Extraction, categorization and unusual motion signaling of small moving objects

Levente Kovács, Ákos Utasi and Tamás Szirányi

Distributed Events Analysis Research Group, Computer and Automation Research Institute Hungarian Academy of Sciences, Kende u. 14-17, Budapest, Hungary

ABSTRACT

The paper presents an automatic approach for small moving object detection, categorization and unusual motion pattern signaling on camera feeds on sky background. The method uses a local blind deconvolution based foreground detector for small object mask and contour edge extraction, spatio-temporal localized histogram evaluation for object classification, and a hidden Markov model based evaluation for learning usual motions and signaling unusual motion patterns. The method is able to mask moving objects, fit them into learned categories and signal unexpected motion behavior.

Keywords: object detection, shape classification, unusual motion

1. INTRODUCTION

Image/video based segmentation of small moving objects is an ever-evolving research area. The hardest tasks involve moving object segmentation in noisy images, among cluttered backgrounds, in images taken with a shaking/moving camera, with changing lighting conditions, and a generally low resolution footage. In this paper we present a robust moving object detection, segmentation, extraction and categorization approach, on a sky background, which works in low resolution images, moving/shaking camera, clear or cloudy sky, and has a built-in unusual motion/event signaling function. Altogether, this new method or more exactly a combination of methods provides a suitable framework for detection and recognition of small to larger flying objects. Our approach differs from methods with a similar goal in that it has no object shape or class restriction, it is robust, fast, and extension - both of the detection and the recognition phase - is very easy.

In Ref.1 a small ship target detection method is presented, where point-like infrared images of small ships are processed to automatically detect ships on the sea level from a distance. Simple edge detection on a media-filtered image is used to extract possible ship locations. In other works3 small targets above a sea or sky background are extracted by infrared processing by using directional derivative operators and clustering. Elsewhere, low flying targets are segmented4 above the sea-sky line, by first locating the skyline, then using neighborhood averaging and directional Sobel operators to enhance the object boundaries. In Ref.10 a small object detection approach is presented where foreground separation and a local histogram-based evaluation is combined to obtain the moving object masks. In Ref.5 small moving patterns are extracted on an earth-sky background, by gradient operators, a multi-step adaptive thresholding and binarization to obtain an object mask. For noisy environments a wavelet-based object separation was used6 to find objects on a noisy background, by using a separate denoising pass, the method being used to process mammography images. Generally, cloudy skies2 are not taken into account, and the recognition/categorization of the detected silhouettes is not addressed.

The results presented in this paper involve Gaussian mixture model based object extraction with support for cloud separation and focus area extraction approach with local histogram evaluation. Then, a shape/contour extraction step is presented with indexing and retrieval solution for the recognition of the extracted objects. Finally, an unusual motion/event detection step is presented, which can be used to signal unusual object movements.

Further author information: E-mail: {levente.kovacs,utasi,sziranyi}@sztaki.hu, Web: http://web.eee.sztaki.hu
2. OBJECT EXTRACTION

The first step in the processing is the detection of the moving objects/targets. Then follows the extraction of their masks and contours, which will be used for two purposes. First, the object contours will be used for recognition of the extracted object/target, by using a contour-based shape retrieval approach. Secondly, the continuously extracted contours of the objects will be used to signal unexpected behavior, meaning sudden changes in motion on one hand, and detection of unusual motions on the other. In the following two subsections the methods used for detection and extraction of the moving objects’ masks are presented. The two approaches are based on local blind deconvolution and relative focus maps\(^7\) for detection of small moving targets, and on a Gaussian mixture model based approach - similar to Ref.9 but different in many ways, as will be presented - for more general moving object extraction with included sky-cloud-object separation. In our investigations we used hundreds of video captions mostly from air shows, model plane videos, and other online aircraft resources.

2.1 Extracting small object masks from focus maps

The relative focus map extraction approach we use here is based on Ref.8 and was first presented in detail earlier.\(^7\) The method is a way to extract a relative focus map from one image, giving a detailed separation of regions that are more in focus (i.e. foreground) from those that are less (i.e. background). Deconvolution techniques are generally used for reconstruction of degraded signals (images in this case) by optical devices and channel noise. The blurring function, which represents the distortion usually caused by the optics, is called the point spread function (PSF). When some knowledge about the possible PSF is available, high accuracy reconstructions are possible. When the PSF is unknown, or just estimated, blind deconvolution\(^11\) can be used to produce an estimated reconstruction.

In our approach, we use blind deconvolution to estimate the local blur of the images. We run a shift variant deconvolution scheme to obtain a local estimation of the image blurredness. Then, we use this information for image area classification, to differentiate those areas which are more in focus then the others. Thus we obtain a so called region of relevance based segmentation. The use of this approach for small object detection and segmentation was already presented,\(^10\) so we will include the details here only briefly.

Let us consider local image regions as individual small images, \(g\), which are formed by the convolution of the unknown original image \(f\) with the unknown point spread function \(h\). Given the observed degraded image \(g\), we search for the original image \(f\) which maximizes the Bayesian probability of observing \(g\) given the estimation of \(f\) of the form (indices point to image locations):

\[
P(f_i|g_l) = \frac{P(g_l|f_i)P(f_i)}{\sum_j[P(g_l|f_j)P(f_j)]} \tag{1}
\]

To estimate the local areas and the blur we use a localized version of Richardson’s iterative formula:

\[
\begin{align*}
    f_{k+1}(r) &= f_k(r) \left[ h_k(r) * \frac{g}{g_k}(r) \right] \\
    h_{k+1}(r) &= \frac{h_k}{\gamma} \left[ f_k(r) * \frac{g}{g_k}(r) \right]
\end{align*} \tag{2}
\]

where \(k\) denotes the iteration steps, \(r\) denotes the location vector of the respective image region, and \(\gamma\) is a weighting factor to constrain the PSF values between 0 and 1. The initial values of the iteration process are an average gray image for \(f_0\) with its DC value equal to the average of the observed \(g\), and circular constant unity for \(h_0\). A novelty of the approach is that we do not run a full deconvolution process. Instead, we only run a few iterations (typically between 5-10), just to be able to differentiate the image areas relative to each other. After the iterations we classify the areas relative to each other. The basis of this classification is an error measure based on testing the angle deviation of the estimations which is a function of the local reconstruction error and the local image contrast:

\[
E_r(g, g_k) = |\arcsin \frac{g - g_k}{|g - g_k|} g| \cdot \frac{C_r(g_r)}{\max_r \{C_r(g_r)\}} \tag{3}
\]
where $C_r(g_r)$ is the contrast of the region at location $r$. The resolution of the produced focus map can range from accurate pixel-level maps to large non-overlapping blocks. Fig. 1 shows an examples for an extracted focus map for a general image.

In the case of detecting small moving objects on a sky background, the above method can be run effectively to detect even the smallest observable moving pixels. In this case the iterations are run on the pixel level, producing the finest extraction map. Fig. 2 shows some examples. The drawback of this approach is, that the pixel level detection is a slow process, taking up to 1 minute for a 320 by 240 image, thus it is only practically useful for small resolution images, and for the detection of very small moving objects.

2.2 Object mask extraction from spatio-temporal intensity histograms

To alleviate the speed issues of the above method, and to provide a more robust approach, we present a pixel level statistics method which achieves real time speeds, works with objects with various sizes, and on changing and cloudy sky background as well.

There exist several pixel-level background estimation techniques,\textsuperscript{9,13} however these methods require a static camera to construct pixel-level statistical models, which makes them unfeasible for video sources with flying objects (e.g. aircrafts), where the camera is not static or sometimes even follows the moving airplane, also, the background can change from frame to frame (e.g. clouds, illumination changes, and so on).

Before introducing our foreground-background separation scheme, we define some assumptions about the scene:

- the moving object or foreground (the moving target, e.g. plane) is significantly smaller then the background (sky);
- the background is not completely homogeneous (clouds, etc.) but contains large homogenous areas.

The former assumption is usually fulfilled by most of the videos we investigated, however a large number of recordings contained clouds. That is why we also implemented a step to remove clouds from the foreground mask.

For background estimation we collected all pixel values (CIE L*u*v* uniform color space was used) in a moving time window and trained statistical models using maximum-likelihood estimation on the pixel values. Let $K$ denote the number of video frames with $h$ height and $w$ width, let $i_k$ denote a particular frame ($1 \leq k \leq K$) and $r$ the radius of the moving time window centered around $i_k$. $F_k^r = [f_{k-r}, \ldots, f_k, \ldots, f_{k+r}]$ denotes the frames selected around $f_k$ in the radius of $r$. Let $N$ denote the number of pixels in the time window, which can be calculated as $N = (2 \times r + 1) \times w \times h$. Let denote $P^{(k)} = [p_1, \ldots, p_N]$ the set of pixels of the frames.
2.2.1 Single Gaussian Background Model

From the sample set $P(k)$ we estimate the parameters of a Gaussian (with $\mu$ mean and $\Sigma$ diagonal covariance matrix, i.e. $\Sigma = \sigma I$, where $\sigma = [\sigma^L, \sigma^u, \sigma^v]$) as follows:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} p_i,$$
$$\Sigma = \frac{1}{N} \sum_{i=1}^{N} |\mu - p_i|^2.$$  (4)

Having the Gaussian model $\mathcal{N}(\cdot; \mu, \Sigma)$ the pixel $p_i \in P(k)$ is classified as background if the following condition holds:

$$\sqrt{\sum_c \frac{|p_i c - \mu c|^2}{\sigma^c}} < T$$  (5)

where $c \in L^* u^* v^*$ and $T$ is a threshold and is typically between 2.5 and 4 (the larger $T$ is the less restrictive the model is, hence more pixels will match the Gaussian).

2.2.2 Mixture of Gaussians

We extended the model of Sec. 2.2.1 to a multimodal statistical model by mixing several Gaussians (Mixture of Gaussians; MOG):

$$p(\cdot) = \sum_{l=1}^{M} w_l \mathcal{N}(\cdot; \mu_l, \Sigma_l)$$  (6)

where $M$ is the number of components, $w_l$ are the weights, $\mu_l$ and $\Sigma_l$ are the parameters (mean and covariance) of the Gaussians. The MOG was trained with $P(k)$ using the iterative Expectation-Maximization algorithm. 14

In the detection phase the distributions are ordered according to the ratio:

$$R_l = \frac{w_l \sqrt{\sum_c \sigma^c}}{\sum_l \sqrt{\sum_c \sigma^c}}.$$  (7)

The first $B$ distributions are chosen as the background model, where the $I$ parameter controls the modality. 9

$$B = \arg\min_{b} \left( \sum_{l=1}^{b} w_l > I \right) .$$  (8)

For a particular pixel $p_i$ we select first the matching distribution using Eq. 5 (if the pixel is matching more than one distribution then the one with the highest $R$ ratio is selected). The pixel is classified as background if the matching distribution is an element of the background model $B$.

This method can produce usable target object extractions if the sky background is fairly static and homogenous. In practical situations however, there are real possibilities of clouds appearing, lightning condition changes and so on. The next subsection will provide a solution for such cases, however Fig. 3 shows some example for foreground object target separation with the presented approach.
2.2.3 Cloud Removal

The presented background models usually learn the colors with the highest cardinality as backgrounds, hence small clouds or vapor trails can be misclassified as foreground (this is especially true for single Gaussian estimation Sec. 2.2.1).

For our cloud removal method we used the observation that real moving objects (planes) have a visible contour while clouds do not have this property. For video frame $f_i$ we use the horizontal and vertical Sobel operator, resulting in the $h_i$ and $v_i$ edge magnitudes. The total edge magnitudes are the absolute sum of $h_i$ and $v_i$, i.e. $e_i = |h_i| + |v_i|$. The output of the background-foreground separation step of Sec. 2.2.1 or Sec. 2.2.2 contains the number of $B$ connected components (blobs), and let $b_j$ denote the $j^{th}$ blob and $C_j = [c^1_j, \ldots, c^K_j]$ the set of $K$ contour points of $b_j$. Then, we define the energy of $b_j$ as

$$E_j = \frac{1}{K} \sum_{c^k_j \in C_j} \frac{1}{W} \sum_{(x,y) \in N^k_j} |e_i(x,y)|^2$$

(9)

where $N^k_j$ denotes the neighborhood of contour pixel $c^k_j$ and $W$ is the size of the neighborhood area.

After obtaining the blob energies $E = [E_1, \ldots, E_B]$, the energy values are sorted in ascending order - denoting the highest energy by $E_{max}$ - and linearly classified into $C$ number of classes/layers, thus:

$$E'_j = \frac{E_j}{E_{max}} \cdot C$$

(10)

The regions belonging to the highest energy classes/layers will be taken as targets, layers below will be the clouds and the lowest energy regions will be the sky background. Fig. 4 contains comparisons with other approaches, and Fig. 5 shows examples for extraction on cloudy sky background.

3. CONTOUR EXTRACTION AND CATEGORIZATION

The above presented methods produce segmented object masks from the input image sequences. This section presents a method for contour-based description of these objects, and using the contour-based representation for automatic recognition. The base of this method is to use a representation of the object contours for a scale and rotation invariant indexing and retrieval, thus providing the means for recognizing newly extracted objects. First, a dataset of previously extracted
object contours is constructed, and an index structure is built from these contours that is easily and quickly searchable for recognition. The base of comparison is a turning function metric, which compares the directionalities of the contour points to each other. In the following subsections we present the contour extraction, the indexing structure for these contours, and evaluation data for the performance of the presented indexing and retrieval approach. In the end, this method will be able to recognize extracted objects if the class of object is already present in the indexed data, and signal that it is a new object, if it is not recognized. Adding new object classes to the index structure is an easy task: contours belonging to the new class need to be added to the previous dataset and the index structure needs to be rebuilt.

For the purposes of this paper we deal with plane shape contours, but other shape types/classes can also be added with no changes in the methods or the algorithms contained.

### 3.1 Contour extraction

For contour representation, a list of coordinate points is extracted from the input shape images. The point list is extracted by a bug follower contour tracking method with backtracking, able to start from any random object point and follow a contour of the closed object shape, having any free form. The resulting point list contains hundreds to thousands of neighboring coordinate pairs, and sometimes points can be repeated in the case of certain small bulges present on the contour. Thus, a post-processing step eliminates coordinate re-occurrences. The resulting list will be finally stored.

During the indexing and retrieval process, a turning function based distance metric will be used (see Section 3.2) for which we also generate a smoothed/denoised contour version, thus considerably improving the indexing and retrieval times, and the also the recognition rate.

The smoothing procedure is also a lightweight approach. Let $C = \{ C_i = (x_i, y_i) \mid i = 0, n \}$ be the contour with $C_i$ as the contour points, $n$ the number of point coordinates, starting from a random contour position $(x_0, y_0)$, and ending with a neighbor of the start position. Then, the smoothed contour will be

$$C' = \{ C'_i = (x'_i, y'_i) \mid i = 0, m, m < n \} \quad (11)$$

where $(x_i, y_i) \in C'$ if $d((x_{i-k}, y_{i-k}), (x_{i+l}, y_{i+l})) < \varepsilon$, with $\varepsilon$ being a threshold and $k >= 2, l >= 2$. This is basically a primitive version of an outlier detector, working well enough for most practical purposes (see e.g. Fig. 6), yet remaining lightweight and very fast. The resulting contours are only used in the comparison distance metric in the indexing and retrieval phase, the points of the original contour are not changed. This simple smoothing does not disturb the features of the contour, but results in considerable improvement in recognition rates and indexing/retrieval times. Sample extracted contours are presented in Fig. 7.
Figure 8. Using the smoothed contour representations as the base of comparison increases the indexer’s speed by as much as 5 times on average.

### 3.2 Contour based indexing

The indexing step of the presented method takes as input the contours extracted in the previous section, and produces a serialization of an index tree structure, which will be used in the retrieval step.

The trees we use are customized BK-trees,\textsuperscript{12} which we will call BK*-trees. Traditionally BK-trees have been used for string matching algorithms. Essentially they are representations of point distributions in discrete metric spaces. That is, if we have feature points with an associated distance metric, then we can populate a BK*-tree with these points in the following way:

1. Pick one of the points as the root node, $R$.
2. Each node will have a constant number of $M$ child nodes.
3. A point $P_j$ will be placed into the child node $N_i$ ($i = 0...M−1$), if $i \cdot \frac{d}{M} < d(P_i, P_j) < (i + 1) \cdot \frac{d}{M}$ where $d$ is the maximum distance that two points can have (respective the associated metric) and $P_j$ is $P_i$’s parent node. Thus, a node will contain a point if its distance from the parent falls into the interval specified above; each node representing a difference interval $[i \cdot d/M; (i + 1) \cdot d/M]$.
4. Continue recursively until there are no more points left to insert.

As it is, this structure can be used to build quickly searchable index trees for any descriptor which has a metric. We also use this indexing structure for content-based video retrieval, as the base structure of indexed content descriptors, where multidimensional queries can be performed using the BK*-tree indexes of different feature descriptors. As the performance of the indexer, it takes 64.4 seconds to index our 6494 shape test database, as a single thread running on a 2.4GHz Intel Core2 CPU core. Of course, the index structure needs only be built rarely, when new shapes are added to the dataset.

Here we need to say a few words about the effects of using the above described simple smoothing/denoising step on the indexing performance. We compared the indexing of the raw contour point sets (in which only the reoccurring coordinate points have been eliminated) with the indexing where the smoothed contour versions were used as the input of the comparison function. Fig. 8 shows the difference, where we obtained a decrease of at least 5 times in the running time of the indexer, if the smoothing step was used.

The distance metric of the indexing - also used in querying the index structure in the retrieval phase - is a comparison of the results of turning/tangent function over the shapes. The output of the turning function is a 2D function representing the directions of the shape points over its contour positions. The turning function $\theta(s)$ is a standard way of representing a polygon. It measures the angle of tangent as a function of arc length $s$. The most important feature of this representation is that it is invariant for translation and scaling, but it is unstable when noise is present - this is why we implemented the smoothing step presented above. We measure the distance between two normalized (for scale invariance) turning function representations with the following formula:

$$D(P_1, P_2) = \min\left\{\int |\theta_1(s + t) - \theta_2(s) + \theta|ds\right\} \quad (12)$$

where $\theta$ is a translation parameter, which makes the distance metric rotation invariant, by comparing the shifted versions of the first function to the second function. Thus the distance becomes translation, scaling and rotation invariant.
Table 1. Classes, and the number of shape variations each class contains.

<table>
<thead>
<tr>
<th>class id.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>shapes in class</td>
<td>207</td>
<td>665</td>
<td>836</td>
<td>544</td>
<td>210</td>
<td>300</td>
<td>572</td>
<td>211</td>
<td>79</td>
<td>184</td>
<td>104</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>class id.</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>shapes in class</td>
<td>224</td>
<td>445</td>
<td>124</td>
<td>11</td>
<td>340</td>
<td>96</td>
<td>392</td>
<td>90</td>
<td>74</td>
<td>285</td>
<td>501</td>
</tr>
</tbody>
</table>

Figure 9. Examples for used object classes (each row from a class) and intraclass shape variations (elements of a row). We used a total of 22 classes with a total of 6494 different shapes.

3.3 Recognition evaluation

For the evaluation of the presented contour based recognition solution, we used a dataset of 6494 extracted plane and helicopter shapes, with a total of 22 different plane types. We treated each plane type as a separate shape class, and in the recognition phase a positive answer is only when the correct plane type is recognized.

Given the index tree generated with the above described method, the retrieval of shapes can be performed. The query of the retrieval is a shape image similar in contents of the ones indexed, that is a black and white image containing the contour of the query object. Table 1 shows how many different shape variations each class had in itself, and Fig. 9 shows examples for shape classes and in-class shape variations.

Given a content-based query ($Q$), the index tree is searched for similar entries:

1. If $d_0 = d(Q, R) < t$ ($t$ is user-adjustable), the root $R$ element is a result.
2. Let $N_i$ be the children of node $P_j$ ($P_0 = R$), and let $d_k = d(N_k, P_j)$ for the child $N_k$ where

$$
\begin{align*}
\{ & k \cdot \frac{d}{d_i} \in [d_{j-1} - t, d_{j-1} + t] \\
& (k + 1) \cdot \frac{d}{d_i} \in [d_{j-1} - t, d_{j-1} + t] \\
& k \cdot \frac{d}{d_i} \leq d_{j-1} - t \text{ and } (k + 1) \cdot \frac{d}{d_i} \geq d_{j-1} + t
\end{align*}
$$

then if $d_k < t$ the element from child $N_k$ is a result.
3. Repeat step 2 recursively until the whole tree is visited.
4. Sort all the results in the increasing order of their $d$ distances and return the ordered result list.

With real extracted plane shape data, we used the dataset of 6494 different shapes, belonging to 22 different classes, and we ran 80 queries on the indexed dataset. We evaluated the recognition rates of the retrievals, that is, in what percentage has the retrieved class been the same as the queried shape. Table 2 shows the results, where average recognition rates have been satisfactory. A query is an image sequence (a video) of a flying plane, and during the video, at every 10th frame a query is made over the indexed dataset, to determine the type of the currently extracted shape contour. In Table 2 a recognition value of 1 means perfect recognition (i.e. at every query the returned class has been correctly identified), and 0 would mean that the shape was not recognized. Fig. 10 shows sample query frames (i.e. actual frames that are captured and processed for mask extraction and recognition).
Table 2. Recognition rate (retrieval precision) data for the 80 different queries, from 6 different classes.

<table>
<thead>
<tr>
<th>target class id.</th>
<th>22</th>
<th>7</th>
<th>13</th>
<th>12</th>
<th>2</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>nr. of different queries</td>
<td>10</td>
<td>16</td>
<td>14</td>
<td>7</td>
<td>21</td>
<td>12</td>
</tr>
<tr>
<td>avg. recognition rate</td>
<td>1</td>
<td>0.85</td>
<td>0.6</td>
<td>0.88</td>
<td>0.85</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 10. Sample query frames from 6 different classes.

Fig. 8 showed that the indexing speed is greatly increased by the smoothing, but that in itself brings nothing, unless the retrieval performance is conserved. We will show that not only does the smoothing conserve the recognition rate of the retrieval, but it also improves the retrieval speed more than 10 times over. Fig. 11 shows retrieval times without and with smoothing, containing the retrieval times of 22 different queries, and their averages. The average running time for the raw retrieval over a 6492 piece test dataset is 27.2 seconds, which - by including the smoothing step - drops down to 3.3 seconds (which could also be improved by multithreading the searching algorithm).

4. UNUSUAL EVENTS

The above presented methods can provide extracted object masks and recognition data on the currently observed frames. In this section we will present methods that make use of these details to signal sudden changes in movement and unusual motion.

4.1 Signaling changes

The first method we present has the goal of signaling sudden changes in movement of the observed objects/targets. To achieve this goal, we use the extracted object details. First, we track changes in the profile ratios of the detected masks. Secondly, we construct a mask motion history of the objects, and track shifts in object orientation with regard of the actual mask and the history.

Profile ratios of the masks are the ratios of the width and height of the object, that is \( \gamma_1(i) = \frac{\text{width}_i}{\text{height}_i} \). If \( D_1(i) = |\gamma_1(i-1) - \gamma_1(i)| > \beta_1 \) we signal a change. \( \beta_1 \) is expressed in percentage, and a typical setting is \( \beta = 0.25 \).

The mask motion history of the current input is constructed by integrating the masks over the last \( N = 50 - 100 \) frames, producing statistics displayed as an image in Fig. 12. The extracted masks are scaled to the size of the first mask, then the history is constructed on a pixel level basis:

\[
\gamma_2(i) = \frac{1}{\varepsilon} \int_0^N \text{mask}_p \, dp
\]

Fig. 11. Comparing retrieval times without (left) and with (right) the smoothing step in the distance metric.
where $\varepsilon$ is a weighting factor for converting the integral history into grayscale intensities. A change is signaled when

$$D_2(i) = \text{diff}(\text{mask}_i, \gamma_2(i)) < \beta_2.$$  

Here $\text{diff}$ is a difference metric which calculates the overlap between the current mask ($\text{mask}_i$) and the high intensity history region of the current $\gamma_2(i)$, and $\beta_2$ is expressed in percentage, with a typical working setting of 0.35.

Signals are also shown visually (see Fig. 15), with flashing red signals on the current frames. Fig. 13 shows examples of changes signaled with the presented approaches.

4.2 Detecting unusual motion patterns

A Hidden Markov Model (HMM) is a finite state statistical model in which the system being modelled is assumed to be a Markov process (the states of a process of length $T$ is denoted by $Q = \{q_1, q_2, \ldots, q_T\}$). However, the states of process $Q$ cannot be directly observed, but it generates another process which is observable, $O = \{o_1, o_2, \ldots, o_T\}$. In this paper we will use the notations of Ref.15, thus the HMM with $N$ states ($S = \{S_1, S_2, \ldots, S_N\}$) is defined by $\lambda = (\pi, A, B)$, where $\pi = \{\pi_1, \ldots, \pi_N\}$ is the set of initial state probabilities, $A = \{a_{i,j}\}$ is the state transition matrix (the $i$th row contains the transition probabilities from $S_i$), and $B = \{b_i(.)\}$ is the set of emission probabilities. We use a single mixture of Gaussians (with $M$ components) to define the emission probabilities, i.e.

$$b_i(o_t) = \sum_{l=1}^{M} w_{i,l} N\left(o_t | \mu_{i,l}, \Sigma_{i,l}\right)$$  \hspace{1cm} (15)

where $w_{i,l}$ is the weight, $\mu_{i,l}$ is the expected value and $\Sigma_{i,l}$ is the covariance matrix of the $l$th component in the mixture. In our case the observations are the visual features generated from the video frames and are extracted as follows. From the $t$th image frame we extract the foreground mask as described in Sec. 2.2.2. The resulting mask is centered and resized to a $64 \times 64$ rectangle. We integrate these $64 \times 64$ images in a small temporal window (duration of half second) and the result will be our observation $o_t$.

4.2.1 Model Training

One of the most widely used methods for estimating the model parameters ($\pi$, $A$, $B$) is the maximum-likelihood estimation on a training observation sequence and can be performed by using the iterative Baum-Welch method.\textsuperscript{15} We extract the observation sequences $O = \{O_1, \ldots, O_K\}$ from the $K$ training videos. To initialize the model we create $N$ clusters from $O$ with $K$-means clustering. Let $C_i$ denote the set of elements in cluster $i$. Then the parameters of the emission probability of state $S_i$ is initialized as:

$$\hat{\mu}_{i,l} = \frac{1}{|C_i|} \sum_{c \in C_i} o_c, \hspace{0.5cm} \hat{\Sigma}_{i,l} = \frac{1}{|C_i|} \sum_{c \in C_i} (\hat{\mu}_i - o_c)^2$$  \hspace{1cm} (16)
Figure 14. HMM-based unusual event detection example. Left graph shows $-\log(P(o))$ values over time, curve behavior corresponding to changes between HMM states (unusual motion starting at frame 300). On the right we show frames from the “normal” phase (top), frames during the “unusual” behavior phase (middle), and some of the HMM state means from the detection process (bottom).

while the weights are randomized. Finally the Baum-Welch estimation is used to estimate the model parameters.

4.2.2 Event detection

In the case of HMMs, the probability of observing at time $t$ the $o_t$ which is generated by state $S_i$, given the previous state $q_{t-1}$, is

$$P(o_t, q_t = S_i | q_{t-1}) = \begin{cases} \pi_i b_i(o_t) & q_{t-1} = -1 \\ a_{q_{t-1}, i} b_i(o_t) & q_{t-1} \neq -1 \end{cases}$$

(17)

where $q_{t-1} = -1$ denotes that the previous state was unknown. Let $S_*$ be the state where the above probability is maximal, i.e.

$$S_* = \operatorname{argmax}_{S_i} \{P(o_t, q_t = S_i | q_{t-1})\} .$$

(18)

Then the probability that $o_t$ is usual, is defined as:

$$P_\lambda(o_t) = P(o_t, q_t = S_* | q_{t-1}) .$$

(19)

Fig. 14 shows an example where an image sequence of a flying plane was captured, which flew in a “normal” straight direction up until frame 300, when it started to turn and rotate. Thus “unusual” motion is detected from frame 300 and on. The figure shows masks from the “normal” phase and from the “unusual” phase, as well as some of the states of the 16 state HMM model that has been used for this example.

5. RESULTS

Fig. 15 shows actual frames from the output of a running processing, where current objects are detected, their contour is overlayed on the object, the current mask is displayed on the top right corner, below it the mask motion history is shown, and the change signaling values (in the case of an alert they flash in red). In the bottom of the frame the actual recognition is displayed (in the bottom row) and above the candidates with the highest probabilities (along with the recognition percentage values). At a quick glance one can see how confidently the actual object is recognized, how the mask extractor is performing, and the alerts regarding motion changes. The algorithm presented in this paper for object detection, mask extraction, recognition and unusual motion signaling runs at almost real-time speeds (8-12 frames per second) on an Intel Core2 CPU at 2.4GHz.

6. CONCLUSIONS

In this paper we have presented an approach for small to larger moving target detection, extraction and recognition for flying objects on homogeneous and cloudy sky background. Also, methods for signaling sudden changes in motion and unusual motion patterns of the extracted objects have been presented. The method can be used to extract masks and detect motion anomalies of flying objects, and can serve as a basis for flying object recognition, tracking and targeting. Future work is planned on model-based plane and missile tracking and recognition, on various cluttered backgrounds.
Figure 15. Frames from the actual output of the running system. For each: Middle: current input frame with detected mask contour overlay. Top right: current mask and mask motion history. Bottom: recognition details for the current object.

ACKNOWLEDGMENTS

This work was partially supported by the MEDUSA project of the European Defense Agency and the Hungarian Scientific Research Fund under grant number 76159.

REFERENCES