

Visual real-time detection, recognition and tracking of ground and airborne targets

Levente Kovács and Csaba Benedek

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

ABSTRACT

This paper presents methods and algorithms for real-time visual target detection, recognition and tracking, both in the case of ground-based objects (surveyed from a moving airborne imaging sensor) and flying targets (observed from a ground-based or vehicle mounted sensor). The methods are highly parallelized and partially implemented on GPU, with the goal of real-time speeds even in the case of multiple target observations. Real-time applicability is in focus. The methods use single camera observations, providing a passive and expendable alternative for expensive and/or active sensors. Use cases involve perimeter defense and surveillance situations, where passive detection and observation is a priority (e.g. aerial surveillance of a compound, detection of reconnaissance drones, etc.).

Keywords: object detection, recognition, tracking

1. INTRODUCTION

Visual surveillance for defensive and offensive purposes has been in continuous research and development during the last decades. Most of the research tries to target the automatism possibilities of such fields, since the number of used sensors increases exponentially, while the ever increasing quantities of data obtained from such sensors just can not be handled by sole manpower anymore. Such automatism possibilities include automatic target detection, tracking, recognition, unusual behavior detection, certain levels of situation assessment, with the goal of aiding Command and Control decision making. A tendency also easily recognized is the turn towards deploying a high number of cheap sensors instead of a low number of expensive nodes. Deploying passive sensors is also important in several situations. The above considerations were among the motivation for the work presented in this paper, i.e. creating methods that use cheap visual sensors (normal electro-optical cameras), and provide visual target detection, tracking and recognition capabilities, both on ground and aerial sensing nodes.

Regarding works for target detection and recognition, Lu et al.¹ presented a small ship target detection method, 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 works² 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 segmented³ above the sea-sky line, by first locating the skyline, then using neighborhood averaging and directional Sobel operators to enhance the object boundaries. Weng et al. present a flying target detection method⁸ based on infrared processing which is robust against weather conditions, but is not able to recognize the flying targets. Other works^{12,13} present more recent approaches to target tracking and recognition, based on infrared sensing for detection and adaptive feature selection for recognition. A robust infrared based approach¹² is presented for target detection and tracking, although it is somewhat constrained since it required static cameras. Wang et al.¹³ present target recognition on aerial imagery by a multi-feature method, sensitive to various geometrical shapes (circles, lines, etc.) of ground targets. In our previous works⁶ we have presented approaches for flying target detection and tracking, combined with shape-based recognition.

The novelties of the approach presented in this paper are that it presents visual, real-time methods (i.e. high information content at a low cost), it does not require static cameras (it is robust against camera movement and zooming), it can provide detection capabilities both for aerial and ground based targets (i.e. suitable for ground processing units and UAV's as well), and contains a shape and texture based recognition module. Target segmentations include adaptive foreground

Further author information: E-mail: {bcsaba, levente.kovacs}@sztaki.hu, Web: <http://web.eee.sztaki.hu>

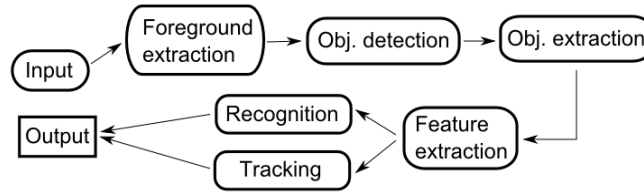


Figure 1: Basic steps of the presented detection and tracking approach.

and object extraction steps involving multi-layer Gaussian Mixture Modeling and object segmentation, view registration based on invariant feature point detection, ground object detection and tracking. Tracking is implemented on GPU for high resolution real-time processing. Recognition of the extracted targets is based on fused shape and texture information, providing estimations for the classification of the observed targets. Recognition evaluation is presented, by using a database of extracted and categorized object shapes, collected from real video sources.

Fig. 1 shows the steps of the presented methods and algorithms.

2. TARGET DETECTION

In this work we concentrate on two main types of targets for detection purposes. On one hand, aerial targets, i.e. flying objects (mostly planes), on the other hand, ground based targets (e.g. vehicles). The goal in both cases is to detect the moving objects, so as to provide a starting point for tracking (Section 4) and recognition (Section 3) tasks.

2.1 Detection of aerial targets

Detecting airborne targets is not a trivial task. Extensive research is continuously being conducted, as practical results are still far from being perfect. Works^{2,3,8} dealing with flying target detection have provided algorithms for various approaches, including infrared or hyperspectral imagery. In this case we concentrate on traditional electro-optical generic cameras, for multiple reasons: they are cheap, provide passive detection capabilities, and have a high visual information content.

One of the hardest steps in aerial object detection and segmentation is the robust separation of the objects from the background, which in this case includes the sky, clouds, smoke, vapor trails, etc. In earlier works⁶ we introduced a single Gaussian and a Gaussian Mixture Model (GMM)⁴ based approach for separating flying objects, an approach which we now use for extracting various sized flying objects in real-time, with adaptivity to support changes in lighting conditions, sky color changes, presence of clouds, single or multiple present objects. In the following we will present a short overview of the flying object segmentation approach, for further detail see Ref.⁶

The reason behind developing an extended GMM-based background model was that most other approaches require a static camera for robust modeling. In our approach the camera movements are not restricted, the only assumptions are that the object is smaller than the background and that the background is not completely homogeneous. 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 . $F_k^r = [f_{k-r}, \dots, f_k, \dots, f_{k+r}]$ 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 $P^{(k)} = [p_1, \dots, p_N]$ denote the set of pixels of the frames.

To model the background in an environment where the camera movements are not restricted and the background can consist of different clutter (e.g. clouds, vapor trails, smoke, etc.), we use a mixture of Gaussians (MOG) approach, where the model has the following form:

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

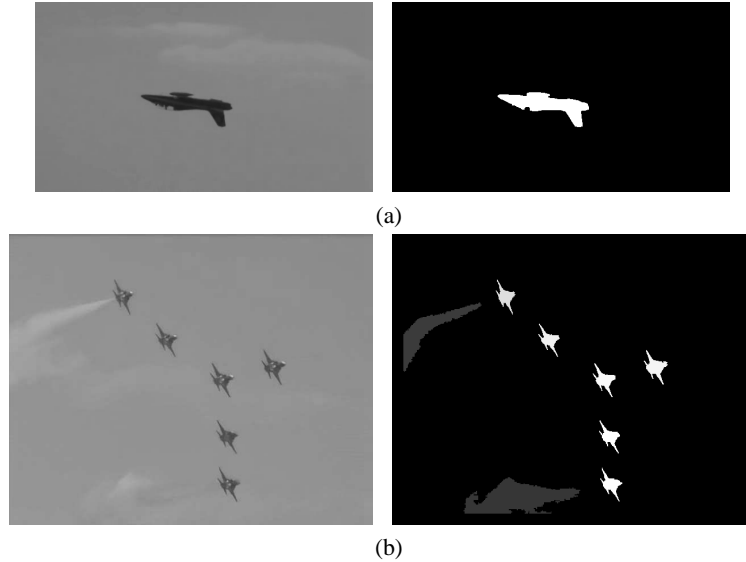


Figure 2: Examples for extracted foregrounds without (a) and with (b) background clutter (white: foreground, black: background, gray: clutter belonging to the background layer).

where M is the number of components, w_l are the weights, μ_l and Σ_l are the parameters (mean and covariance) of the Gaussians $\mathcal{N}(\cdot; \mu, \Sigma)$. Denoting with $p_i \in P^{(k)}$ the i^{th} pixel of frame P , the mean and covariance parameters of the Gaussians are $\mu = \frac{1}{N} \sum_{i=1}^N p_i$ and $\Sigma = \frac{1}{N} \sum_{i=1}^N |p_i - \mu|^2$.

The MOG model is trained with $P^{(k)}$ sample set, using iterative Expectation Maximization. During the segmentation the distributions are ordered according to the ratio R_l and the first B distributions are chosen as the background model, where the I parameter controls the modality:⁹

$$R_l = \frac{w_l}{\sqrt{\sum_c \sigma_l^c}}, \quad B = \underset{b}{\operatorname{argmin}} \left(\sum_{l=1}^b w_l > I \right). \quad (2)$$

For a particular pixel p_i we select first the matching distribution, then the pixel is classified as background if the matching distribution is an element of the background model B . For removing clouds from the background we used the observation that real moving objects (planes) have a clear contour while clouds do not. For video frame P^i we extract horizontal (h_i), vertical (v_i) and the total edge magnitudes: $e_i = |h_i| + |v_i|$. The output of the foreground separation step produces B connected regions. If b_j denotes the j^{th} region and $C_j = [c_j^1, \dots, c_j^K]$ the set of K contour points of b_j , then the energy of b_j will be:

$$E_j = \frac{1}{K} \sqrt{\sum_{c_j^k \in C_j} \frac{1}{W} \sum_{(x,y) \in N_j^k} |e_i(x,y)|^2}, \quad (3)$$

where N_j^k denotes the neighborhood of contour pixel c_j^k and W is the size of the neighborhood area. After obtaining the region energies $E = [E_1, \dots, E_B]$, the energy values are linearly classified into layers. The regions with the highest energy will be taken as targets. Fig. 2 shows examples of extracted foregrounds.

2.2 Detection of ground targets

The detection of ground based moving objects from an airborne camera (e.g. from an UAV) involved multiple steps. The presented approach consists of:

- frame stabilization and registration,
- foreground extraction,

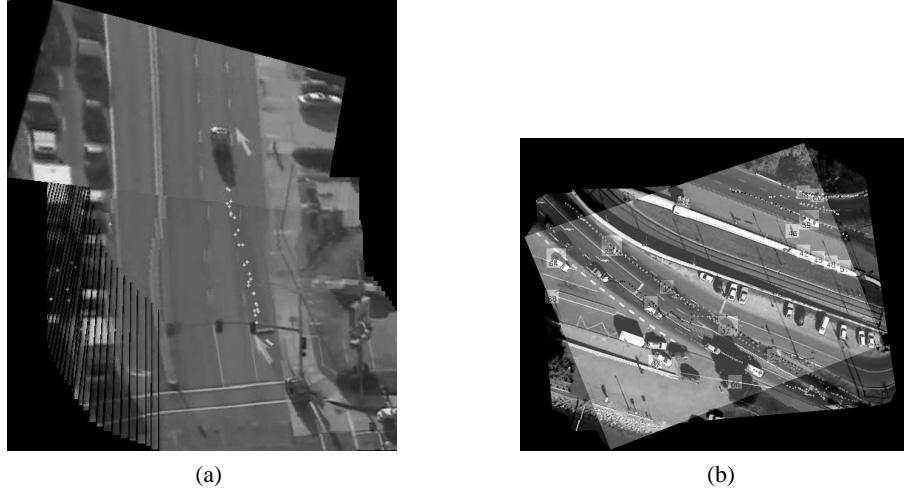


Figure 3: Samples from the process of registered and warped frames from video sequences.

- object detection.

The frame stabilization and registration step involves the calculation of the homography matrix¹⁴ between consecutive frames. The perspective transformation between two images taken of a plane surface can be described by the homography matrix H . One point p_0 is transformed to p_1 by the following equation:

$$p_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{H13} \\ H_{21} & H_{22} & H_{H23} \\ H_{31} & H_{32} & H_{H33} \end{bmatrix} \times \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}. \quad (4)$$

We use a feature based registration approach to calculate these matrices. We extract feature points that can be used to align two images S_1 and S_2 , by finding the transformation H that fits to the feature points, thus $S_2 = H \cdot S_1$. In case of aerial images the transformation can not 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.

We use the Harris¹⁵ corner detector which is suitable in man-made environments where corners are abundant, also, it is less computationally expensive than the other feature point detectors (e.g. SIFT, SURF). Then, corresponding points from the following are searched by Lucas-Kanade optical flow, which yields the positions of the feature points on the next image, thus the transformation between the frames can be calculated. Having obtained the transformation, frames can be aligned into a common coordinate system. For that we need two homographies, one for consecutive frames ($H_{n,n-1}$), and one for aligning frames to a reference frame ($H_{n,0}$), where:

$$H_{n,0} = H_{n,n-1} \cdot H_{n-1,n-1} \cdot \dots \cdot H_{1,0}. \quad (5)$$

The current frame gets transformed into the coordinate system of the reference image by the homography H :

$$I_d(x, y) = \left(\frac{H_{11}x + H_{12}y + H_{13}}{H_{31}x + H_{32}y + H_{33}}, \frac{H_{21}x + H_{22}y + H_{23}}{H_{31}x + H_{32}y + H_{33}} \right) \quad (6)$$

where I_d denotes the pixels of the destination and I_s of the source frame. Fig. 3 shows examples for warped frames.

The foreground-background separation step is based on modeling the background and extracting the moving foreground segments. The background image is modeled and updated in the common coordinate system, calculating the pixel-by-pixel running average and variance of the consecutive aligned video frames. Mixture of Gaussians (MoG) approaches⁴ cannot be used in the case of UAV images, since usually because of the fast camera motions the observed samples (per surface

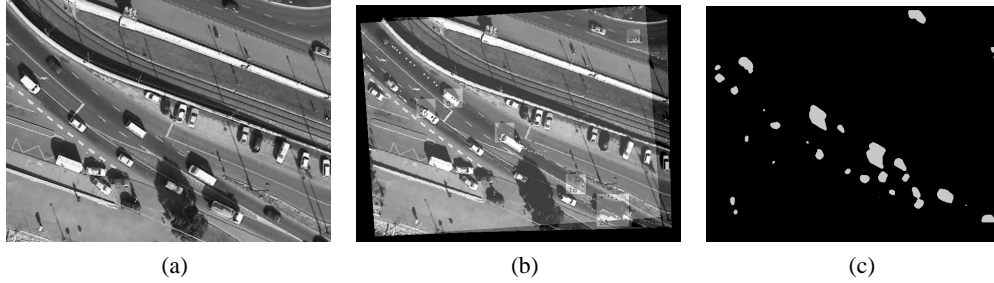


Figure 4: Example of extracted foreground mask: (a) input frame, (b) current registration, (c) current foreground mask.

point) are not enough to create a MoG model. The mean value and variance for the frames are (with α being a refresh rate constant):

$$\bar{x}_n = (1 - \alpha)\bar{x}_{n-1} + \alpha x_n \quad (7)$$

$$\sigma_n^2 = (1 - \alpha)\sigma_{n-1}^2 + \alpha(x_n - \bar{x}_n)(x_n - \bar{x}_{n-1}) \quad (8)$$

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:

$$d(p_n) = \sqrt{\sum_{i=1}^3 \frac{(p_{n,i} - \bar{p}_{n-1,i})^2}{\sigma_{n-1,i}^2}}. \quad (9)$$

Fig. 4 shows examples for such detected foreground masks, with object areas shown in white.

3. TARGET RECOGNITION

The above presented steps provide detected and extracted foreground object blobs that can be used for further processing. The next step is to extract features from these blobs that can be used for classification and recognition purposes. Multiple previous works deal with general and specific object feature extraction. Here we focus on feature based recognition that has the goal of extracting and comparing object features for target recognition purposes. In one work aerial images (taken from airborne platforms or satellites) are the basis of ground target recognition¹³ where specific feature sets are pre-calculated that are sensitive to certain object parameters and can help in detection. Such features include geometric descriptors (e.g. rectangular, circular features) and context features (how certain features generally relate to each other). Generally, object based recognition methods use one or more features for the categorization steps, and this is also the path we follow in this work. The features we use are shape (based on extracted object contours), texture and histogram (object internal parameters) descriptors, which provide a fast and generic object categorization capability.

Traditionally, contours/shape descriptors have been extracted and compared with a series of methods, including Hidden Markov Models, comparisons based on Scale Invariant Feature points, tangent/turning functions,¹⁹ curvature maps, shock graphs, Fourier descriptors,¹⁰ polar coordinates¹¹ and so on. They all have their benefits and drawbacks, regarding computational complexity, precision capabilities, implementation issues, robustness and scalability. Other edge based approaches include Chamfer distance based methods⁹ for recognizing objects through smaller shape fragments. These methods generally work by converting the high level metrics into a distance function based comparison, which in turn works by using some kind of chain code shape representation. They incorporate scale and rotation invariance on the high level.

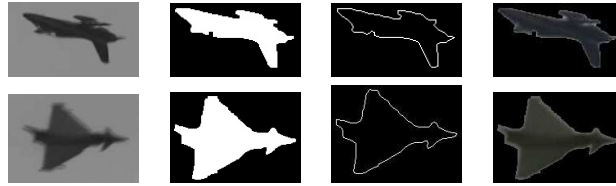


Figure 5: From left to right: section of the input frames with object; extracted object blob; extracted object contour; region of the original frame with texture.

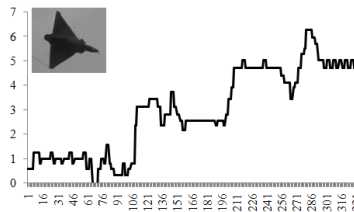


Figure 6: An object and its contour's turning function representation.

3.1 Feature extraction, indexing

We use two object features as a basis for recognition tasks. One is the shape/contour of the detected objects, another is the texture information of the image region of the object. For shape extraction we use a simple blob shape extraction step by going over the boundary points of each object and storing all the contour points. For texture information, we use the standard MPEG-7 homogeneous texture descriptor.²⁰ Fig. 5 shows some examples of extracted shapes from the binary foregrounds obtained in above presented sections and which regions will be used for texture feature extraction.

For the purposes of this work we collected an object dataset containing 27 classes of object shapes, 9140 shapes in total. The shapes have been automatically extracted from real life videos of flying planes.

For the comparison of the features, two distance metrics are used. In the case of shapes, we use the turning function representation for speed considerations, where the contour coordinate lists get transformed into a 2D function based on the directions of the contour at all of its points. Fig. 6 shows an example. Comparison is performed by calculating the point wise distance between scaled and shifted versions of two turning functions, thus obtaining a scale and rotation invariant distance. In the case of texture, we use the standard homogeneous texture descriptor distance metric from the MPEG-7 descriptor reference software. This texture descriptor extracts local statistics of the image, by filtering the region with orientation and scale sensitive Gabor filters, and computing means and standard deviations of the results in frequency domain. Comparison is done by calculating the Euclidean distance between two feature vectors containing mean and deviation values. Experiments²⁰ have shown an average accuracy of 77% for this descriptor.

In this work we use the above two features in a complementary manner. First, a shape based query is performed for an object, then a texture based query, and the two results are weighted in the favor of the shape descriptor, which generally performs higher. The role of the texture feature is to aid the recognition step in cases when the shape based recognition performance is low.

To be able to perform content based queries (either with shape or texture), we need an indexing structure, on which the queries can be run. Thus, we build index trees for the objects, containing shape and texture features. The trees we use are customized BK-trees.⁵ Traditionally BK-trees have been used for string matching algorithms. Essentially they are representations of point distributions in discrete metric spaces. For classical string matching purposes, the tree is built so as to have each subtree contain sets of strings that are at the same distance from the subtree's root, i.e. for all e leaves below sub-root r the $d(e, r) = \varepsilon$ is constant. In our case, the used structure contains tree nodes that can have an arbitrary number of children (N), where the leaves below each child contain elements for which the distance d falls in a difference interval: $d(e, r) \in [\varepsilon_i; \varepsilon_{i+1})$, where $i \in [0, N] \cap \mathbb{N}$. The distance intervals in the child nodes (denoted by $\varepsilon_i, \varepsilon_{i+1}$ above) depend on the maximum error E_{max} that the feature-dependent distance metric can have, more specifically, $\|\varepsilon_{i+1} - \varepsilon_i\| = E_{max}/N$, thus the difference intervals are linearly divided buckets.



Figure 7: The aggregated first 3 recognitions are continuously presented (numbers at the end show current recognition probabilities).

Table 1: The object classes and the number of objects in each class.

class	1	2	3	4	5	6	7	8	9	10	11	12	13	14
objects	108	505	207	316	665	854	549	682	300	603	325	258	79	184
class	15	16	17	18	19	20	21	22	23	24	25	26	27	total
objects	104	233	451	120	124	12	344	644	395	96	75	285	622	9140

The indexing of the used 9140 shape dataset takes about 80 seconds (on a single thread, Core i7 2.8GHz). Retrieval time (time to return results for a query) are in the range of 80–400 ms. Since we are trying to stay as close as realtime as possible, the recognition steps are not performed on each processed frame, but as a parallel process in an SMP architecture. Basically one thread processed the foreground separation and object extraction steps, another thread processes the tracking, and new threads are spawned at specific intervals (typically every 20-30 frames) to query the indexed dataset. One query thread is run for each object present on the frame, thus the retrieval speed depends on the number of the objects and on the number of available threads on the CPU. The recognition results are aggregated over the processing of the input, and a statistics is built from the first 3 most probable recognitions and continuously presented to the user (example in Fig. 7).

3.2 Retrieval, performance

For evaluating the recognition rates and performance of the presented approach, we use the above mentioned object dataset with 9140 objects (Table 1 shows the number of classes and number of objects, and Fig. 8 shows a few examples from the first 5 classes).

For testing the recognition rate of the above described approaches, we used the above dataset, and picked 9 query videos of planes whose classes were present in the dataset, but the query videos were not included in the indexed database. That is, although the query videos contained objects from known object classes, but the specific objects in the query videos were not part of the dataset. Testing is performed by the following steps:

1. Extract objects from the query video frames.
2. Obtain object shape and texture features.
3. Perform two queries:
 - (a) Query based on shape (recognition based on shape information).
 - (b) Query based on combination of shape and texture (recognition based on shape+texture information).
4. Keep track of first 3 results of each query, accumulate them throughout the query video, and continuously show the 3 most probable recognition candidates.
5. At the end of the query video, take the candidate with the higher probability as the overall best recognition (if an object has been correctly recognized 7 out of 10 queries, that will be a 0.7% recognition rate).

Fig. 9 shows recognition rates for 9 query videos containing a total of 1573 frames, for shape only and shape+texture based recognitions. Fig. 9a and 9b show the same results for lower frequency queries (querying every 20 frames) and higher frequency (every 10 frames) respectively. These results show that the average recognition rate for shape based queries is

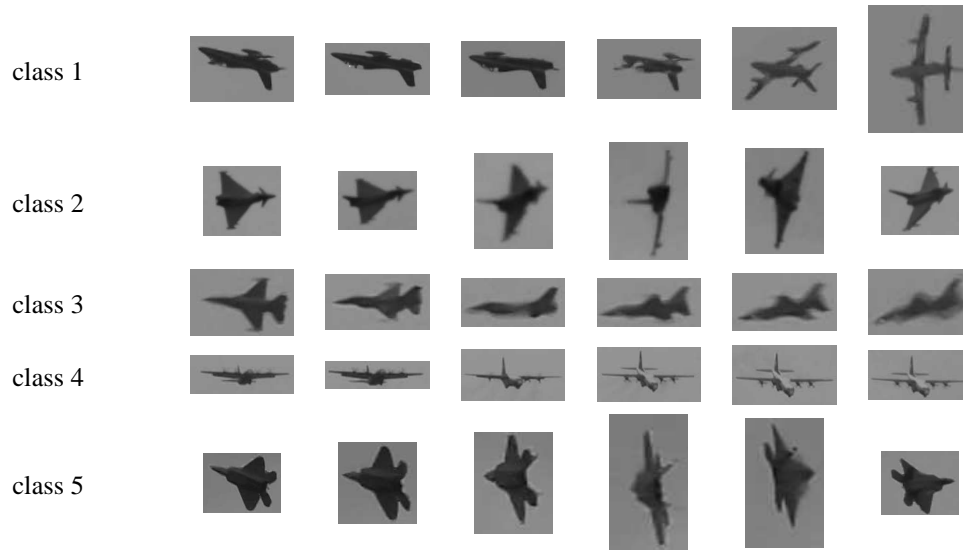


Figure 8: A few examples from dataset classes 1–5 (from 27 classes in total).

Table 2: Numerical recognition rates for different query videos, for shape only and shape+texture based queries.

id of query video	1	2	3	4	5	6	7	8	9
frames in video	322	134	67	55	134	132	130	143	75
nr. of queries	15	6	2	2	6	6	6	6	3
shape only recog. rate	0.71	0.8	0	0	0.66	0.67	0.5	0.6	0.5
shape+texture recog. rate	0.87	0.8	0.8	1	0.62	0.69	0.6	0.83	1

0.49 while for combined shape+texture it is 0.8 for Fig. 9a (for higher frequency querying in Fig. 9b these values become 0.85 and 0.83 respectively). That means, that in almost all cases texture information can help the shape-based recognition process. When comparing the lower and higher frequency querying cases (see Fig. 10), it turns out that querying more frequently can make the recognition rates better (both standalone and combined) – but we have to keep in mind that more frequent querying means (much) higher computational complexity and stress on the real time system, especially in the case of multiple objects being present (as we wrote above, retrieval time can be in the range of 80–400 ms per object, and the queries run in parallel threads). Table 2 also shows numerical recognition rates for the above used query examples

The point we wish to make with these results is that the goal is to make less frequent, but better performing queries for object recognition, and for such goals the shape+texture combined recognition solution performs better than shape-based recognition alone.

Of course, there are cases when the recognition can fail totally. Such misclassification generally happens when the query object is very similar to objects belonging to other classes than its own (at least from a certain viewing angle). Here we show two examples, where both the shape and shape+texture based queries returned 0% recognition rates, but the reported results are very close (visually) to the queries.

4. TARGET TRACKING

In both research and practice, there are quite a number of object tracking algorithms for different purposes. Most of them use either robust background modeling and/or object features for obtaining objects and descriptors to serve as the base for keeping track of their movements. E.g. the FPSS algorithm¹² presents a combined approach where background modeling combined with image filters for equalization and noise suppression are used to obtain the foreground containing the moving objects, then using a predictor to build a destination probability density map which aids the tracking process.

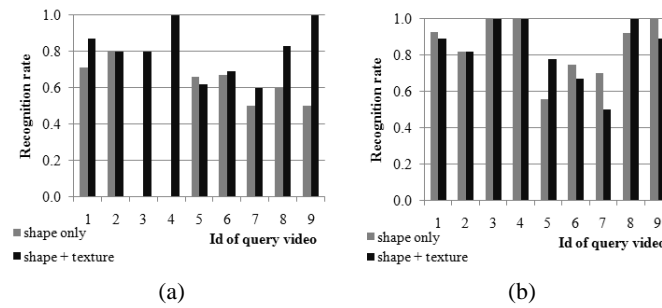


Figure 9: Comparison of shape only and shape+texture based recognition rates for a query lower frequency (every 20 frames) (a) and higher frequency (every 10 frames) (b).

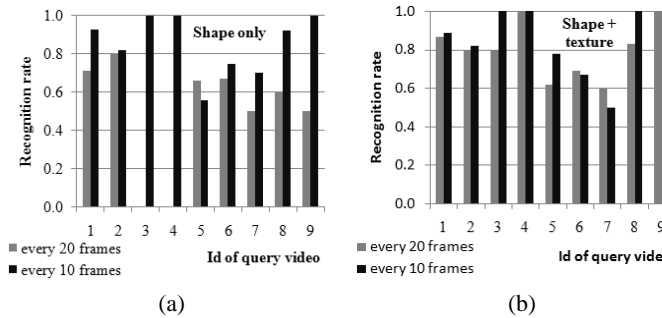








Figure 10: Comparing shape and shape+texture based recognition rates w.r.t. increased querying frequencies. (a) shows shape-based rates for queries performed every 20 and 10 frames, while (b) shows similar data for shape+texture based queries. Higher frequency querying can result in improved rates, but also increases computation time.

Table 3: Example for failed recognition. Queries from the two samples (with id 10 and 11) resulted in falsely reporting the objects in the videos as belonging to classes 11 and 27, instead of 19, 22. The sample objects show that in these cases the in-dataset objects and the query objects have been very similar (causing the misclassification).

id of query video		10	11
ground truth class of query objects		19	22
reported recognized class (false positives)	shape	11	27
	shape+texture	11	27
query object sample			
reported (false positive) object sample			
true positive object sample (from dataset)			

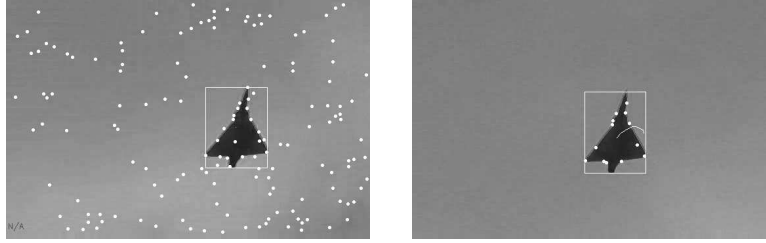


Figure 11: Left: frame with extracted feature points. Right: a later frame where only object-relevant points are retained.

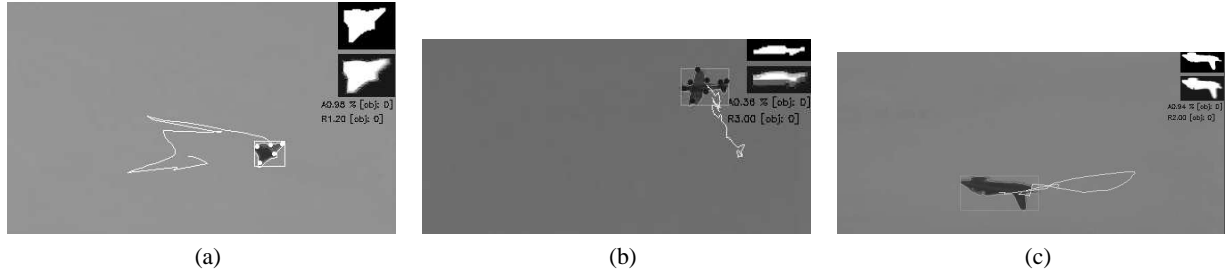


Figure 12: Example of flying object tracking, with both object and camera movements.

For tracking of the extracted *flying object* blobs we use a GPU-based KLT tracker implementation based on⁷. The idea behind the KLT tracking approach is to extract stable feature points from an object, then keep track of their movements through continuous frames based on Lucas-Kanade optical flow estimations. On a 470-series Nvidia GPU, the performance of the tracker is above 60 frames per second for PAL resolution videos, for multiple targets. It works by tracking feature points obtained by the gradient-based Harris¹⁵ corner point extractor. Feature point tracking in itself is mostly local image processing, thus can be highly parallelized intra-frame, which this approach exploits.

The original tracker is basically a feature point tracker. Our addition to this method is the object-based tracking extension. This works by obtaining the feature points and tracks of the GPU-based tracker, combining tracks corresponding to the same object - since the object regions have already been extracted -, then dropping the tracks that do not correspond to a known object, and combining (by calculating the means of valid tracks of feature points) the tracks belonging to one object into one track, which will be assigned and stored with that object. Fig. 11 contains an example frame (left) showing all the feature points extracted from the original tracker, and a following frame (right) where only feature points relevant to the object are retained. Fig. 12 shows results on the same video of the used object tracker.

In the case of tracking *ground objects* from aerial video sources (e.g. from UAVs), we use a different approach, mostly because of the differences in types of background and contents of the videos (e.g. occlusions, more clutter, higher noise, more dynamic backgrounds). This method has been first introduced in¹⁸, with the goal of UAV based area surveillance application.

Object detection is processed for each frame independently, thus the separate detections have to be registered along the frames to yield object tracks. The difficulty is that usually the number of object detections for consecutive frames can vary also in the case of perfect detections, i.e. objects enter and leave, get occluded. To handle the disappearing and later reappearing objects Kalman filtering is used, which 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.

Motion in this environment can be described by the following equations:

$$\vec{x}_k = \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \end{bmatrix} + w_{k-1} \quad (10)$$

$$z_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \vec{x}_k + v_k \quad (11)$$

where x_k is the position in one direction, z_k is the measured position, w_{k-1} is the process noise, v_k is the measurement noise.

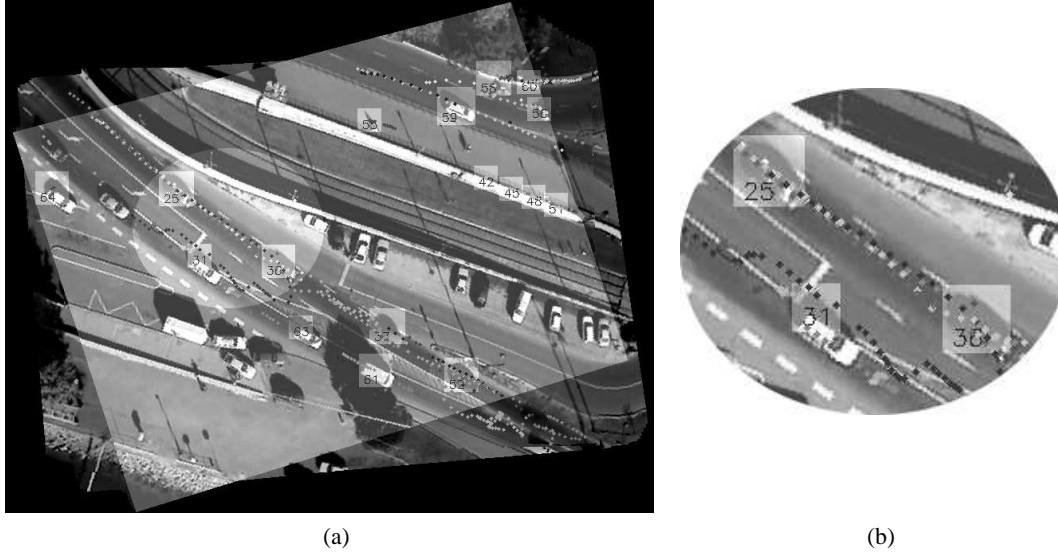


Figure 13: (a) Example frame showing tracked objects (in rectangles with associated id numbers) and tracks (small dot series with different colors). (b) highlighted region zoomed in.

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^{16,17} solves this assignment problem in polynomial time. We solve the assignment problem with a greedy algorithm, which is computationally simple and gives good results. The cases when the greedy algorithm fails can be neglected, since this is caused by objects being present for short periods of time (i.e. noise).

A score matrix S_{nk} is calculated based on the Euclidean distance of the predicted and detected positions and the objects color histograms. The elements of the matrix are fitness values which describe how good the objects from the previous frames match the current ones:

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

where d_{pos} is the Euclidean distance of positions, ϑ is a weighting constant, O_i is previous frame object i , D_j is the detected object and d_{hist} is the histogram distance:

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

where \hat{p} is the predicted position of object O .

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. Otherwise, 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.

Fig. 13 shows an example of tracked ground objects.

5. CONCLUSIONS

In this paper we have presented approaches for visual aerial and ground based object detection, tracking and recognition based on shape and texture features. The goal was to use fast solutions running on cheap commodity hardware, providing passive (visual) detection and recognition capabilities for area surveillance and protection. The flying object detection methods are suitable for detection, tracking and recognition of airborne targets from ground-based sensor stations, while the ground object detection and tracking methods can be integrated on flying UAVs for ground area surveillance and ground

based target tracking applications. Further work involves more robust recognition development, speed enhancement, real life integration onto UAV sensor platforms. The work has been partially supported by the MEDUSA project of the E.D.A.

REFERENCES

- [1] Lu, J. W., He, Y. J., and Li, H. Y., "Detecting Small Target of Ship at Sea by Infrared Image," Proc. IEEE Intl. Conf. on Automation Science and Engineering, 165-169 (2006).
- [2] Zheng, S., Liu, J. and Tian, J. W., "An SVM-based small target segmentation and clustering approach," Proc. Intl. Conference on Machine Learning and Cybernetics (6) 3318-3323 (2004).
- [3] Zhang, S., Liu, W., and Xue, X., "Research on tracking approach to low-flying weak small target near the sea," Optical Information Processing, Proc. SPIE 6027, 962-968 (2006).
- [4] Stauffer, C. and Grimson, E., "Learning patterns of activity using real-time tracking," IEEE Tr. on Pattern Recognition and Machine Intelligence, 22(8), 747-757 (2000).
- [5] Burkhard, W. and Keller, R., "Some Approaches to Best-Match File Searching," Proc of CACM (1973)
- [6] Kovács, L. and Utasi, Á., "Shape and motion fused multiple flying target recognition and tracking," Proc. of Automatic Target Recognition XX, SPIE Defense, Security and Sensing, SPIE v. 7696, 769605-1-12 (2010).
- [7] Zach, C. Gallup, D. and Frahm, J.M., "Fast Gain-Adaptive KLT Tracking on the GPU," Proc. of Computer Vision on GPUs (CVGPU) Workshop at IEEE CVPR, 1-7 (2008).
- [8] Weng, T.L., Wang, Y.Y., Ho, Z.Y. and Sun, Y.N., "Weather-adaptive flying target detection and tracking from infrared video sequences," Expert Systems with Applications, 37(2), 1666-1675 (2010).
- [9] Shotton, J., Blake, A. and Cipolla, R., "Multi-scale categorical object recognition using contour fragments," IEEE Tr. on Pattern Analysis and Machine Intelligence, 30(7), 1270-1281 (2008).
- [10] Wong, W. T., Shih, F. Y. and Liu, J., "Shape-based Image Retrieval Using Support Vector Machines, Fourier Descriptors and Self-Organizing Maps," Intl. Journal of Information Sciences, 177(8), 1878-1891 (2007).
- [11] Frejlichowski, D., "An Algorithm for Binary Contour Objects Representation and Recognition," Proc. of ICIAR, Lecture Notes in Computer Science vol. 5112, 537-546 (2008).
- [12] Chan, A.L., "A robust target tracking algorithm for FLIR imagery," Proc. of Automatic Target Recognition XX, SPIE v. 7696, 769603-1-11 (2010).
- [13] Wang, Y., Yao, W., Song, Y., Sang, N. and Zhang, T., "Multi-class target recognition based on adaptive feature selection," Proc. of Automatic Target Recognition XX, SPIE v. 7696, 769609-1-9 (2010).
- [14] Szeliski, R., "Image alignment and stitching: a tutorial," Foundations and Trends in Computer Graphics and Computer Vision, 2(1), 1-104 (2006).
- [15] Harris, C. and Stephens, M., "A combined corner and edge detector," Proc. of Alvey Vision Conference, 147152 (1988).
- [16] Taixe, L. L., Heydt, M., Rosenhahn, A. and Rosenhahn, B., "Automatic tracking of swimming microorganisms in 4D digital in-line holography data," Proc. of WMVC, 1-8 (2009).
- [17] Kuhn, H. W., "The Hungarian Method for the assignment problem," Naval Research Logistic Quarterly, 2, 83-97 (1955).
- [18] Mátyus, G., Benedek, Cs. and Szirányi, T., "Multi target tracking on aerial videos," Proc. of ISPRS Workshop on Modeling of Optical Airborne and Space Borne Sensors, 11-13 (2010).
- [19] Latecki, L. J., and Lakamper, R., "Application of planar shape comparison to object retrieval in image databases," Pattern Recognition, 35(1), 15-29 (2002).
- [20] Manjunath, B. S., Ohm, J. R., Vasudevan, V. V. and Yamada, A., "Color and texture descriptors," IEEE Trans. on Circuits and Systems for Video Technology, 11(6), 703-715 (2001).