

On-line object tracking for colour video analysis

GIANCARLO IANNIZZOTTO¹ AND LORENZO VITA²

¹*G. Iannizzotto is with the Department of Mathematics, Faculty of Engineering, University of Messina. C.da Papardo, Salita Sperone, 98166, Messina - Italy. E-mail: ianni@ingegneria.unime.it*

²*L. Vita is with the Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, University of Catania, Viale A. Doria 6, 95025 Catania - Italy. E-mail: vita@diit.unict.it*

Abstract

Real-time object tracking is recently becoming more and more important in the field of video analysis and processing. Applications like traffic control, user-computer interaction, on-line video processing and production and video surveillance need reliable and economically affordable video tracking tools. It seems, however, that most of the available solutions are computationally intensive and sometimes require expensive video hardware, quite often without guaranteeing a suitable level of reliability. In this paper we present a new approach to real-time object tracking from colour video sequences. It relies on contours in order to track the shape, position and orientation of objects, without exploiting snakes or “traditional” active contours. A Closed-Loop control approach is adopted to enforce motion tracking stability, while a separate shape model is maintained, featuring a two-stage model and a *median filtering* technique to cope with temporary occlusions and noise. The system was tested in several different environments with different constraints, and gave very encouraging performance. Experimental results are reported and commented on.

1. Introduction

Tracking an object through a sequence of frames is the main task in several video analysis applications. The tracked object may translate and rotate in the 3D space: consequently, its projection on the image plane will undergo projective transformations causing substantial deformations in its 2D shape. The object may be partially and temporarily occluded by another object, or may gradually

disappear from the scene: in such cases only a part of the object will be visible and some way to *recall* its shape will be needed to track it until it is fully visible again. On the other hand, the object may change its real shape: for example, a man may raise an arm; this situation is extremely ambiguous, because it is necessary to decide whether the model made of the object (the man) should be modified, or whether the change in its shape is to be considered transitory (i.e. noise). An object may *split* into two or more objects: this occurs when the first image of the sequence shows a cluster of similar objects very close to each other. Depending on the technique adopted to determine the initial shape of the object to track, it is very likely that the whole cluster will be selected as a single object, until the objects it consists of move away from each other. It is therefore necessary to update the model of the object promptly, according to the changes in the image and an appropriate semantics.

Unlike off-line (batch) video characterisation, real-time video analysis is subject to strict time constraints (*deadlines*) which, if missed, may lead to quality of service degradation (soft deadlines) or even to some kind of failure (hard deadlines). Applications cover autonomous mobile systems (robots, etc), automatic surveillance, real-time network traffic control, human-computer interaction and content-based, on-line, video browsing and retrieval.

Most applications in real-time image analysis exploit very powerful and sometimes even dedicated hardware. In this paper we present a real-time object tracking framework which exploits different families of features according to the particular application, in order to track moving objects. The algorithm tracks the position, orientation and shape of an object through a sequence of frames, dealing with a moderate amount of occlusion and perspective transformation. The main features of the approach presented are *a low degree of complexity* and a high frame rate, even when running on a low-cost PC-based platform. By applying the prediction - correction paradigm and multi-feature observation (exploiting multiple features to locate point-to-point correspondence), stability can be enforced, thus providing reliable tracking for several applications. The framework was tested in a number of cases, as a human face tracker, a hand tracker and a traffic monitoring system. In the next two sections we will give some of the reasons for the research presented in the paper and a brief survey of related work, while in Section 4 our approach is described. Experimental results are provided in Section 5 for some of the applications referred to, and Section 6 concludes the paper.

2. Real-Time object Tracking

A considerable amount of research has been done in past and recent years into object tracking. The early work was mainly based on tracking artificial features, such as coloured spots or lights placed *ad hoc* in the scene and on objects to be tracked. The applicability of this approach was obviously quite limited, but according to the number and quality of the markers used, the results were often

satisfactory. Lately, the availability of more computational power at a lower cost has induced researchers to develop algorithms with a wider applicability, capable of tracking *natural features* in the images, i.e. features extracted directly from the objects in the scene. Basically, three different approaches to natural feature tracking have emerged during the last few years.

- Low-Level tracking identifies low-level features and tracks them through a sequence of frames, if necessary adapting the set of features according to any modifications which may have occurred in the scene during the process. Different techniques have been developed for feature selection [30] [36] and accurate tracking, most of which are not especially devised for real-time applications [37] [2] [25]. Due to the lack of structural information, exploiting only low-level features often results in poor tracking, since occlusions, noise, and similarities between feature values from different points can mislead the tracker [24] [30]. A very good feature selection and a robust tracking algorithm can result in accurate and stable tracking, at the price of a higher computational cost (hardly suitable for real-time tracking). For this reason, most algorithms exploit some kind of higher-level coherence, by grouping low-level tracked features which show similar (coherent) motion and monitoring how the resulting regions evolve through the frame sequence. Regions which show internal homogeneous and coherent motion are labelled as moving objects in the image sequence, and merged together if they are contiguous and their motion parameters are similar [26] [9] [31]. However, the combination of low-level feature extraction, selection and clustering can amount to a very slow algorithm, unlikely to be applicable to real-time processing, unless very fast, expensive and, in some cases, dedicated hardware is exploited [31] [6].
- Strictly Model-Based feature tracking is a possible, application-specific, solution to the problem of needing both high-level features and fast computation. This approach is based on the availability of a predefined model of the object to be tracked, expressed in terms of its low-level features. The model is matched against each frame, and the combination of rigid motion and/or similar transformations which results in the best match is found [21] [10] [13]. The need for predefined models is the main weakness of this approach, since a suitable model is often not available.
- High-Level feature tracking usually relies on some fast initialisation procedure which creates a “startup” high-level model of the object to be tracked, by processing the first few frames. Subsequently the model is matched with each frame and adapted to any changes occurring in the frame sequence. The changes are detected by comparing the model with its best match in each frame, thus obtaining “temporary local” changes. Temporary local changes are then integrated in a time window, thus obtaining a “stable” change, which results in a modification of the model [27]. This approach is highly sensitive to initialisation, since incorrect initialisation will probably

result in very poor tracking. Moreover, this initialisation process usually needs a *segmentation* phase, which is generally known to be an open issue but can be solved, provided that some conditions on the frame sequence are met [9].

Our research concentrates on real-time object tracking, i.e. the tracking of *objects* in a sequence of frames. We can define an object as an image area with a well-known shape and some “unifying” characteristic, common to all the points belonging to the area. An object is always surrounded by a border, a *contour*; this contour divides the object from other objects, and its 2D representation is always a *closed line*. Objects can be tracked through a sequence of frames by matching their characteristics in each frame; from this point of view, object tracking belongs to the high-level feature tracking class.

3. Contour-based object tracking

Several techniques for real-time object tracking are based on the ability to detect and track the contour of objects: this is due to the belief that the main information about the shape of a 2D object lies in its contour. Moreover, since 2D contours are closed curves, it is possible to represent them in a 1D fashion, thus reducing the computational complexity of the algorithms involved and the size of the resources needed for shape comparison [1] [29] during the tracking process. Other approaches to object tracking rely on blob tracking [39]: since they need to match 2D areas instead of contour lines, these approaches can turn out to be computationally expensive and they require fast hardware.

Shape-tracking techniques relying on *active contours* have received a large amount of attention since the late 80’s. Active contours are models representing open or closed curves which, having a very high capacity for morphological adaptation, can accurately reveal the contours of any shape. These models are called *active* because they automatically respond to specific characteristics of the image points, changing their shape accordingly. An active contour can, for example, respond to the edgeness values of points in an image by changing its shape till it fits all the points with the (locally) highest edgeness values (*local edgeness maxima*). A particular type of active contour is the *snake*: it responds both to a quantity called *external energy*, related to the local image properties (image gradient, brightness, contrast, etc) and to a quantity called *internal energy*, related to specific internal laws ruling its shape and way of deformation [19] [20].

There are two main reasons for exploiting active contours for real-time object tracking:

1. Since the objects to be tracked can change in shape, position, orientation and distance from the camera, and since the image perceived by the vision system is substantially a projection of the 3D scene onto a 2D plane, it is necessary to adopt a dynamic model to represent the shapes of the objects;

2. Active contours have proven to be stable and reliable, and it is also relatively easy to implement an active contour with appropriate properties to deal with almost any particular kind of shape [23] [34].

The most popular active contours rely on iteratively fitting some kind of parametric curve onto the contour points [8] [19]. This fitting process slows the whole tracking process and is likely to introduce some unwanted smoothing in the retrieved contour. Other active contours, on the other hand, exploit local properties, integrating them along the whole contour to obtain a global behaviour [20] [5]. This approach, however, implies some energy minimisation in order to determine the correct motion for the contour, and this energy minimisation implies calculating an energy value for each point of the contour.

Several discrete implementations of active contours have been developed, but their properties and behaviour have only been demonstrated in the continuous case, so their behaviour is sometimes not very clear [5] [38]. In order to avoid a high level of computational complexity due to the minimisation process, different active contours have been developed, which do not rely on any energy or functional minimisation. The underlying idea is that it should be possible to develop a discrete active contour which is not derived from any continuous one. This can be accomplished by aggregating a chain of moving points acting as autonomous agents in the image and looking for object edges. The behaviour of the agents should be well-defined and should be implemented in a fast way, in order to speed up the motion; look-up tables, for example, can be exploited for fast decision making [14]. An example of discrete active contours for fast object tracking can be found in [15], but this algorithm currently only relies on image gradients as pixel features.

Active contours alone cannot, however, reliably deal with real-world object tracking applications: without some knowledge about the motion of the object, it is extremely likely that the tracker will *lose contact* with the tracked object [1]. Since in most cases some hypotheses or, at least, a set of constraints can be stated about object motion, a suitable way should be found to exploit such knowledge in the tracking algorithm. This issue has been thoroughly addressed in the recent past, mainly exploiting techniques from control theory and statistics [4]. An important contribution in this respect is perhaps the idea of exploiting the concept of *closed-loop control* to correct the output of a *predictive model*, used to predict the behaviour of the tracked object. The predictive model is basically a mathematical, usually linear, model embedding as much information as possible about the constraints and hypotheses possibly stated on object motion. The whole problem is usually dealt with in a statistic framework, in order to take into account the uncertainty related to the observation process (observation density distribution), as well as the need for some flexibility in the models (prior distribution). Very interesting and thorough studies have been conducted in the case of Gaussian [1] and non-Gaussian [16] observation density distribution.

Suitable mechanisms should also be introduced to cope with objects which

may undergo major changes in their shape and with composite objects (i.e. objects composed by other objects, which may be occluded or in some way not perceivable in a number of frames). This problem has not been completely addressed yet (though some significant studies have been conducted, for example, on the separability of rigid and non-rigid face motion [3]).

In the next section we present a shape tracking technique which does not rely on any traditional active contour: it is based on a *radial representation* of the object contour [7], which is regularly updated according to changes in the object shape, exploiting a suitable mechanism to cope with noise and moderate occlusions and perspective deformations. Since the algorithm performs, at each iteration (i.e. for each frame), just a few comparisons and integer calculations for each point of the tracked contour, its complexity is inherently low and the tracking process is *fast*. This allows us to use it for real-time video analysis.

4. The proposed solution

In our work we followed the prediction-measure-assimilation paradigm characteristic of process control literature but recently adopted in real-time visual tracking, as described in [4]. As shown in Fig. 1, the tracking system is basically composed of two modules, the predictor and the controller. The two modules are connected in a closed-loop fashion, so the controller can modify the parameters of the predictor, according to the measure, in order to enhance the tracking performance and take into account possible changes in object motion. The process of correcting the prediction according to the observed measures is often called *Assimilation*.

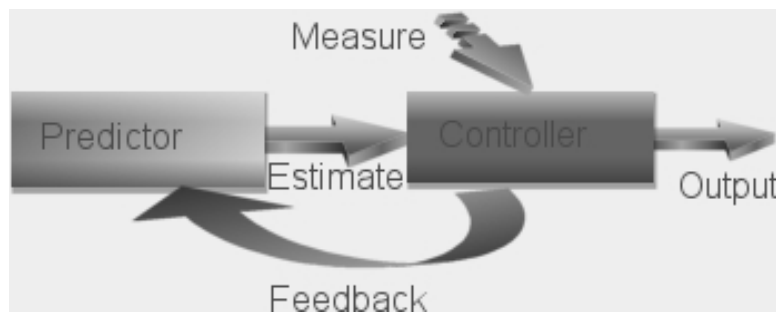


Figure 1: Closed-Loop tracking paradigm

The tracker needs to be initialised before it can start following the object. The initialisation is usually accomplished by a segmentation algorithm [35], often user-driven [11]; it has the twofold purpose of locating the target and building a representation of its shape. The segmentation technique should be chosen according to the particular application, since no general-purpose approach has as yet been identified for image segmentation. The main issue is probably the choice

of the image properties to be exploited to extract the image regions which are to be considered as objects. Discussing segmentation techniques is beyond the scope of this paper: we currently rely on a technique described in [14], and the criteria adopted for choosing the image properties are briefly described in Section 5.

In our algorithm we exploit a radial representation of the shape. As shown in Fig. 2 (left), the centre of mass of the contour extracted by image segmentation (during the initialisation stage) is calculated. Then, with uniform spacing, a sequence of radii are driven from the centre of mass to intercept the object contour. The angle ϑ between two consecutive radii is constant, and the choice of this angle is a trade-off between shape resolution and tracking performance. Multi-scale solutions are exploitable, in order to realise an approximate algorithm which iteratively refines the shape model according to the time available before the deadline expires (i.e., according to the time available before a new frame has to be processed). This deadline can be adapted according to prediction of the scene dynamics, i.e. according to the speed of the changes in the scene. Currently, a constant angle solution is adopted in our prototype system, but multi-scale algorithms are under development. The distance, from the centre of mass and along each radius, of the intercept point on the outer contour of the object is recorded, and the resulting vector is taken as a representation of the shape of the object. In Fig. 2 (right), the radial representation is superimposed on the outer contour of a face. The points of the contour were sub-sampled to make the figure more readable.

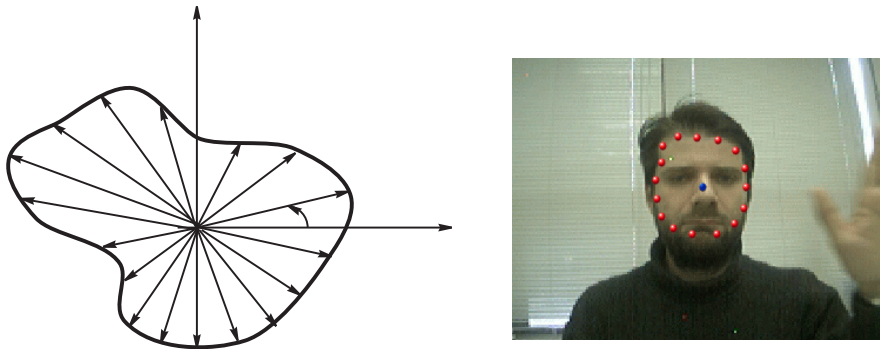


Figure 2: Scheme of the radial representation of a shape

4.1. A motion model and a shape model

After retrieving a representation of the shape and position of the contour, the predict-measure-assimilate loop should be applied, but since its computational cost is $O(N_X^3)$, where N_X is the size of the status vector (the vector to be

tracked), simply applying this paradigm to the whole shape model results in a very slow operation. We therefore decided to maintain two models, separating the **shape model** from the **motion model**.

The shape model is the radial representation of the contour:

$$SM_i = (sm_1^i \quad \dots \quad sm_n^i) \quad (1)$$

For reasons which will be clear later in this section, we actually maintain three shape models: SM_0 is the shape model vector which was returned by the initialisation process; SM_i is the current shape model at the discrete time instant i ; MSM_i is the **Main Shape Model** at the same discrete time instant i .

The motion model is a status vector representing the position, planar orientation and size (scaling factor with respect to the initial size) of the object at the discrete instants i and $i - 1$:

$$X_i = \begin{pmatrix} x_{i-1} & y_{i-1} & \theta_{i-1} & f_{i-1} \\ x_i & y_i & \theta_i & f_i \end{pmatrix} \quad (2)$$

In Eqn. (2), (x_i, y_i) is the position of the centre of mass of the tracked object (in image co-ordinates); θ_i is the relative orientation with respect to the initial orientation (i.e. the orientation returned by the initialisation phase); f_i is the scaling factor, i.e. the relative size with respect to the size returned by the initialisation step.

The centre of mass is calculated at each iteration while building the shape model SM_i , as the centre of mass of the sequence of n points composing the contour:

$$\begin{aligned} x_c &= \frac{1}{n} \sum_{j=1}^n x_j \\ y_c &= \frac{1}{n} \sum_{j=1}^n y_j \end{aligned} \quad (3)$$

The relative orientation θ_i can be determined by *aligning* the shape-model vector SM_i with the one retrieved from the initialisation step, SM_0 . Let us define the alignment distance function $adf_k(A, B)$ of two vectors A and B of size n , as follows:

$$\begin{aligned} adf(A, B, k) &= \sum_{j=1}^n |a_j - b_{((i+k) \bmod n)}| \\ k &= 0, 1, \dots, n - 1 \end{aligned} \quad (4)$$

This distance is function of the two vectors and of the circular shifting parameter k . Then, the two vectors A and B are aligned if the shifting parameter k minimises the function $adf(A, B, k)$. Since the angle between two consecutive

rays in the shape model (i.e. the sample step ϑ) is constant, this means that, provided that this sampling step is small, the angle θ_i can be approximated by:

$$\theta_i \approx k\vartheta \quad (5)$$

The scaling factor f_i can be determined as the ratio of the energies associated to the shape representation vectors SM_i and SM_0 :

$$f_i = \frac{\sum_{j=1}^n sm_j^i}{\sum_{j=1}^n sm_j^0} \quad (6)$$

where we have used the notation (1).

The model adopted for prediction is a linear, second-order auto-regressive process (AR), and its parameters were determined manually according to the particular application (see Fig. 5).

Let us now go back to the shape models and explain their meaning. As is probably quite evident by now, the shape model SM_0 is maintained in order to have a fixed point to calculate the relative orientation and scaling factor. The model SM_i represents the current *measure* for the object shape and is used to calculate the current measure for the motion status as well. The *Main Shape Model*, MSM_i , has a slightly less evident meaning. The target is to cope with shape variations, which can be produced by actual variations in the shape of the object, as well as by occlusions, noise, shadows and perspective effects. If the cause is noise or some temporary occlusion, we need to *filter out* those changes; but if they are caused by long-lasting occlusions or actual changes in the object's shape, we need to take them into account, changing our shape model. Similarly, perspective deformations should be taken into account as actual changes in the shape of the object. Last but not least, changes due to the appearance of new parts of the object, previously not visible due to some occlusion, should also be taken into account, changing the shape model accordingly. Several previous works exploited *similar transformations* [4] [28], but the drawback of this approach is that it cannot cope with occlusions and the appearance of new object parts, which produce deformations not compatible with similar transformations. Moreover, calculating the similar transformations is computationally intensive, since it requires the solution of a linear system of a usually large number of equations*. We therefore decided to handle all of the shape changes in the perceived object as actual changes in the object: to cope with temporary occlusions and noise, we implemented a filtering algorithm which removes spotty noise and short-lasting, non-smooth changes, most likely caused by temporary occlusions or noise.

*Theoretically, only 6 equations should be enough to solve the system, since 6 similar transformation parameters are sought. But to obtain reliable results a larger number of equations is usually required, and some robust regression method should be adopted to reduce the effect of *outlier points* in the linear system [17]

4.2. Filtering shape changes

The Main Shape Model is our actual shape model, while the Temporary Shape Model is only used to calculate the motion parameters and to update the Main Shape Model, according to equation (7) and the algorithm described below and outlined in Fig. 3:

$$msm_j^i = \underset{\nu=1\dots m}{median} (sm_j^\nu) \quad (7)$$

This algorithm is basically an application of the **median filter** to the shape model. The median filter [12] is a non-linear filter commonly used to reduce the effect of spotty noise in digital signals. Given a 1-D signal represented by its sample vector V of length n , a window of size m is set to slide on V , centring on each of the elements of V , so that for every element of V , a “neighbourhood” W of size $m < n$ is identified. The elements of W are then sorted, and the element of V that is the centre of the neighbourhood is replaced with the median value of W .

As is well known, the median M of a set of values is such that half the values of the set are less than M and half are greater than M .

We maintain a “buffer” of temporary shape models, arranged as a circular queue, with a size of m . At every time instant i , this buffer contains the temporary shape model SM_i and the temporary shape models from the previous $m - 1$ instants. At each iteration, the temporary models stored in the buffer are aligned with the Main Shape Model (this is a fast operation, since the values of θ are known for both the temporary models and the Main Shape Model) and the Main Shape Model is updated, replacing each of its elements with the median of the corresponding elements in the temporary models. This basically means that each element in the Main Shape Model is replaced with the median of the values that the corresponding point in the temporary model has taken in the last m discrete time intervals. As regards the effect of changes in the temporary models on the Main Shape Model, a modification in one element will affect the Main Shape Model only if its duration is at least equal to $m/2$ discrete instants, as can easily be proved.

```

Repeat  For each shape model stored in the buffer:
        Align and scale the model according to the main model;
Repeat  For each point in the main model:
        Pick the corresponding points from the stored temporary models;
        Sort them according to their displacement from the one
            [in the main model;
        Pick the median value as the new point value in the main model.
```

Figure 3: Main Shape Model Update algorithm

Gaussian distribution, a zero mean and respective variances of B_i and R_i . The variance of X_i is P_i .

Prediction-Assimilation paradigm	
	$\tilde{X}_i = A_{i-1}X_{i-1} + w_{i-1}$ <i>Prediction</i>
	$Z_{i-1} = H_{i-1}X_{i-1} + v_{i-1}$ <i>Observation model</i>
w_i and v_i have a zero mean and variances of B_i and R_i	
X_i has a variance of P_i	
	$\tilde{P}_i = AP_{i-1}A^T + BB^T$ <i>Riccati eqn.</i>
	$K_i = \tilde{P}_{i-1}H_i^T \left(H_i\tilde{P}_iH_i^T + R_i \right)^{-1}$ <i>Kalman Gain</i>
	$\hat{X}_i = \tilde{X}_i + K_i \left(Z_i - H_i\tilde{X}_i \right)$ <i>Assimilation</i>
	$P_i = (I - K_iH_i) \tilde{P}_i$

Figure 5: Prediction - Assimilation algorithm

The performance of the Kalman controller [32] described above is closely related to the hypothesis that both the noise vectors and the status vector have a Gaussian distribution. This hypothesis does not always hold, for example when the background is cluttered and contains points with features very similar to those of the object contour. In these cases, other approaches to the problem of prediction control can be exploited, like the CONDENSATION filter described in [16]. Our algorithm is basically independent of the predictor-controller used. In our current implementation, both Kalman and CONDENSATION filters were tested, and while the Kalman filter is faster, the CONDENSATION filter gave slightly better results for cluttered backgrounds. The results presented in this paper were obtained using the Kalman filter.

4.4. “Measuring” the object shape

How can we build a temporary shape model at each frame? So far we have only dealt with shape initialisation (which, being based on image segmentation, is intrinsically slow and cannot be repeated for each frame), and shape filtering, i.e. building the Main Shape Model. The algorithm for building the temporary shape model is outlined in Fig. 6: a very simple active contour is placed in the new frame (frame i), assuming the position, orientation and scaling factor produced by the motion model described above. The shape of the active contour is the shape of the Main Shape Model. A point of the active contour is only allowed to move along a radius connecting the centre of mass of the contour with the point itself. The point, then, starts scanning the radius, looking for the position

which minimises the following (mis)matching function:

$$mf(x, y, x_p, y_p) = \sum_{h=1}^{N_f} (c_h |f_h(x, y) - f_h(x_p, y_p)|) \quad (8)$$

where, with obvious notation, (x_p, y_p) are the co-ordinates of the point in the Main Shape Model and (x, y) are the co-ordinates of the currently scanned position along the radius.

The *image features* f_h are the image properties chosen to measure the degree of similarity of two points in the image: according to the particular application, different features can be used. In this work, we exploit colour information on either RGB or HSV colour spaces according to the application[†]. C_h is a positive weight associated with the feature h .

Though substantially neglected for a long time, colour provides powerful information for several image analysis applications. In the last 10 years colour matching techniques have gained a progressively increasing momentum due to research in content-based image indexing [33] and human face/hand detection and recognition [18]. Recently, colour has also been successfully exploited for real-time segmentation and tracking [22].

The problem of colour constancy is well-known and has not yet been solved; we addressed it by conducting most of our experiments in the HSV colour space, and neglecting the value related to pixel brightness. This is not considered a definitive solution, since for pixels with a very low intensity value hue becomes very noisy, while with a very low saturation, hue is not defined. This means that the performance of our tracker is in some way related to the quality and stability of illumination. We have not yet performed any quantitative evaluations of this relationship.

- | |
|---|
| <ul style="list-style-type: none"> • Set the active contour in the new frame to position, rotation and scaling factor given by the motion model; • Let the active contour adapt to the image; • Record the new shape in the temporary shape model; • Record position, orientation, and scaling factor in the observation (measure) vector Z_i; • Calculate R_i |
|---|

Figure 6: Shape measure algorithm

The points of the active contour never scan the whole radius: a size S_i for

[†]It should be pointed out that conversion from one colour space to the other does not add any substantial information, so it has not yet been proved, in general, that changing colour space can significantly improve the matching accuracy.

the scanning interval is defined, at each frame, according to the norm $\|\tilde{P}_i\|$ of the status variance predicted by the algorithm in Fig. 5. We used the following formula:

$$S_i^2 = \kappa \cdot \text{tr} \left(\tilde{P}_i \right) \quad (9)$$

where κ was determined manually and the norm of the matrix \tilde{P}_i was calculated as its **trace**.

A point in the active contour can stop moving either because it has found an optimal position (minimising mf) or because it has entirely covered its scanning interval. When all the points in the active contour have stopped, the new position of the points is saved in the temporary shape model (as distances from the new centre of mass).

Finally, the variance matrix R_i is calculated comparing the new temporary shape model with the Main Shape Model and determining the error for the position of the centre of mass, rotation and scaling factor. If it is greater than a fixed threshold $MaxVar$, then the whole measure is invalidated and a new initialisation is invoked.

5. Experimental results

The tracking system presented was implemented in C language on a Windows-based PC platform, and several tests were performed. For the specific purposes of this paper, to demonstrate the effectiveness of our approach, we carried out three different tests. The first test was performed on a video downloaded from the Internet, featuring an ordinary road with a bus approaching the camera (see Fig. 7). The scene is substantially cluttered, and a severe occlusion appears. Simply applying colour-based tracking would not give satisfactory results, since the colours of the occluded parts of the bus are different from the non-occluded parts. We obtained the results shown here by exploiting both colour and gradient: the image features used in equation (8) are colour (R,G,B) and the result of the Sobel gradient operator. Since the gradient usually has a high value on the border of an object, it is a useful feature for contour tracking. This is the only test sequence which was not grabbed directly and processed in real time. Its length was about 500 frames.

The hardware configuration used to capture and process the next two test sequences was a low-end PC with a Pentium II CPU and Windows98. The camera is a webcam capable of capturing colour video sequences of 320×240 pixels at 30 frames per second.

The *hand test sequence* (see Fig. 8) shows the performance of the tracker while tracking a user's hand: the sequence lasts about 600 frames, and the user is moving his hand almost naturally. The size of the buffer for the temporary shape model is set to 5 frames, thus fixing the minimum duration for a meaningful change in the hand shape at 3 frames. Since the sampling rate is about 30 frames per second, this means that a gesture should last at least 0.1 seconds to be



Figure 7: The *Bus* test sequence

correctly tracked. Faster gestures can only be correctly tracked in motion but not in shape. The image features used for equation (8) are the hue and saturation components in the HSV colour space.

Fig. 9 shows how the tracker reacts to occlusions: the user is waving his hand in front of the camera, thus occluding the tracked object (his head). In this case, since the probability of shape changes for the tracked object is definitely low (only motion parameters are substantially affected by head movements), we set the buffer size to 19, thus fixing the minimum duration for meaningful shape changes at 10 frames. The sampling rate is 25 frames per second, so the maximum duration for the occlusion to be ignored is 0.4 sec. After this duration, the occlusion will progressively affect the shape model. In the example shown, the occlusion is considerably faster and does not sensibly affect the shape model. This sequence was about 590 frames long.

The approach presented naturally has some weaknesses. The main problem



Figure 8: The *hand* test sequence

to be pointed out is related to the size of the buffer for the temporary shape model. Since it is currently fixed in advance, it does not adapt to the scene. In Fig. 10, a typical effect is shown: the tracked object, the user's hand, changes its shape very rapidly, while the size of the buffer is set to 19, so the minimum duration for a change to be detected is 10 frames (about 0.4 sec.). The result is that, though the motion is still tracked, the shape is definitely missing. Reducing the size of the buffer will solve the problem (see Fig. 11), but the tracker will be more sensitive to occlusions.

6. Conclusions

In this paper, an algorithm for real-time object tracking from colour video is presented. The algorithm can be used to track objects which change in shape and size, for example due to projective transformations. Its current implementation in "C" language performs at about 25 frames per second. Very encouraging results have been obtained with both outdoor and indoor scenes, and no particular lighting conditions are needed. A two-stage shape model has been introduced, and some techniques from control theory have been exploited in order to enforce



Figure 9: The *occlusion* test sequence

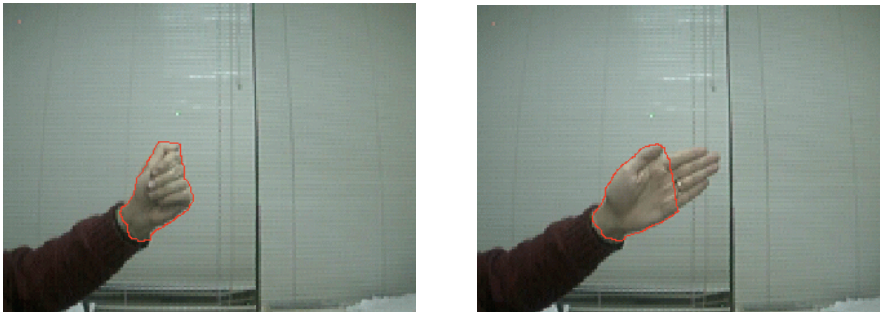


Figure 10: The *fault* test sequence

tracking stability. Experimental results have been provided to demonstrate the performance of the algorithm. It is currently being tested in a human-computer interaction environment, as a person-tracking system. Other applications are in the area of traffic monitoring and on-line video editing and processing, for example as an editing tool to track objects for on-line video production.

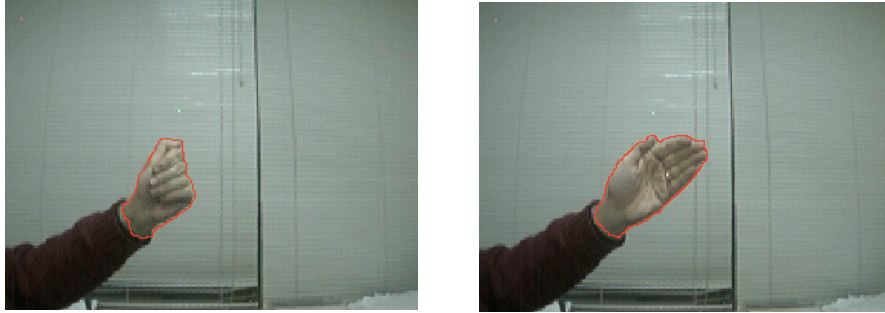


Figure 11: The *Correct-timing* test sequence

References

- [1] A. Zisserman, A. Blake, R. Curwen. A framework for spatio-temporal control in the tracking of visual contours. In C. M. Brown and D. Terzopoulos, editors, *Real-Time Computer Vision*, pages 3–33. Cambridge University Press, 1994.
- [2] A. Azarbayejani and A.P. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):562–575, June 1995.
- [3] B. Bascle and A. Blake. Separability of pose and expression in facial tracing and animation. In *ICCV98*, pages 323–328, 1998.
- [4] Andrew Blake and Michael Isard. *Active Contours*. Springer-Verlag, 1998.
- [5] L.D. Cohen and I. Cohen. Finite-element methods for active contour models and balloons for 2-d and 3-d images. *Transactions on Pattern Analysis and Machine Intelligence*, 15:1131–1147, 1993.
- [6] R. Cucchiara, M. Piccardi, and N. Scarabottolo. Real-time detection of moving objects. In *Proc. of ICIAP99*, 1999.
- [7] E. R. Davies. *Machine Vision, 2nd Edition*. Academic Press, 1997.
- [8] M. Flickner, H. Sawhney, D. Pryor, and J. Lotspiech. Intelligent interactive image outlining using spline snakes. In *Proceedings of 28th Asilomar Conference on Signals, Systems, and Computers*, 1994.
- [9] Hironobu Fujiyoshi and Alan Lipton. Real-time human motion analysis by image skeletonization. In *Proc. of the Workshop on Application of Computer Vision*, October 1998.
- [10] D.M. Gavrila and L.S. Davis. Towards 3-d model-based tracking and recognition of human movement: a multi-view approach. In *Int. Workshop on face and Gesture Recognition*, Zurich, 1995.
- [11] D. Geiger, A. Gupta, L.A. Costa, and J. Vlontzos. Dynamic programming for detecting, tracking, and matching deformable contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):294–302, March 1995.
- [12] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.
- [13] G.D. Hager and P.N. Belhumeur. Real-time tracking of image regions with changes in geometry and illumination. In *Proc. of IEEE CVPR96*, 1996.

- [14] G. Iannizzotto and L. Vita. Fast and accurate edge-based segmentation with no contour smoothing in 2-d real images. *IEEE Trans. on Image Processing*, July 2000.
- [15] G. Iannizzotto and L. Vita. Real-time shape tracking with movels and affine transformations. In *ICIP2000*, Vancouver, Canada, September 2000.
- [16] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *IJCV*, 29(1):5–28, August 1998.
- [17] R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision*. McGraw-Hill, 1995.
- [18] M.J. Jones and J.M. Rehg. Statistical color models with application to skin detection. In *CVPR99*, pages I:274–280, 1999.
- [19] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–333, 1988.
- [20] K.F. Lai and R.T. Chin. Deformable contours: Modeling and extraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1084–1090, November 1995.
- [21] Y. Li, S.D. Ma, and H. Lu. Human posture recognition using multi-scale morphological method and kalman motion estimation. In *ICPR98*, page PRP1, 1998.
- [22] S.J. McKenna, Y. Raja, and S. Gong. Tracking colour objects using adaptive mixture models. *IVC*, 17(3/4):223–229, March 1999.
- [23] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation trough physics-based synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15 (6):580–591, 1993.
- [24] U. Neumann and Suya You. Natural feature tracking for augmented reality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:53–64, March 1999.
- [25] N. Paragios and G. Tziritas. Adaptive detection and localization of moving objects in image sequences. *SP:IC*, 14:277–296, February 1999.
- [26] G.S. Pingali, Y. Jean, and I. Carlbom. Real time tracking for enhanced tennis broadcasts. In *CVPR98*, pages 260–265, 1998.
- [27] Y. Raja, S.J. McKenna, and S. Gong. Tracking and segmenting people in varying lighting conditions using colour. In *AFG98*, page Tracking and Segmentation of Moving Figures, 1998.

- [28] A. Psarrou S. Gong, S. McKenna. *Dynamic Vision*. Imperial College Press, 2000.
- [29] Brian Scassellati, Sophoclis Alexopoulos, and Myron Flickner. Retrieving images by 2d shape: A comparison of computation methods with human perceptual judgments. In *SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 2185, pages 2–14, February 1994.
- [30] J. Shi and C. Tomasi. Good features to track. In *CVPR94*, pages 593–600, Seattle, June 1994.
- [31] S.M. Smith and J.M. Brady. Asset-2: Real-time motion segmentation and shape tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):814–820, August 1995.
- [32] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis and Machine Vision*. Chapman & Hall Computing, 1993.
- [33] M.J. Swain and D.H. Ballard. Color indexing. *IJCV*, 7(1):11–32, November 1991.
- [34] R. Szeliski and D. Terzopoulos. Physically-based and probabilistic modeling for computer vision. In *SPIE Proceedings of: Geometric Methods in Computer Vision*, volume 1570, pages 140–152, July 1991.
- [35] M. Takahata, M. Imai, and S. Tsuji. Determining motion of non-rigid objects by active tubes. *ICPR*, A:647–650.
- [36] C. Tomasi and Takeo Kanade. Shape and motion from image streams: a factorization method - part 3 detection and tracking of point features. Technical Report CMU-CS-91-132, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, April 1991.
- [37] T. Tommasini, A. Fusiello, E. Trucco, and V. Roberto. Making good features to track better. In *CVPR98*, pages 178–183, 1998.
- [38] D.J. Williams and M. Shah. A fast algorithm for active contours. In *ICCV90*, pages 592–595, 1990.
- [39] C.R. Wren, A. Azarbayejani, T.J. Darrell, and A.P. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.