REAL-TIME TEMPLATE BASED TRACKING WITH GLOBAL MOTION COMPENSATION IN UAV VIDEO

Yuriy Luzanov and Todd Howlett
Air Force Research Laboratory, USAF, 525 Brooks Rd, Rome NY, 13441, USA

Mark Robertson

Keywords: Kalman filter, frame alignment, target tracking, UAV video.

Abstract: In this paper we describe a combination of Kalman filter with global motion estimation, between consecutive frames, implemented to improve target tracking in the presence of rapid motions of the camera encountered in human operated UAV based video surveillance systems. The global motion estimation allows to retain the localization of the tracked targets provided by the Kalman filter. The original target template is selected by the operator. SSD error measure is used to find the best match for the template in video frames.

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are used in military and law enforcement for surveillance and reconnaissance missions. There are several types of sensors available on UAV platforms. But, video cameras are the most common type of information gathering sensors. At this time, real-time detection and tracking of ground targets in UAV video is accomplished manually. Automatic target detection and tracking algorithms do not possess the level of reliability required for real-time operation in the field.

There are many challenges in automatic real-time tracking of ground targets in video from UAVs. To name a few, the background is continuously changing due to the motion of the airframe and motion of the pan/tilt/zoom camera; the quality of the video is poor; camera assemblies tend to undergo sudden rapid motions, primarily due to human operators, but also turbulence and air frame maneuvering.

A popular technique for tracking targets in a video stream is template matching. This paper discusses an improvement to the real-time, template matching based, ground target tracking algorithm in UAV video.

A typical target tracking system consisting of a template matching module and a Kalman filter was implemented. The template matching module was used to locate the target in each successive video frame. The Kalman filter was used to predict the location of the target based on its previous location in the image and a target's motion model. The search region was restricted to a small window, centered at the predicted location. This allows for a tight localization of the target and the reduction of false positive matches. But, when sudden and rapid changes occur in the video the predictions of the targets' locations can have significant errors, often resulting in the failure of the tracking algorithm.

To negate the effects and account for such rapid motions in the video it is proposed to use a frame alignment algorithm to determine the global motion between consecutive frames. The global motion is applied as a control input to the Kalman filter. As a result, the prediction of the location of the target is significantly improved, increasing the tracking performance. The system presented in this paper is able to achieve a real-time performance of 24 ms of computational time per frame, with 640x480 video, on a Pentium IV class workstation.

The paper is organized as follows. First, template matching techniques used in the implementation of the tracking algorithm are discussed in section 2. Then, an implementation of the Kalman filter is discussed in section 3. The next section, 4, presents the details of the frame alignment algorithm, describing the integration with the Kalman filter. Finally, in section 5, the results and conclusions are presented. Values for all of the parameters used in the implementation are presented in the Appendix.

2 TEMPLATE MATCHING

Template matching is used to determine the location of the target in each frame of the video sequence. The main assumptions made with this technique are that there are no significant occlusions of the tracked target and the target's appearance changes gradually.

The objects in the video change appearance over time. The initial template selected by the operator in the first frame will become inaccurate after a period of time. A simple approach to try and overcome this problem is to update the template after a certain amount of frames have been processed. The main difficulty here is to find the proper update rate. If it is too high, the template is being updated too often, then the error will accumulate quickly resulting in a drift off of the object originally selected by the operator. On the other hand when the update rate is too slow, the target will be lost due to dissimilarities between its current appearance and the template. There were several techniques proposed in order to determine when and how to update the template (Matthews et al., 2004). Here the template is simply replaced every p frames by the best matched region. The update rate, $\frac{1}{p}$, is determined experimentally.

Let $I(x,y,t)$ denote a pixel intensity at location (x,y) in the video frame of size $N \times M$ pixels at time t, where $x \in [0, N-1]$ and $y \in [0, M-1]$. The template is initialized by the operator at time $t = 0$. Let $S(i, j, l)$ be a pixel intensity at location (i, j) of a template extrated from $(lp)^{th}$ frame, $I(x,y,lp)$, where l and p are nonnegative integers. The size of the template $S(i, j, l)$ is $K \times K$ pixels, so that $i, j \in [0, K-1]$. To eliminate any ambiguities in determining the center of $S(i, j, l)$, K is picked to be an odd positive integer.

Using Sum of Squared Difference (SSD) error measure, the error between the template $S(i, j, l)$ and the image $I(x,y,t)$ at point (a,b) can be written as follows:

$$e = \sum_{i=0}^{K-1}\sum_{j=0}^{K-1} \left(S(i,j,l) - I(a - \frac{K}{2} + i, b - \frac{K}{2} + j, t)\right)^2 \tag{1}$$

where $\frac{K}{2}$ is rounded down to the nearest integer.

By computing the error in equation (1) for every pixel of the frame at time t, and finding the minimum error, the best matched region could be determined. This approach would result in a longer computation time and poor localization of the target due to false positive matches. One solution is to define a region R_s of size $W \times W$ pixels, centered at the most probable location of the target. Then the search for the best match would be carried out only within R_s. In the implementation W ranges between W_{min} and W_{max} and is determined by how fast the target moves in the image.

3 KALMAN FILTER

In order to approximate the next location of the target in the video a Kalman filter is employed (Welch and Bishop, 2004). A basic Kalman filter was found to be sufficient for this application. When the operator initializes the template in the first frame, by selecting a point inside the target region, a Kalman filter is also initialized. The point, originally selected by the operator, will be tracked through the video. The discrete-time state-space representation of the linear process, the state of which is being estimated, can be expressed as:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{2}$$

$$z_k = Hx_k + v_k \tag{3}$$

In equation (2) the state vector, $x \in \mathbf{R}^4$, contains the position and the velocity of a 2D point being tracked. State transition matrix, $A \in \mathbf{R}^{4\times4}$, relates the previous state, x_{k-1}, to the current state, x_k, where, k, is a nonnegative integer. Control input matrix, $B \in \mathbf{R}^{4\times2}$, provides coupling between the control input vector, $u \in \mathbf{R}^{2\times1}$, and the state. The control input is the 2D global displacement of the image. In equation (3), $z \in \mathbf{R}^2$, is the measurement vector. The measurement is a 2D position of the target's point. The observation matrix, $H \in \mathbf{R}^{2\times4}$, relates the state to the measurement. The random variable, w, represents the process noise and the random variable, v, represents the measurement noise. They are independent, white and normally distributed:

$$w \sim N(0, Q) \tag{4}$$

$$v \sim N(0, R), \tag{5}$$

where $Q \in \mathbf{R}^{4\times4}$ is the process noise covariance matrix and $R \in \mathbf{R}^{2\times2}$ is the measurement noise covariance matrix. Both Q and R are assumed to be constant.

A Kalman filter consists of two stages: the prediction and the correction. In the prediction stage an estimate of the current state, $\hat{x}_k$, and estimate of the current error covariance matrix, $\hat{P}_k$, are made:

$$\hat{x}_k = Ax_{k-1} + Bu_{k-1} \tag{6}$$

$$\hat{P}_k = AP_{k-1}A^T + Q, \tag{7}$$

where error covariance matrix $P \in \mathbf{R}^{4\times4}$. The correction stage uses the measurement z_k to refine the prediction and compute the Kalman gain K_k, the current state x_k and error covariance P_k:

$$\begin{aligned} K_k &= \hat{P}_k H^T (H\hat{P}_k H^T + R)^{-1} && (8)\\ x_k &= \hat{x}_k + K_k(z_k - H\hat{x}_k) && (9)\\ P_k &= (I - K_k H)\hat{P}_k && (10) \end{aligned}$$

For each frame the following computations are performed. First a prediction of the target's location is made using (6), (7). Then the search region R_s is initialized with predicted target's position from $\hat{x}_k$. Now, the template matching is performed within the region R_s and the center of the best match is used as the measurement z_k to correct the filter with equations (8), (9) and (10).

The motion of the object in the video is usually linear, except when the camera undergoes sudden and rapid movements. If that happens the assumption of linearity is violated and the Kalman filter fails.

4 ROBUST FRAME ALIGNMENT

Frame alignment is used to compute the global motion and supply the control input to the Kalman filter. Section 4.1 provides the formulation of the alignment algorithm, after which Section 4.2 shows how to make the formulation robust to outliers and data not well modeled by the original formulation.

4.1 Aligning Background

We denote the intensity of pixel (x,y) at time t as $I(x,y,t)$. To relate the pixels of two frames together, one can apply the intensity constraint

$$I(x,y,t) = I(x+\Delta x, y+\Delta y, t-1), \qquad (11)$$

which if we approximate with the linear terms of a Taylor approximation we obtain

$$\Delta x \frac{\partial}{\partial x} I(x,y,t) + \Delta y \frac{\partial}{\partial y} I(x,y,t) - \frac{\partial}{\partial t} I(x,y,t) \approx 0 \quad (12)$$

The motion at pixel (x,y) is $(\Delta x, \Delta y)$. For notational convenience, we define f_x, f_y, and f_t as the partial derivatives in (12). These partial derivatives are implemented as digital approximations of horizontal, vertical, and temporal gradients. The various quantities are subscripted with an "i" to denote their values at the i^{th} pixel.

By imposing a global motion model at each pixel location (x_i, y_i) in the image, we can parameterize the motion to make a solution at each pixel more tractable. A two-parameter motion model that accounts for translation in the horizontal and vertical directions would be

$$\Delta x_i = a_{13}, \qquad \Delta y_i = a_{23} \qquad (13)$$

A weighted LS derivation yields the following system of equations:

$$\begin{bmatrix} \sum w_i f_{x_i}^2 & \sum w_i f_{x_i} f_{y_i} \\ \sum w_i f_{x_i} f_{y_i} & \sum w_i f_{y_i}^2 \end{bmatrix} \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix} = \begin{bmatrix} \sum w_i f_{x_i} f_{t_i} \\ \sum w_i f_{y_i} f_{t_i} \end{bmatrix} \qquad (14)$$

where w_i is the weight for the i^{th} pixel position. These equations are easily solved for the two motion parameters. More complex global motion models are also possible, but at the cost of additional computational time.

4.2 Making Alignment Robust

Frame alignment as described in the previous subsection works well when the two frames are well modeled by the particular global motion model in use. However, even when the underlying background is well modeled, sometimes large errors can still result. For example, if there are objects moving independently in the scene, their motion will not match that of the background, and large errors will occur in these positions. Similarly, if there are burned-in metadata on the frames, they will not move according to the background, and large errors will result as well. To make the frame alignment procedure more robust to such errors, we employ an iterative reweighted LS scheme (Odobez and Bouthemy, 1994). On iteration $k = 0$, when there is no knowledge about errors, all weights are chosen as one, $w_i^{(0)} = 1$. The motion parameters (for whichever model is being used) are solved to yield motion estimates for the k^{th} iteration, $(\Delta x_i^{(k)}, \Delta y_i^{(k)})$. The residual error for location i at iteration k is then computed as

$$r_i^{(k)} = I(x_i, y_i, t) - I(x_i + \Delta x_i^{(k)}, y_i + \Delta y_i^{(k)}, t-1) \quad (15)$$

The residual error represents the difference between the i^{th} pixel value of frame t, and the pixel value at location in frame $t-1$ to which we believe (at iteration k) it has moved. If we use the Tukey biweight function (Odobez and Bouthemy, 1994), we then pick the weight for the next iteration $k+1$ as:

$$\sqrt{w_i^{(k+1)}} = \begin{cases} C^2 - \left[r_i^{(k)}\right]^2 & |r_i^{(k)}| < C \\ 0 & \text{otherwise} \end{cases} \qquad (16)$$

This weight is then used in the weighted LS formulation to get an estimate of the motion at iteration $k+1$, after which the procedure continues iteratively until a

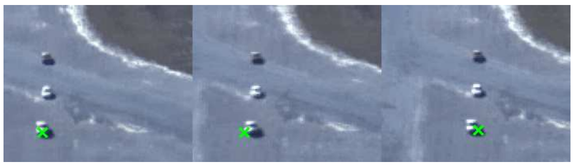
Figure 1: Tracking a car in consecutive frames, left to right.

convergence criterion is met or else some fixed number of iterations have completed. A similar weighting scheme to (16) is given as

$$\sqrt{w_i^{(k+1)}} = \begin{cases} C - |r_i^{(k)}| & |r_i^{(k)}| < C \\ 0 & \text{otherwise} \end{cases}, \qquad (17)$$

which makes use of an absolute value rather than a square, and may be slightly faster on some architectures.

Reweighting according to (16) or (17) is equivalent to a formulation that minimizes not the sum of the squares of the error as in LS, but rather the sum of a function of the errors.

Once the global motion, $(\Delta x, \Delta y)$, between the last and the current frames is estimated it is applied to the Kalman filter prediction step as a control input, $u_{k-1} = [\Delta x, \Delta y]^T$ for $k \geq 1$, in equation (6).

5 RESULTS AND CONCLUSIONS

The algorithm was implemented on a 3.0 GHz Intel Xeon workstation in a single process, without using any special instructions. A real-time performance was achieved. On average, 24 ms was required to process a single 640x480 frame of video. This includes the computation of the global motion between the current and the previous frame, Kalman filter prediction and correction, and template matching.

As part of the test, sample UAV videos from the DARPA sponsored VIVID program were used, (Collins et al., 2005). Figure 1 illustrates the algorithm tracking a car in three consecutive frames in the presence of a significant camera motion.

As a result of using the global motion compensation the tracker was able to maintain the track of the target using a small search region.

There are certain advantaged of employing the global motion compensation. This is apparent in situations with complex scenes, i.e. urban scenes, where multiple objects in the video frame have similar appearance. In order to successfully track a target a small size search window is used to avoid false positive matches.

The disadvantage of using the global motion compensation is that it takes a very long time to compute. If the complexity of the scene permits, the target has a unique appearance, it may be faster to use a larger search area, rather then to compute the global motion.

REFERENCES

Collins, R., Zhou, X., and Teh, S. K. (2005). An open source tracking testbed and evaluation web site. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*.

Matthews, I., Ishikawa, T., and Baker, S. (2004). The template update problem. *IEEE Transactions on Pattern Analisys and Machine Intelligence*, 26(4):810–815.

Odobez, J.-M. and Bouthemy, P. (1994). Robust multiresolution estimation of parametric motion models applied to complex scenes. Publication Interne IRISA 788, Institut de Recherche en Informatique et Systèmes Aléatoires.

Welch, G. and Bishop, G. (2004). An introduction to the kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill.

APPENDIX

The following are the values for the parameters used in the implementation of the algorithms.

For template matching 11×11 templates were used, thus $K = 11$ and $\frac{K}{2} = 5$. The size of the search area R_s was between $W_{min} = 10$ and $W_{max} = 32$ pixels. The template was updated every 15 frames, thus $p = 15$.

In the implementation of the Kalman filter the following state-space matrices were employed:

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad (18)$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad (19)$$

The process noise and measurement noise covariance matrices were:

$$Q = 0.01 I_{4\times4},\ R = I_{2\times2} \qquad (20)$$

The filter is initialized with x_{k-1}, P_{k-1} and u_{k-1} at $k = 0$, thus the initial values are x_{-1}, P_{-1} and u_{-1} and they are:

$$x_{-1} = \begin{bmatrix} x_{init} & y_{init} & 0 & 0 \end{bmatrix}^T \qquad (21)$$

$$P_{-1} = 10 I_{4\times4} \qquad (22)$$

$$u_{-1} = \mathbf{0} \qquad (23)$$

In equation (21) the values x_{init} and y_{init} are provided by the operator's initial selection of the target.

The Tukey biweight parameter was $C = 40$.