MULTI TARGET TRACKING ON AERIAL VIDEOS

Gellért Mátyus, Csaba Benedek and Tamás Szirányi
Distributed Events Analysis (DEVA) Research Group
Computer and Automation Research Institute of the Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, 1111, Kende utca 13-17., Hungary
mattyus@sztaki.hu, bcsaba@sztaki.hu, sziranyi@sztaki.hu
http://web.eee.sztaki.hu

KEY WORDS: Motion Compensation, Motion Detection, Multi Object Tracking, UAV Video

ABSTRACT:
In this paper we propose a new method to detect and track multiple moving targets on image sequences recorded by Unmanned Aerial Vehicles (UAVs). Our approach focuses on challenging urban scenarios, where several object are simultaneously moving in various directions, and we must expect frequent occlusions caused by other moving vehicles or static scene objects such as buildings and bridges. In addition, since the UAVs are flying at relatively low altitude, the 3D-ness of the scene affects strongly the camera motion compensation process, and the independent object motions may be often confused by artifacts of frame registration. Our method enables real time operation, processing 320x240 frames at around 15 fps and 640x480 frames at 5 fps.

1 INTRODUCTION
Nowadays Unmanned Aerial Vehicles (UAVs) are becoming more and more important in military operations (Kumar et al., 2001). Since there is no pilot in such aerial vehicles, they can be sent to missions without endangering human life. The lack of personnel has several other benefits, e.g. reduced weight, the mission length is not a function of pilot fatigue, and the planes can achieve better maneuverability since the human tolerance to acceleration is not a limitation anymore.

Detecting objects of interest is a key task in reconnaissance and surveillance applications and also on the combat field. The moving objects are in most cases relevant, since they are frequently vehicles or persons. The automatic detection of these moving objects can help the operator by giving a caution to them. The tracking of the moving objects can also give useful informations i.e. a vehicle is moving toward the defended camp.

Change detection in video sequences can also reduce the size of the data to be transmitted to the control station. To avoid redundancy, there is no need to transmit the pixels belonging to an unchanged area.

In this paper we introduce a method which deals with the above subtasks. The main steps of the proposed approach are demonstrated in Figure 1:

- (i) video stabilization
- (ii) foreground extraction
- (iii) moving object detection
- (iv) tracking

The first step is the compensation of the camera’s ego-motion (Benedek et al., 2009). This can be achieved by warping the frames to a common coordinate system, where the images can be considered still. This step can also provide a better visual information to the operator by avoiding the shaking of the camera. The images in the common coordinate system can be handled by similar algorithms to ones developed for fixed cameras. However, we must consider that ego-motion compensation is not totally accurate, thus efficient filtering of the registration noise is needed. The tracking also needs the world coordinate system, where the position of an object in the image refers to it’s real position in the world. The transformation to the world coordinate system needs additional information, i.e. global position, camera parameters.

In the image warped into the common coordinate-system, the motion detection is done by background subtraction. Then we obtain a foreground mask, which shows the moving pixels. The moving objects are blobs on the mask, which can be detected and tracked. Note that foreground detection may contain errors, e.g. some blobs can be split, or only parts of a given moving object are covered by its blob. Therefore, some a priori knowledge is needed about the objects of interest. In aerial videos the size of the objects is a good feature, because it can be easily estimated for several targets such as cars or pedestrian, using the camera parameters (camera altitude, angles, focal length).

In the final step, the detection results from the separate frames are assigned to each other by applying Kalman filtering for the consecutive positions and considering the object histograms. The assigned object positions on the frames yield the track.

2 VIDEO STABILIZATION

The ego-motion compensation is achieved by calculating an optimized homogen linear transform between the frames, and warping the images to a common coordinate system.

The applied image registration and alignment techniques are detailed in (Szeliski, 2006).

The perspective transformation between two images taken of a plane surface can described by the homography matrix $H$. One point $p_0$ is transformed to $p_1$ by the following equation:

$$
\begin{bmatrix}
    x_1 \\
    y_1 \\
    z_1
\end{bmatrix}
= 
\begin{bmatrix}
    H_{11} & H_{12} & H_{13} \\
    H_{21} & H_{22} & H_{23} \\
    H_{31} & H_{32} & H_{33}
\end{bmatrix}
\begin{bmatrix}
    x_0 \\
    y_0 \\
    1
\end{bmatrix}
$$

(1)
2.2 Feature based registration

The feature based techniques begin with extraction of feature points on the images. These points can be found and aligned on the two image. This yields a points set \( S \) on one image which is aligned to point set \( S' \) on the other image.

\[ S \Rightarrow S' \]

By fitting a transformation to these points the transformation matrix can be estimated.

\[ S' = HS \]  \hspace{1cm} (4)

Popular feature point detectors are the Harris corner point detector (Harris and Stephens, 1988), the SIFT detector (Lowe, 2004) and the SURF detector (Bay et al., 2006).

In case of aerial images the transformation cannot be restricted to translation or rotation, thus the more general affine or perspective transformation has to be used. Therefore, we use the feature based method to find the homography matrix of the perspective transformation.

2.3 Feature points

We use the Harris corner detector which is more suitable in man-made environments where corners are abundant. It is also computationally less expensive than the other feature point detectors such as SIFT or SURF. Note that if there are no feature points, like in large homogenous areas, the registration fails.

2.4 Point alignment

Corresponding points are searched on two consecutive frames by the Lucas-Kanade pyramidal optical flow algorithm. This step yields the positions of the feature points on the next image, thus the transformation between the frames can be calculated. The applied optical-flow algorithm assumes small displacement and nearly constant pixel intensity across the consecutive frames. This constraint is fulfilled on the considered videos. The transformation is fitted by RANSAC (Fischler and Bolles, 1981) to the extracted point correspondences to reduce the effect of outliers.

If the transformation between two frames is available, the frames can be warped into a common coordinate-system. The common coordinate system is a periodically chosen reference frame. There is a homography matrix between the two following frames \( n, n−1 \) \( H_{n,n−1} \), and a homography between the frame and the reference frame \( H_{n,0} \):

\[ H_{n,0} = H_{n,n−1}H_{n−1,n−2} \cdots H_{2,1}H_{1,0} \]  \hspace{1cm} (5)

The current image is warped into the coordinate-system of the reference image by the Homography matrix \( H \). \( I_d \) is the pixel value in the destiny image, \( I_s \) is the pixel value in the source image.

\[ I_d(x,y) = I_s \begin{pmatrix} H_{11}x + H_{12}y + H_{13} \\ H_{21}x + H_{22}y + H_{23} \\ H_{31}x + H_{32}y + H_{33} \end{pmatrix} \]  \hspace{1cm} (6)

The warped image has artifacts, because of the unperfect estimation of the transformation and discretization errors. The homography transformation results in continuous coordinate values which
should be discretized into pixel positions, which may be relevant in strong perspective cases, when a given source pixel is warped to several pixels of the output image. This reduces the effective resolution of the image.

3 FOREGROUND EXTRACTION

The image registration yields an image that looks like a window in a global image (see Figure 2). If the registration is optimal only the pixels belonging to a moving object are changing, though this cannot be fully achieved due to image registration and parallax errors.

Working on the considered videos, these errors are typically located along the edges and their expansion is narrow (a few pixels).

3.1 Background model

The background image is synthesized and updated in the common coordinate system, calculating the pixel-by-pixel running average and variance of the consecutive warped video frames. Note that the widely used Mixture of Gaussians (MOG) approach (Stauffer and Grimson, 1999) cannot be adapted to our case, since due to the fast camera motion we can often observe only a few samples (less than 10) for each surface point, which is not enough to describe the distribution by MOG.

We calculate the mean value $\bar{x}_n$, and the variance $\sigma^2_n$ for every pixel on-line:

$$\bar{x}_n = (1 - \alpha)\bar{x}_{n-1} + \alpha x_n$$

where $\alpha$ is a constant that gives the refresh rate.

$$\sigma^2_n = (1 - \alpha)\sigma^2_{n-1} + \alpha(x_n - \bar{x}_n)(x_n - \bar{x}_{n-1})$$

3.2 Foreground detection

The pixels of the actual frame are classified either as foreground or background based on the normalized Euclidean distance from the background pixel values in the CIE L\*u\*v\* color space. This is the Mahalanobis-distance for diagonal covariance matrix.

$$d(p_n) = \sqrt{\sum_{i=1}^{3} \frac{(p_{n,i} - \bar{p}_{n-1,i})^2}{\sigma^2_{n-1,i}}}$$

Figure 2: Warped frames

This yields a distance image, which is noisy as one can see on Figure 3. This noisy image is filtered by a special Difference of Gaussians filter, applying Gaussian blur and threshold. The blurring spreads the narrow pixel errors, thus the concerning values in the difference image drop below the threshold level. The moving objects correspond to blobs on the foreground mask, though the mask of an object can be split and incomplete. Figure 4 shows the foreground mask marked by red color on the image.

4 MOVING OBJECT DETECTION

The foreground detection yields a binary mask on which the moving objects, e.g. cars and pedestrians are blobs. These blobs can be noisy, split, etc., and there can be false detected blobs which do not belong to moving objects. So in general the blob detection is under-constrained. By using a priori knowledge the blob detection can be restricted to special blobs, i.e. by shape or size.

We propose a fast object detection algorithm which is based on the foreground mask; meanwhile it considers split and incomplete object blobs. The input is the size of a car. The size of the cars can be precisely defined. If we know the altitude, the angles (raw, pitch) and the focus length of the camera (the airborne vehicles have an inertial navigation system which can provide these parameters) the size of the car can be approximately calculated by assuming that it is moving on the ground. On the videos we have tested these parameters are unknown, thus we have estimated the size of the cars manually.

The initial step of the object detector algorithm divides the mask image to disjunctive rectangles with size $x \times x$, where $x$ is the size of the car in pixels, and the foreground covering ratio is calculated for each rectangle. The rectangles containing foreground pixels above a threshold are kept as object candidates (OC).

Next, the OC-s are shifted and/or merged by an iterative algorithm. An OC is shifted by mean-shift based on the binary mask.
The steps of tracking are shown in figure 6.

5 TRACKING

The object detection is processed for each frame independently. Thus these detections have to be assigned across the frames to yield the tracks of the objects. The difficulty is that in general the number of object detections for consecutive frames can vary even in the case of perfect detection, i.e. objects enter and leave the scene, objects are occluded by buildings or bridges. To handle the disappearing and later reappearing objects Kalman filtering is used.

The steps of tracking are shown in figure 6.

5.1 Kalman filter

The Kalman filter is an efficient tool for filtering a noisy dynamic system. It predicts the new states of the system and then corrects it by the measurements.

In tracking we do not have information about the control of the motion, therefore the acceleration is assumed to be zero, and the change in velocity is modeled by the process noise. Consequently, we do not include the acceleration in the process equation, and the effect of the acceleration noise is described by the velocity noise. The motion can be described by the following equations:

\[
\begin{align*}
\dot{x}_k &= \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \end{bmatrix} + w_{k-1} \\
z_k &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x}_k + v_k \end{bmatrix} 
\end{align*}
\]

Where \( x_k \) is the position coordinate in one direction, \( z_k \) is the measured position, \( w_{k-1} \) is the process noise, \( v_k \) is the measurement noise.

5.2 Assignment

On each current frame the \( k \) detected objects have to be assigned to \( n \) tracked objects from the previous frames. If \( n = k \), this can be done in \( n! \) ways. The Hungarian method solves this in \( O(n^3) \) running time. We solve the assignment problem with a greedy algorithm, which is computationally simple and gives good result in most cases. The cases when the greedy algorithm fail can be neglected, since they last only a few frames, and on a long term the Kalman filtering corrects these errors.
We construct a $n \times k$ score matrix, $S$, whose elements are calculated based on the euclidean distance of the predicted and detected positions and the object’s color histogram. The elements of the matrix are fitness values which describe how good the objects from the previous frames match the objects detected on the current frame.

$$S_{ij} = \vartheta \frac{1}{d_{\text{pos}}(O_i, D_j)} + (1 - \vartheta)d_{\text{hist}}(O_i, D_j)$$  \hspace{0.5cm} (12)

where $d_{\text{pos}}$ is the euclidean distance of the positions, $d_{\text{hist}}$ is the histogram similarity and $\vartheta$ is a weight value.

$$d_{\text{pos}}(O_i, D_j) = \sqrt{(\hat{p}_{i,x} - p_{j,x})^2 + (\hat{p}_{i,y} - p_{j,y})^2}$$  \hspace{0.5cm} (13)

$\hat{p}_i$ is the predicted position of object $O_i$, $p_j$ is the position of detected object $D_j$.

The steps of the score table $S_{ij}$ calculation for every $i,j$ pair:

1. Calculate the distance $d_{\text{pos}}$ of the predicted position $\hat{p}_i$ of object $O_i$ and the position $p_j$ of the detected object $D_j$.
2. if $d_{\text{pos},ij} > \varsigma$, $S_{ij} = 0$ and terminate. $\varsigma$ is a threshold parameter.
3. if $d_{\text{pos},ij} \leq \varsigma$ calculate $S_{ij}$ according to (12)

If the number of detected objects is equal or greater than the number of tracked objects from the previous frames, the assignment is done forward, this means that the tracked objects are assigned to the detected ones.

**The steps of forward assignment:**

1. $i=1$
2. for $O_i$ find max $m_i$ in $S_{1...k}$, $m_i = S_{ij}$
3. if $m_i > \varsigma$, assign $D_j$ to $O_i$, set the column $S_{1...k} = 0$
4. $i = i + 1$, go to step 2 until $i \leq n$
5. set the not assigned objects $O_{na}$ to passive, $O_{na} \rightarrow \text{passive}$
6. create new objects $O_{\text{new}}$ for the detections which are not assigned to an object $D_{na}$, and assign these new objects to them. $O_{\text{new}} \rightarrow D_{na}$

If the number of detected objects is less than the number of tracked objects from the previous frames, the assignment is done backward, this means that the detected objects are assigned to the tracked ones. Distinguishing between the two assignments is needed because the algorithm is greedy, thus the first objects in the order have priority.

**Assignment backward:**

1. $j=1$
2. for $D_j$ find max $m_j$ in $S_{1...n}$, $m_j = S_{ij}$
3. if $m_j > \varsigma$, assign $O_i$ to $D_j$, set the row $S_{1...k} = 0$
4. $j = j + 1$, go to step 2 until $j \leq k$
5. set the not assigned objects $O_{na}$ to passive, $O_{na} \rightarrow \text{passive}$
6. create new objects $O_{\text{new}}$ for the detections which are not assigned to an object $D_{na}$, and assign these new objects to them. $O_{\text{new}} \rightarrow D_{na}$

**During the assignment process:**

- Objects are assigned to detections.
- New objects are created and assigned to detections.
- Objects are set to passive. (no detections are assigned to them)
- Objects are deleted.

The figure 7 shows an assignment. A detection is assigned to an object only if it is close enough (this is given by the threshold parameter $\varsigma$) to the predicted position of the object. Object A is assigned to detection “3”, Object B is not assigned to a detection thus it is set to passive. For the detections 1, 2, 4 new objects are created.

### 6 RESULTS AND CONCLUSION

The algorithm was tested on various videos, taken in rural, urban and suburban environment from plane and balloon. We evaluate the tracking in a qualitative way.
On the videos which contained corner points over all the image, the registration was accurate, therefore the tracking was also accurate. The videos which lacked corner points could not be registered accurately therefore the algorithm failed.

In the urban environment the algorithm was accurate in weak and middle traffic. Also in the presence of a bridge which occluded the cars.

The dense traffic caused errors, because in this case the algorithm can hardly distinguish between the background and the moving vehicles.

The algorithm was implemented in OpenCV 2.0. The program was tested on an Intel Core i7, 2.67 GHz processor. The execution time per frame, for a 640 × 480 video was about 250 ms.

One of the most important input is the relative size of the car in pixels. The blob detection on the foreground mask highly depends on this. It has to be considered that the size of the cars varies by the pixel position on the image if the camera view is not vertical. The proposed algorithm calculates with a fixed car size along all the image, thus in horizontal camera views it can fail. This could be handled by a size correction based on the camera’s view angle and altitude.

The refresh rate for the foreground detection is also crucial. The ideal value depends on the frame rate, the ego-motion speed of the camera and the speed of the objects. The value was set in an experimental way.

The proposed algorithm is well suited for real-time application, because the computational complex is kept low. The proposed detection algorithm can be used in more complex, e.g. color based object detection as a pre filtering step to reduce the needed computations.

Figure 8(a) and 8(b) show the track results for urban environment containing a bridge that occludes the cars. The whole image is the mean value of the background, the brighter part is the current frame, the rectangles are the objects with their IDs, the colored dots are the tracks.

ACKNOWLEDGEMENT

This work was supported by the Hungarian Research Fund (OTKA 76159).

REFERENCES


Figure 8: Track results. The current frame is the brighter region, the objects are marked with rectangles and IDs, the colored dots are the tracks of different cars.