LOW-LEVEL TRACKING OF MULTIPLE OBJECTS

L. A. Sechidis, P. Patias, V. Tsioukas

The Aristotle University of Thessaloniki, Department of Cadastre Photogrammetry and Cartography
Univ. Box 473, GR-54006, Thessaloniki, Greece
sechidis@hypernet.hyper.gr, tsioukas@egnatia.ee.auth.gr, patias@topo.auth.gr

Commission III, WG 5

KEY WORDS: Digital Photogrammetry, tracking, video, background estimation

ABSTRACT:

Tracking of multiple objects from 'real world' is one of the most complicated problems in computer vision since the behaviour/motion of these objects is unpredictable and cannot be assumed. Tracking is also "application depended" task; there is no one general tracking methodology for solving all tracking problems. Therefore, different algorithms and procedures are needed and used for different applications. However, it is possible to rank tracking applications in two general categories: the ones that need a moving camera and the ones where a stable camera is enough. This paper deals with the second category and the technique of the "closed-world" for object tracking. It defines the lower level problem, the possible circumstances of the objects movements and the algorithms to solve some of the tracking situations. The obtained experimental results are pretty promising, approaching 100% hit rate, even in cases of two-object collision. In three-object collision occasional re-initialisation is required.

1. INTRODUCTION

Object tracking from "real-world" is one of the most complicated problems in computer vision. Projected objects size and radiometric values change from frame to frame, making almost impossible the use of a description template. Also, their motion is unpredictable. They can move rapidly to any direction; objects can also move isolated, together with other objects or been hidden for some time before they reappear in the tracking scene.

In this paper, a tracking algorithm of multiple, non-rigid objects, that uses a stable camera, is described. The algorithm is based on the "closed-world" assumption (Intille 1994, Intille et al 1995,1996) but, contrary to this technique, it does not use any contextual information in order to keep the algorithm as more general as it can be.

2. ESTIMATION OF THE BACKGROUND

Extracting the background image from sequences of frames is a very important task in order to help tracker detect motion. This task is repeated from time to time in order to incorporate any changes in the illumination of the tracking scene. There are several methods used to extract the background image from a sequence of frames but three are the most popular. These are based on statistical characteristics on the pixels of the frames: mean, median and highest appearance frequency methods. In all methods, every pixel of the background image is separately calculated using the mean or the median or the highest appearance frequency value from the series of frames.

Each method has its advantages and disadvantages - shown on Table 1 - ranging from CPU time and memory requirements to easiness of adaption on light changes and objects motion.

Method	Requires		Applications		Adaption to	
	High memory	High CPU time	Indoors	Outdoors	Object movement	Light changes
Hi freq	✓	✓	✓	✓	good	good
median	✓	✓	✓	✓	good	good
mean			✓	✓	medium	medium

Table 1: Advantages and disadvantages of background estimation methods.

An additional advantage of all above methods is that there is no need to filter and blur the background image after the extraction procedure.

The method that is finally chosen depends on application needs and the available hardware.

3. THE DIFFERENCE IMAGE

A difference image is the image that is produced by subtracting the background image from each frame and it is used in order to detect the parts of the image where movement is taking place. Additionally, in order to avoid noise and camera jitter, that produces false motion detection, the difference image must be thresholded. Threshold value cannot be calculated theoretically; it is usually determined after the sampling of many frames and its determination is based on the highest pixel differences. However, it is possible that some noise will still be present after threshold.

In general, if G(i,j) is the background image and F(i,j) any single frame, then the difference image D(i,j) is calculated and thresholded using the following formula:

$$D(i,j) = \begin{cases} 0 \text{ if } \left|F(i,j) - G(i,j)\right| \le threshold \\ 1 \text{ if } \left|F(i,j) - G(i,j)\right| > threshold \end{cases} \quad (1)$$

All pixels of the difference image that have value 1 include motion and *probably* belong to an object, whereas the pixels with value 0 are the same as the background and are ignored.

In color images, where there are usually three bands of color information, the difference image can be calculated either separated in each band or in a single grayscale band, which is the combination of the three-color bands.

Figure 1: Moving objects in a difference image from a basketball game

4. BLOBS AND OBJECTS

4.1 Blobs

Blobs have a long history in computer vision as a representation of image features (Kauth et al 1977, Pentland 1976.). A blob can be described as a set of connected pixels that share a common attribute. This attribute can be any one of the color, texture, brightness, shading or other salient spatio-temporal attribute, derived from the image sequence, or any combination of the above. In tracking, the common attribute is usually motion.

All motion pixels in the difference image are clustered in blobs. There are several techniques to cluster and extract blobs; most common techniques are “meandering process” (Rossey, 1997) and “connected pixels”. In our implementation we have used an updated version of the popular image processing FILL procedure; the implementation can extract blobs either from the whole image or from a specific part of it.

After blob detection, every blob can be represented by its properties; the most important of them are size, color and position.

4.1.1 Size: The size N of a blob is the number of the pixels that comprise the blob.

4.1.2 Color: The color C_m of a blob is the mean color of the N pixels that comprise the blob.

$$c_m = \frac{\sum_{1}^{N} F(i,j)}{N} \quad (2)$$

It must be noticed here that the pixel color values F(i,j) are taken from the original frame image.

4.1.3 Position: The position $P(x_m, y_m)$ of a blob is the geometric center of the N pixels that comprise the blob.

$$x_m = \frac{\sum_{1}^{N} i}{N}, \quad y_m = \frac{\sum_{1}^{N} j}{N} \quad (3)$$

4.2 Objects

Since objects create motion, difference image detects motion and blobs are image features that describe motion, objects can be described by blobs. One would expect that every object could be described by only one blob but the common case is that an object is comprised by more than one blobs, as shown in image 2.

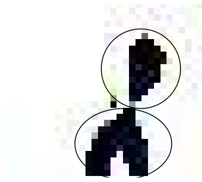


Figure 2: An object (basketball player) that is comprised by two blobs (head and body).

Much like blobs, every object can be represented with its properties such as color, size and position. These properties are calculated just like the blob ones, with the difference that in calculation all the actual blobs (and not their properties) are used. Additionally, the object must have some more properties to help tracking. The most important of these additional properties is the bound box that includes all the blobs of the object.

5. TRACKING ALGORITHM

The tracking algorithm uses two basic data structures:

1. A list of all active objects with their current properties
2. A history list that keeps all objects properties and information about matching type and score in time space.

The process of tracking follows the next steps:

1. Initialization of the objects to be tracked
2. Prediction of objects future status
3. Find new objects in predicted positions
4. Matching of objects
5. Register objects with new properties
6. Update history list
7. Go to step 2 and repeat process

5.1 Prediction of new object status

In order to help tracker find the objects in a new frame, the “expected status” of all objects is predicted. The “expected status” includes all object’s *main* properties.

The most common technique to predict these properties is the use of Kalman filters, technique that gives very good results. Alternatively, if tracking speed is high (usually more than 5 FPS) and the speed of the objects is relatively low, the motion of the objects can be assumed locally as linear and linear interpolation can be used with very good results, too.

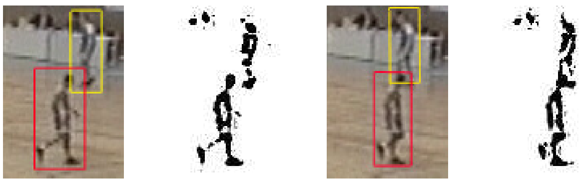
5.2 Collision detection

Since objects in "real world" applications can move unexpectedly in the tracking scene, they can be projected isolated or in touch with another or been hidden by some other. And, since the tracking of isolated objects is much easier than in other two situations, it is useful to help tracker predict in which status the objects are. This task can be achieved by predicting possible object collisions.

The property that is used to achieve collision detection is the object's bounding box. Every corner of each object's box is examined whether it is inside any other box. If any of the corners are inside, the objects are in collision and the tracking algorithm changes for these objects.

Three different collision stages may exist:

i. An object is isolated
ii. An object is collided with one or more but it's blobs are not in touch with other objects blobs
iii. Object is collided with one or more and it's blobs are in touch with other object blobs, producing common huge blobs



(a) (b) (c) (d)

Figure 3: Collision stage ii (a & b) and iii (c & d). Images (a) & (c) come from the original frame where images (b) & (d) come from the difference image.

Unfortunately, collision detection procedure usually cannot distinguish stage (ii) from (iii); this will be done later, during object matching.

5.3 Assigning blobs to objects

For every object to be tracked (from now on it will be referred as *old* object) a new one is created; it inherits all the properties of the old one *except* it's main properties. Then, the blobs that exist into the expected bounding box are assigned to new object - depended on old object's expected collision stage - in order to calculate the *new* main properties that will be used to identify if two objects are similar.

If the old object is expected to be isolated, *all* blobs in expected bounding box are assigned to the new object. Additionally, from time to time, the expected bounding box can grow up in order to avoid a bad prediction of its size or to re-estimate the actual object size.

If the old object is expected to be in collision stage (ii) or (iii), only the blobs in expected bounding box that their distance is smaller than a distance threshold are assigned to the new object. The others are dropped. The distance threshold value depends on image scale and object mean size.

Blob dropping may help in distinguish two or more collided objects but also leads to shrinking of the object's actual size. Therefore, it must be used with caution and only in collision stages. Additionally, when the object comes to isolated stage in later frames, its expected bounding box must grow up to re-estimate object's actual size.

After blob assignment, new object's properties are calculated, as described in §4.2, and object match procedure follows.

5.4 Matching an old object with the new one

The term *matching* has the meaning of *identification*. The new object matches to old one if they are *similar*. If they do match, the position (goal of tracking) of the old object in new frame will be also the position of the new object.

Object main properties are used to calculate the similarity between two objects. That is they are similar if

- the weighted sum of color and size differences is larger than a threshold AND
- the distance between the two objects is lower than a distance threshold.

$$s_1 = p_c(1-(c_n - c_o)/c_n),\ s_2 = p_N(1-(N_n - N_o)/N_n)$$

$$s = \sqrt{s_1^2 + s_2^2} \quad (4)$$

$$d = \sqrt{(i_n - i_o)^2 + (j_n - j_o)^2} \quad (5)$$

match if ($s > t_1$) AND ($d < t_2$) (6)

where s = weighted sum
p_c, p_N = weights for color and size
c_n, N_n = color and size of *new* object
c_o, N_o = color and size of *old* object OR *predicted* ones
i_n, j_n = coordinates of *new* object
i_o, j_o = coordinates of *old* object OR *predicted* ones
t_1 = threshold
t_2 = distance threshold

Parameters p_c, p_N and threshold t_1 are application depended and estimated after sampling on many frames. Threshold t_2 depends on object's maximum speed vector.

There are two weighted sums that calculated:

– Between new and old object *actual* properties and
– Between new object and olds' *predicted* properties.

The second weighted sum is very helpful when sudden or *great* changes in color or size of the object happen. If any of the above sums passes the threshold, the two objects are assumed similar and the new object replaces the old one.

Additionally, separate comparisons are made on every property and flag successful or failed individual matching.

The above matching formula (6) works with very good results when object is isolated or in collision stage (ii). If object is in stage (iii) (Fig3, c&d) then it's size grows up suddenly with the possibility that also its color differs a lot from previous frame; the weighted sums then cannot pass the threshold and the matching flag on size and possibly on color is failure. In this case, matching fails and the *position* of the object is re-estimated using an adaptive correlation matching procedure.

5.4.1 Adaptive correlation matching

The adaptive correlation matching is a variation of the classic cross correlation technique. Their difference stands to the pixels that participate in correlation. In adaptive correlation, only pixels that belong to object's blobs participate, as shown in fig. 4, in order to avoid background disturbance.

(a) (b)

Fig. 4: Correlation templates. (a) typical, (b) adaptive. White pixels do not participate in correlation procedure.

The template of the object is created using the last frame in which this object has been detected isolated or in stage (ii), with the use of the history list. The search area is around object's predicted position in current frame and its data come from the original frame.

As in typical correlation, it is possible that a lot of template positions will have high correlation scores and the correct object's position is not that with the highest one; in this case, the assumed correct position is the one with the minimum distance from the predicted position. The object's *position* is updated in the current frame but its *size* and *color* retain their values from the last frame.

If maximum correlation factor is very low, the matching fails; But in this case, the following assumption is made:

- If there are only two objects that are in stage (iii) then the object is assumed hidden by the other; therefore, its *position* is the position of the object that stands in front of it.
- If there are more than two objects in stage (iii) then it is unknown which object hides the wanted one and the object needs re-initialization in next frame.

Correlation match will work for a short range of sequent frames. If an object is in stage (iii) for a long time, then its shape will be very different after a number of frames from the shape it had at the start of the sequence. Additionally, there will be a drift on template center, as time passes (Intille, 1996). Therefore, correlation matching must be used for a short time period (for few frames).

6. TESTING THE ALGORITHM - PERFORMANCE

6.1 Example

The described tracking technique was tested on a 430 frames video sequence of a basketball game, having 5 fps speed and image size 768x576 pixels. The scale of the tracking scene varied from 1: 130 up to 1:70 while the size of the objects varied from 116 to 400 pixels. The color variation was different for each object and was from 7 gray shades up to 40 (in extreme cases), with a mean variation of 20 gray shades.

The parameters that were used for object identification (matching) were:

p_c : 0.60, p_N : 0.40, t_1 : 0.65, t_2 : 15

6.2 Performance

The tracker managed to track all isolated objects in all frames with no failure and collision stage (ii) objects with two failures. In case of collision stage (iii) objects, tracker succeeded only on two objects collision and in few cases on three objects, when objects shared the same space for a short time period; In general, it failed when *more than two objects shared* the same space. In these cases re-initialization was performed.

7. CONCLUSIONS

The described technique works very good when the objects do not share the same blobs or only two objects are interacting. In general, it suffers when more than two objects are in the same space or the movement of the objects is rapid.

Since the tracking depends on correct interpreting of which objects share the same blobs, in conjunction with objects motion, current research is focused on this aspect.

REFERENCES

Gloyer, B., H. Aghajan, K.Y. Siu, T. Kailath, "Video based Freeway monitoring System using Recursive Vehicle Tracking", Proc. IS&T/SPIE, Symposium on Electronic Imaging, 1995

Intille , S, J, Davis, A, Bobick, "Real Time Closed-World Tracking", CCVPR, IEEE, 1996

Intille, S, A, Bobick, "Visual Tracking Using Closed Worlds", ICCV, IEEE, 1995

Intille, S., "Tracking Using a Local Closed World Assumption: Tracking in the Footbal Domain", Master Thesis , August 1994

Kauth, R.J, A.P. Pentland, G.S Tomas, "Blob: an unsupervised clustering approach to spatial preprocessing of MSS imagery", XI Int. Symp. of RS of the Environment, Ann Arbor, MI, 1977

Ohno Y., J. Miura, Y. Shirai, "Tracking Players and Estimation of the 3D Position of a ball in Soccer games", ARIDA News Letter #16,January 2000, pp. 3-6

Pentland, A, "Classification by clustering", Proc of Symp on MPRSD, IEEE Computer Society Press, 1976

Rossey, L, "Design and Implementation of Opinion-Based Behaviors for an Autonomous Mobile Robot with Vision", Master Thesis, 1997

Wren , C., A. Azarbayejani, T. Darrell, A. Pentland, "Pfinder : Real-Time Tracking of the Human Body", TPAMI, IEEE, vol. 19,no 7, pp. 780-785