t=1}^{T}\sum_{r=-\rho_t+\delta}^{\rho_t} h\left(f_t(r), f_t(r) - f_t(r-\delta)\right)$$

$$\text{where} \quad h(a,b) = \begin{cases} 1, & \text{iff } (a=\bar{r}) \wedge (b=d) \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$N_D = \sum_{t=1}^{T}\sum_{r=-\rho_t}^{\rho_t} g(r)$$

$$g(a) = \begin{cases} 1, & \text{iff } a=\bar{r} \\ 0, & \text{otherwise} \end{cases}$$

Of course, in order to obtain $f_t^{-1}(u)$ used in the above equation we need to have already estimated $\bar{\alpha}(r,d)$. We get around this problem by supplying an initial estimate, $\hat{\alpha}(r,d)$, of $\bar{\alpha}(r,d)$—a Gaussian centered around $d=\delta$ with standard deviation 2δ.

Unfortunately, $\bar{\alpha}(\bar{r},d)$ will not be constant over time, but will vary depending on the scale of the target being tracked–the stretches necessary to fit the target when it is small in the image will be different from those when it is large. Fortunately as the occluding contour of the target is being tracked, an estimate of the scale of the object is available. This means that $\bar{\alpha}(\bar{r},d)$ can be adjusted depending on the expected scale of the object. Initial experiments suggest that simply scaling the d-axis of $\bar{\alpha}(\bar{r},d)$ by the relative scale of the target is sufficient.

6 Tracking

In order to test the algorithm we implemented an active contour tracker [3] and applied the feature search by dynamic programming. The tracker used in the experiments reported here is an estimator for a N span B-spline image curve of order 3 (quadratic) [6]. A given curve is represented as $\mathbf{x}_{s,t} = (x_{s,t}, y_{s,t})$ with length parameter s (the t subscript signifies that the curve's position and shape may alter over time). The coordinates $(x_{s,t}, y_{s,t})$ are given by the equation:

$$\begin{aligned} x_{s,t} &= B_s \mathbf{X}_t \\ y_{s,t} &= B_s \mathbf{Y}_t \end{aligned} \qquad 0 \le s \le N \tag{9}$$

where $\mathbf{X}_t$ and $\mathbf{Y}_t$ are stacked vectors of the X and Y coordinates of the B-spline's control points respectively. B_s is the standard B-spline basis function matrix [6].

The target is assumed to be described in the same form as the curve, with control points $\mathbf{X}_t$ and $\mathbf{Y}_t$ varying over time. The active contour tracker generates estimates of these over time, $\hat{\mathbf{X}}_t, \hat{\mathbf{Y}}_t$, based on measurements of the actual

position of the curve. We also define a shape template, $(\bar{\mathbf{X}}, \bar{\mathbf{Y}})$, which is the *average* shape of the target, and helps to stabilise the tracker[12]. The state of the tracker (position and velocity) at time t is denoted $\mathcal{X}_t$. We assume that this state evolves based on a 2^{nd} order dynamical model.

In order to estimate the state of the curve, measurements are made of its position at each time-step. These measurements are made by casting rays along normals $\hat{\mathbf{n}}_{s,t}$ to the estimated curve, and, simultaneously at certain points s along the curve measuring the relative position, $\nu_{s,t}$ of a feature[3] along the ray so that:

$$\nu_{s,t} = [\mathbf{r}_{s,t} - \hat{\mathbf{r}}_{s,t}] \,.\hat{\mathbf{n}}_{s,t} + v_{s,t} \tag{10}$$

where $v_{s,t}$ is a scalar noise variable, assumed Gaussian, that is taken as constant, both spatially and temporally. The measurement is defined along the normal as displacement tangentially is unobservable, the well-known aperture problem.

The tracking process is performed using a Kalman Filter based around the motion and measurement model. One important feature is that each state estimate is accompanied by an estimate of covariance. This enables a validation, or range gate $\rho_{s,t}$ to be defined [7] which allows measurements to be included only if they lie within a certain distance ($\rho_{s,t}$) of the predicted position. This distance $\rho_{s,t}$ changes over time as the tracker is more or less certain about its prediction. The range-gate mechanism of the Kalman filter means that the feature will only be considered if

$$-\rho_{s,t} \leq \nu_{s,t} \leq \rho_{s,t} \tag{11}$$

This means that we only need consider the segment of $\mathbf{n}_{s,t}$ which lies within $\rho_{s,t}$ of the curve. These segments of the normals form are the *search lines* of the active contour.

The grey-level intensity of the pixel at a position r along $\mathbf{n}_{s,t}$ will be denoted as $I_{s,t}(r)$, and the set of intensities along the whole search line as $I_{s,t}$. Note that $I_{s,t}(r)$ is the intensity of the pixel at location $\mathbf{x}_{s,t} + r\mathbf{n}_{s,t}$.

When using a gradient based feature search the tracker is able to track at 50Hz using an 8 span template on a Sun IPX. Unfortunately when tracking with the dynamic programming based feature search, this drops to 0.2Hz. This is because the complexity of the dynamic programming algorithm is $\bigcirc(\bar{\rho}^2\rho_t)$. Ways of improving the speed of the algorithm are being pursued, but in the mean time application is restricted to recorded video sequences.

Figure 1 shows a tracking problem. A target moves in front of a cluttered background. A gradient based tracker is distracted by the clutter and loses the target. When the foreground is incorporated the tracker is able to reliably detect the boundary of the target, and able to stay locked to it.

[3] This feature should represent the actual underlying position of the curve on the target. Traditionally high contrast edges have been used as "the feature", this paper proposes a different model based on the gray level profile of the target.

7 Conclusions

An algorithm has been proposed to improve the feature detection for active contours so that they can track robustly in a wider range of applications. The use of a statistical template and the dynamic programming algorithm enables the feature search to locate the target, even in the presence of heavy clutter. Results have been shown for hard tracking sequences which clearly demonstrate the possible improvements. Future work will address ways of reducing the computational burden of the algorithm to enable real time tracking. It will also investigate the effect which normalising the probabilities $P_t(\bar{r}, r)$ by the length of mapped search line has on the warps chosen by the algorithm.

References

1. A. Amini, S. Tehrani, and T. Weymouth. Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints. In *Proc. 2nd Int. Conf. on Computer Vision*, 95–99, 1988.
2. B. Bascle and R. Deriche. Region tracking through image sequences. In *Proc. 5th Int. Conf. on Computer Vision*, 302–307, Cambridge, MA. IEEE, IEEE Computer Society, June 1995.
3. A. Blake, R. Curwen, and A. Zisserman. Affine-invarient contour tracking with automatic control of spatiotemporal scale. In *Proc. 4th Int. Conf. on Computer Vision*, 66–95, Berlin. IEEE, IEEE Computer Society, May 1993.
4. T. Cootes, C. Taylor, A. Lanitis, D. Cooper, and J. Graham. Buiding and using flexible models incorporating grey-level information. In *Proc. 4th Int. Conf. on Computer Vision*, 242–246, Berlin. IEEE, IEEE Computer Society, May 1993.
5. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill, 1992.
6. J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics, principles and practice.* Addison Wesley, 2nd edition, 1990.
7. A. Gelb, editor. *Applied Optimal Estimation.* MIT Press, Cambridge, MA, 1974.
8. M. Kass, A. Witkin, and D. Terzopoulos. Snakes:active contour models. In *Proc. 1st Int. Conf. on Computer Vision*, 259–268, 1987.
9. S. Rowe and A. Blake. Statistical background models for tracking with a virtual camera. In D.Pycock, editor, *British Machine Vision Conference 1995*, volume 2, 423–432. BMVA, Sept 1995.
10. D. Terzopoulos and R. Szeliski. Tracking with Kalman snakes. In A. Blake and A. Yuille, editors, *Active Vision*, 3–20. The MIT Press, 1993.
11. P. Woods, C. Taylor, D. Cooper, and R. Dixon. The use of geometric and grey-level models for industrial inspection. *Pattern Recognition Letters*, 5(1):11–17, Jan 1987.
12. A. Yuille and P. Hallinan. Deformable templates. In A. Blake and A. Yuille, editors, *Active Vision*, 20–38. The MIT Press, 1993.