Flying Target Detection and Recognition by Feature Fusion

Levente Kovács, Andrea Kovács, Ákos Utasi, Tamás Szirányi

Distributed Events Analysis Research Laboratory

Computer and Automation Research Institute

Hungarian Academy of Sciences

Kende u. 13-17, 1111 Budapest, Hungary

Phone: +36-1-2796106 Fax: +36-1-2796292

`levente.kovacs@sztaki.hu, andrea.kovacs@sztaki.hu`

`utasi@sztaki.hu, sziranyi@sztaki.hu`

Abstract. This paper presents a near-realtime visual processing approach for automatic airborne target detection and classification. Detection is based on fast and robust background modeling and shape extraction, while recognition of target classes is based on shape and texture fused querying on a-priori built real datasets. The presented approach can be used in defense and surveillance scenarios, where passive detection capabilities are preferred (or required) over a secured area or protected zone. © *2012 Society of Photo-Optical Instrumentation Engineers. DOI: 10.1117/1.OE.51.11.117002*

Subject terms: object segmentation, target detection, classification

Document compiled December 17, 2012

1 Introduction

Visual detection, recognition, classification and tracking of stationary of moving targets are among the most active research areas in computer vision and image processing fields. Applications built on the results of these research areas are constantly sought to be deployed for both defensive and offensive scenarios, including civilian and military use. For civilian applications, wide area surveillance, crowd and traffic monitoring and target tracking are the most important fields, while for military applications troops and asset protection, region of interest surveillance, target detection and tracking are probably the most important scenarios. Aiding such tasks by intelligent and automatic visual processing is important since such methods can aid the detection, recognition and alerting tasks of security personnel. Also, visual processing sensors/nodes can provide a means for passive detection (without requiring active signals), thus making them harder to detect and disarm in case of sensitive scenarios.

This paper presents a solution for one aspect of the above described wide range of possibilities, focusing on automatic airborne target detection and classification. The presented approach can be used in defense and surveillance scenarios, where passive detection capabilities are preferred (or required) over a secured area or protected zone. The goals are to automatically detect and recognize the class of observed flying targets from varying angles, views, size and environmental conditions, running on commodity hardware.

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 median filtered image is used to extract possible ship locations. In other works[2] small targets above a sea or sky background are extracted by infrared processing by using directional derivative operators and clustering. Deng et al.[3] present small target detection in infrared, based on self-information maps and locally adaptive background thresholding and region growing, producing robust detection results. While infrared processing can help the detection task (especially during the night), it is not suitable for generic classification because of the low resolution and less visual information.

Some target detection (without tracking and classification) methods[4,5] present object detection based on multiscale color and saliency information on single frames/images, detecting outlier regions based on local features as target candidates with good results, but not suitable for realtime video processing. Lia et al.[5] present a visual missile-like target detection approach based on image segmentation, motion analysis for target region selection, and a target boundary extraction step, validating on videos captured from 3D simulations. The K-means based segmentation and the histogram-based target region extraction is not suitable for our purposes, since we have highly dynamic backgrounds with changing light, moving clouds and vapor trails with free camera motions, and the grayscale histograms do not contain enough information for such scenarios.

Elsewhere, low flying targets are segmented[6] above the sea-sky line, by first locating the skyline, then using neighborhood averaging and directional Sobel operators to enhance the object boundaries. Bibby et al.[7] resent a color-based tracking approach on a sea background using a stabilized camera on a moving platform. Here they use color and gradient based Mean Shift tracking, without object detection or classification. Such an approach could be integrated with our proposed method for tracking purposes.

Wenga et al. present a flying target detection and tracking method[8] in infrared. Here the goal is detection and tracking, without recognition/classification. Image complexity, number of objects and number of other large areas (e.g. cloud objects) is taken into consideration, and detection is performed depending on the weather condition (clouds or clear skies). Also, clouds are separated based on histogram analysis, assuming white cloud color. On the other hand, our approach does not use or depend on such information, any background clutter (clouds, vapor trails, smoke, independent of their color) get automatically discarded based on their non-relevance as target candidates (based on their features), and the method is independent on the presence of such clutter. Other infrared based approaches[9] also exist for target detection and tracking, although somewhat constrained since it requires static cameras, without classification capabilities. Wang et al.[10] present infrared target recognition on aerial imagery by a multi-feature method, sensitive to various geometrical shapes (circles, lines, etc.) of ground targets. Blasch et al.[11] present an approach for visual and infrared target tracking for day and night applicability, with the goal of keeping the target IDs in cluttered environments for robust long term tracking, that can be used for robust tracking after the objects are detected and categorized.

Noor et al.[12] present a model generation approach for object recognition, using multiple views of objects to build a model database. They start with SIFT descriptors for relevant corner point extraction, used to build a region-neighborhood graph that is used for object matching. In our case we needed more robust interest point extraction because of the variances in backgrounds and viewing angles, thus we present and use a more robust point extraction approach.

In our previous works we have introduced small target detection[13] and flying target tracking.[14] The current paper builds on these results and presents a more thorough investigation concentrating on shape and texture fused target recognition with extensive evaluation.

The novelties of the work presented in this paper are:

- Using an object detection and extraction method based on an enhanced interest point detection approach, which is robust against noise and clutter in the scene (e.g. clouds, vapor trails, other interference like illumination changes), which is a new approach with respect to classical multi-layer background/foreground modeling methods, shifting the complexity from the background modeling to a faster process of robust boundary point and contour segment extraction.
- Presenting results in recognition of the extracted object classes based on the combination of their shape and texture information, using a simple and fast approach based on indexing using BK-trees and a turning function based metric combined with MPEG-7 texture features.
- As opposed to several other methods that use pre-segmented datasets for training a classifier, in our case no manual segmentation or annotation is being done during the training and the detection/recognition process. The dataset for building the initial shape index is gathered automatically by the same algorithm that is used during recognition, by running it on videos containing the target classes. The used video dataset contains real life captures of flying targets on real background, without any simulation. The goal was to concentrate on real life applicability and real environmental properties (varying illumination, cluttered background, multiple simultaneous targets, etc.).

Fig.1 shows the main algorithmic steps of the presented method. The first step of the approach applies a novel *boundary points detection* technique, detailed in Sec. 2. In the second step we create a Markovian *foreground-background separation* method for efficient object extraction, described in Sec. 2.2. Finally, for target classification we extract shape and texture features from the segmented objects, see Sec. 2.3 and Sec. 3.

We will present results and evaluation of the object detection, extraction and recognition phases, showing the viability of the approach.

2 Target Detection and Feature Extraction

The first step towards the recognition phase is object extraction and object feature extraction. In this section we will present these two steps. First, object extraction will be described, based on a contour detection method that applies a novel interest point detector. Then, the later used object feature extraction step will be presented.

For the extraction of the object silhouettes we propose a two-step probabilistic method, which achieves high computation performance, and works with multiple objects with different properties (e.g. size, color,

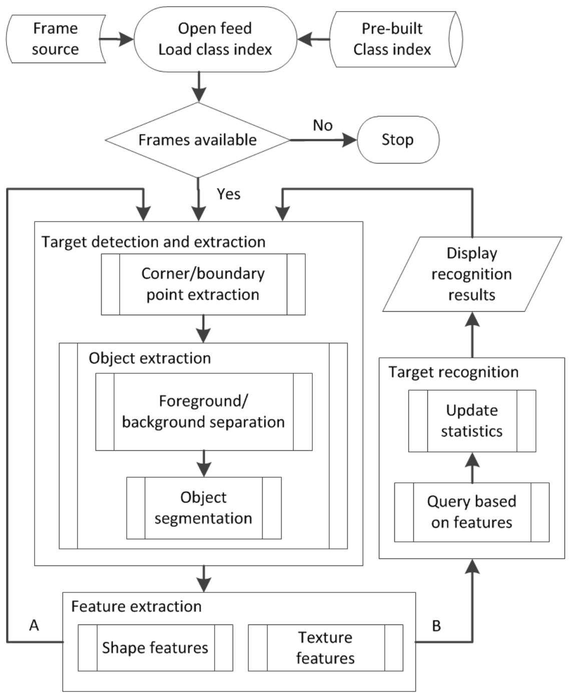


Fig. 1: Sequence diagram of the presented solution. Branches A and B run in parallel.

shape, texture), on a changing cloudy sky background. Before introducing the proposed method we list our assumptions:

1. the camera is not required to be static, but the consecutive video frames should contain overlapping parts;
2. the moving objects are smaller than the background part of the frames;
3. the background is not required to be completely homogeneous (e.g. clear skies), but should contain large homogeneous areas (which can be sky, clouds, vapor trails, etc.).

2.1 Feature point extraction

According to our assumptions, the aim is to find small (relative to the entire frame size) objects in front of a non-homogeneous background (that may contain clouds, sky regions, etc.) in the image I_t at time t. The first step of localizing foreground objects is to extract interest/feature points. The challenge in detecting such points is that the contours of the moving foreground objects are usually of low contrast, and contain high curvature parts. Therefore, traditional point detectors, like the Harris corner detector,[15] cannot represent them accurately (see Fig. 2a). To compensate for such drawbacks, we use a modification of the Harris detector, which was introduced earlier[16] and was applied for complex object contour recognition with parametric active contour algorithms.

The Modified Harris for Edges and Corners (MHEC) method described here, adopts a modified characteristic function and it is able to emphasize not only corners, but edges as well, therefore it is more suitable for contour features than the traditional method.

The original Harris detector is based on the principle that intensity has large changes in multiple directions simultaneously at corner points. The method defines the R characteristic function for classifying image regions:

$$R = \mathrm{Det}(M) - k * \mathrm{Tr}^2(M) \ , \tag{1}$$

where Det is the determinant and Tr is the trace of the M Harris matrix (see Eq. 2), and k is a coefficient, usually chosen around 0.04. The M Harris matrix is defined as:

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} \ , \tag{2}$$

where $A = \dot{x}^2 \otimes w$, $B = \dot{y}^2 \otimes w$ and $C = \dot{x}\dot{y} \otimes w$, with $\dot{x}$ and $\dot{y}$ denoting the approximation of the first order derivatives and w is a Gaussian window.

M describes the shape at an image point and its eigenvalues (denoted by λ_1 and λ_2) give a rotation invariant characterization of the curvatures in the small neighborhood of the point. Eigenvalues separate different regions: both of them are large in corner regions, only one of them is large in edge regions and both of them are small in homogeneous (flat) regions.

In our case, when emphasizing edge and corner regions simultaneously, we exploit the fact that they both have one large eigenvalue, therefore $L = \max(\lambda_1, \lambda_2)$ is able to separate homogeneous and non-homogeneous regions in the image.[16] Let $b_i = \{[x_i - r, x_i + r] \times [y_i - r, y_i + r]\}$ mark a window surrounding a specific $p_i = (x_i, y_i)$ pixel. p_i is the element of the $\mathbf{C}$ contour point set, if it satisfies the following condition:

$$\mathbf{C} = \left\{ p_i : L(p_i) > T_1 \ \text{AND} \ p_i = \operatorname*{argmax}_{q \in b_i} L(q) \right\} \ . \tag{3}$$

p_i is an extracted feature point, if it's $L(p_i)$ value is over a given T_1 threshold and it is a local maximum in b_i. The T_1 threshold is calculated adaptively by Otsu's method.[17] The $\mathbf{C}$ set of contour points is shown in Fig. 2b. It is important to note, that MHEC also emphasizes parts that were dismissed by the original Harris implementation, like the frontal part of the left plane.

Now the $\mathbf{C}$ point set is defining contour points in the image belonging to different flying objects or background. The next step is to separate point subsets of various objects, while eliminating the points of the background.

The separation process of contour point subsets is based on the included points' connectivity in the Canny edge map.[18] If two contour points are connected by an edge in the edge map, then they are supposed to belonging to the same object. The following graph representation formalizes this assumption: a $\mathbf{G} = (\mathbf{C}, \mathbf{N})$ graph is described with the $\mathbf{C}$ vertex set and the $\mathbf{N}$ edge set, where $\mathbf{C}$ is defined by Eq. 3 and $\mathbf{N}$ is built according to the connectivity of the vertices (points) in the edge map.

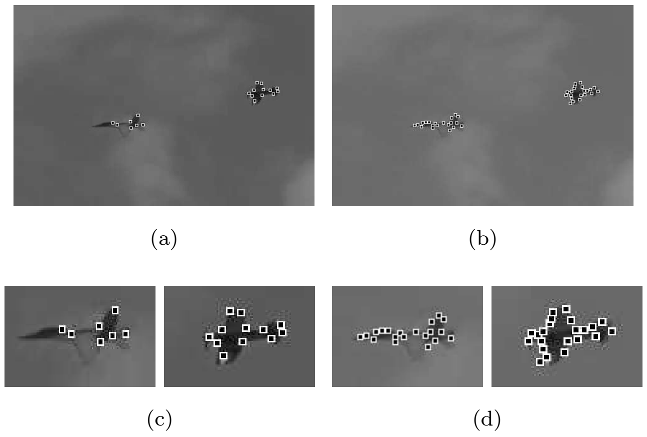


Fig. 2: Contour point detection. (a): Original Harris corner detector;[15] (b): Proposed MHEC point detector. (c-d) show the respective objects zoomed.

Let E mark the dilated Canny edge map of the image (see Fig. 3a), where pixels representing the detected edges get a value of 1, others, representing the background, are 0. Two given vertices $v_i, v_j \in \mathbf{C}$ are connected by an edge in $\mathbf{N}$ if they satisfy the following conditions:

1. $E(v_i) = 1$; v_i is an edge point in E.
2. $E(v_j) = 1$; v_j is an edge point in E.
3. A finite path of pixels in E with value 1 exists between v_i and v_j (i.e. they are connected by an edge in E).

After performing this procedure for all vertices, the $\mathbf{N}$ edge set is defined.

Now the $\mathbf{G}$ graph will contain K disjoint subgraphs (denoting the k^{th} with $\mathbf{G}^k$) with contour point sets $\mathbf{C}^k$ representing separate objects:

$$\mathbf{C}^k = \left\{c_1^k, \ldots, c_{N_k}^k\right\} \ , \tag{4}$$

where N_k is the number of contour points in $\mathbf{G}^k$. Then the following conditions are satisfied by $\mathbf{C}^k$ point subsets:

$$\mathbf{C} = \bigcup_{k=1}^{K} \mathbf{C}^k; \quad \mathbf{C}^i \cap \mathbf{C}^j = \emptyset \quad \forall i, j \ . \tag{5}$$

Subgraphs containing only a few points are supposed to indicate noise or background, therefore we filter out a $\mathbf{G}^k$ subgraph, if the number of its points is smaller than an n threshold ($N_k < n$). In our work, we applied $n = 4$. After this filtering, the remaining set of contour points representing K' flying target is given as:

$$\mathbf{C}' = \bigcup_{k=1}^{K'} \mathbf{C}^k \ . \tag{6}$$

Fig. 3b shows the two separated contour point sets representing the two objects.

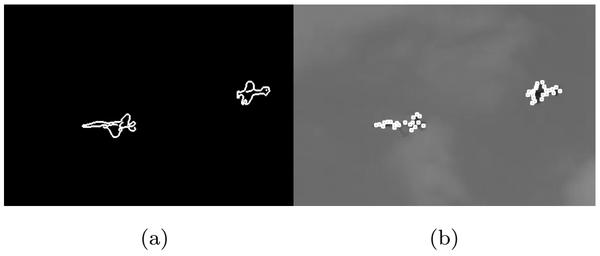


Fig. 3: Object separation. (a): Canny edge map; (b): Separated object contour points.

We have to emphasize here, that while the MHEC point detection process produces more points than e.g. the Harris detector, this is a positive property of the approach, since we use and exploit the higher number of points to find good, connected boundary point locations in the graph analysis process. Thus we arrive to a boundary point extraction, which is robust and at the same time aids in the removal on non-objects from the processed frames.

In situations where objects are so close that they visually occlude one another, the objects might not get separated and the result would be a blob containing both objects. In such situations the recognition phase (Sec. 3) will give a false classification of the blob. However, as it will be described later, the recognition step builds a continuous statistics of the detected classes of objects in time, and if the objects will visually separate later, then their classes will be updated. Also, a tracker using the outputs of this paper could help in separation of such objects.

2.2 Object Extraction

In the previous step we obtained corner and edge points which directly relate to the boundaries of flying objects. Having this information, we create a spatio-temporal multi-modal background model from the pixels which are not related to any of these objects. This background model is used for preliminary object extraction to roughly segment the objects' silhouettes from the background (see Sec. 2.2.1). Then we create a separate appearance model for each object, and we refine the silhouette in a Markov Random Field (MRF) framework using the background and the object appearance (i.e. foreground) models with an additional interaction constraint to express the similarity between neighboring pixels (see Sec. 2.2.2).

Formally, we denote by $\mathbf{C}^k = \left\{c_1^k, \ldots, c_{N_k}^k\right\}$ the set of corner and edge points (referred to as boundary points in the rest of the paper) of the kth object detected by the MHEC detector presented above, where N_k denotes the number of boundary points. Let $\boldsymbol{\mathcal{C}}_t = \left\{\mathbf{C}_t^1, \ldots, \mathbf{C}_t^{K_t}\right\}$ denote the collection of all boundary point sets at time t, where K_t denotes the number of boundary point sets. Hereafter we denote by $\mathbf{C}_t^k \in \boldsymbol{\mathcal{C}}_t$ one element of the collection, where $1 \leq k \leq K_t$. Moreover, we also make the following formal definitions:

- $\mathbf{S}$ denotes the pixel lattice of an image;

- $\mathbf{X} = \{x_s | s \in \mathbf{S}\}$ is the set of pixel values of an image, i.e. x_s is 3-tuple in a given color space;
- $\mathbf{L} = \{\mathrm{b}, \mathrm{f}\}$ denotes the set of two class labels *background* (label b) and *foreground* (i.e. object, label f) respectively;
- $\boldsymbol{\Omega} = \{\omega_s | s \in \mathbf{S}\}$ denotes the labeling of the pixels of an image, where $\omega_s \in \mathbf{L}$ is the label of a particular pixel s;
- $p_l(s) = P(x_s | \omega_s = l)$ is the conditional probability density function of label $l \in \mathbf{L}$ at a given pixel s.

In our implementation we use the CIE L⋆u⋆v⋆ uniform color space, i.e. $x_s = [x_{\mathrm{L}}(s), x_{\mathrm{u}}(s), x_{\mathrm{v}}(s)]$.

2.2.1 Preliminary Segmentation

There exist several pixel-level background estimation techniques[19],[20] however these methods require a static camera to construct pixel-level statistical models, which makes them unfeasible for video sources with flying objects, where the camera is typically not static or sometimes even follows the moving airplane, also the background can change from frame to frame, e.g. clouds, or vapor trails might be visible, and illumination changes can also occur.

Due to the above problems we create one global spatio-temporal background model at each timestep using the pixel values in a small moving time window. Here we use the pixels which do not relate to object silhouettes. In our method this global background $p_{\mathrm{b}}(s)$ is modeled with a finite mixture of M_{b} Gaussians (MoG), i.e.

$$\begin{aligned} p_{\mathrm{b}}(s) &= \sum_{k=1}^{M_{\mathrm{b}}} \omega_{\mathrm{b},k} \cdot p_{\mathrm{b},k}(s) \\ &= \sum_{k=1}^{M_{\mathrm{b}}} \omega_{\mathrm{b},k} \cdot \mathcal{N}(x_s | \mu_{\mathrm{b},k}, \Sigma_{\mathrm{b},k}) \ , \end{aligned} \tag{7}$$

where $p_{\mathrm{b},k}(s) = \mathcal{N}(x_s|\mu_{\mathrm{b},k}, \Sigma_{\mathrm{b},k})$ denotes the 3D Gaussian density function, i.e.

$$p_{\mathrm{b},k}(s) = \frac{\exp\Big(-\frac{1}{2}(x_s - \mu_{\mathrm{b},k})^{\mathrm{T}} \Sigma_{\mathrm{b},k}^{-1} (x_s - \mu_{\mathrm{b},k})\Big)}{(2\pi)^{3/2} \cdot |\Sigma_{\mathrm{b},k}|^{1/2}} \ . \tag{8}$$

Moreover, to speed up the segmentation process we assume a diagonal covariance matrix, i.e. $\Sigma_{\mathrm{b},k} = \sigma_{\mathrm{b},k}^2 \cdot I$, where I is the 3×3 identity matrix.

At this point we utilize the $\boldsymbol{\mathcal{C}}_t$ collection of boundary point sets at time t as follows. Let b_t^k denote the bounding box of the kth point set $\mathbf{C}_t^k \in \boldsymbol{\mathcal{C}}_t$, where the size of the box is slightly enlarged, and $\mathbf{B}_t = \{b_t^1, \ldots, b_t^{K_t}\}$ denotes the set of bounding boxes at time t. Moreover, let $\mathbf{S}'_t \subset \mathbf{S}_t$ denote the pixels, which do no lie within any bounding boxes of $\mathbf{B}_t$, i.e.

$$\mathbf{S}'_t = \{s \in \mathbf{S} \ : \ s \notin b_t^k, \ k = 1, \ldots, K_t\} \ , \tag{9}$$

and let N_t denote the number of these pixels, i.e. $N_t = |\mathbf{S}'_t|$. Furthermore, let r denote the radius of a small

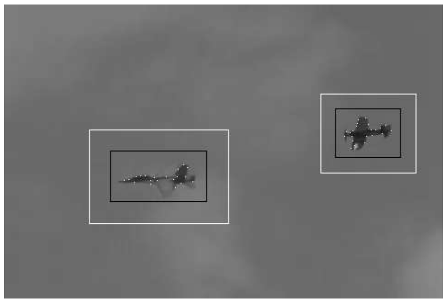


Fig. 4: The inner (darker color) rectangles represent the bounding boxes of the detected edge and corner points. Pixels outside the inner rectangles are used for *training* the global background model. Pixels outside the outer rectangles (lighter color) are always classified as background in the preliminary segmentation step.

temporal window, and $\mathbf{S}'_t(r)$ the union of pixels of all $\mathbf{S}'_t$ sets in the r radius, i.e.

$$\mathbf{S}'_t(r) = \bigcup_{\tau=-r}^{+r} \mathbf{S}'_{t+\tau} \ . \tag{10}$$

From Eq. 9 and Eq. 10 it is obvious that

$$|\mathbf{S}'_t(r)| = N_t(r) = N_{t-r} + \cdots + N_t + \cdots + N_{t+r} \ . \tag{11}$$

Fig. 4 demonstrates this process using one frame from a video with two objects (i.e. $K_t = 2$), where the inner rectangles (darker color) represent the b_t^1 and b_t^2 bounding boxes enlarged in both direction with 20%. The pixels outside these boxes are used to form the $\mathbf{S}'_t$ set of pixels at time t. Having the $\mathbf{S}'_t(r)$ set of training samples in the temporal window with radius r we calculate the maximum likelihood estimate of the global background MoG model using the Expectation-Maximization technique (EM).[21]

The estimated background MoG model $p_{\mathrm{b}}(\cdot)$ can be directly used for foreground-background separation to determine the pixel labels of $\mathbf{\Omega}$. However, in this case the separation is based only on the background model with the risk that some parts of the object might get classified as background. Thus we use this scheme for creating a preliminary classification only, and the results obtained are used for refining the segmentation by using a separate local appearance model for each object (see Sec. 2.2.2).

To obtain a preliminary object extraction, we use the estimated MoG background model of Eq. 7 to separate the foreground from the background. In our method a pixel s is classified as background (i.e. $\omega_s = \mathrm{b}$) in either of the following two cases:

1. the pixel's position is *"far"* from the objects;
2. the pixel's value does not *"match"* the background model.

Otherwise the pixel is classified as foreground, i.e. $\omega_s = \mathrm{f}$. For the first case we simply use another rectangle around the boundary points, which has the size of the bounding box enlarged with 50% (illustrated by the outer rectangles with lighter color in Fig. 4), and we classify the pixels outside these rectangles as background.

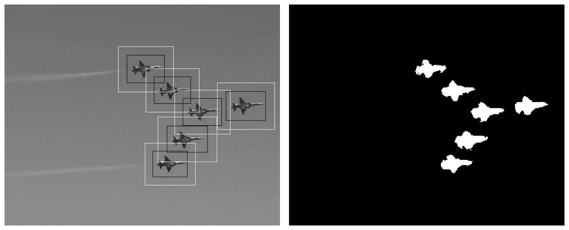


Fig. 5: Situation where rectangles of objects overlap (left), and the produced mask.

In the second case we considered pixel s *"matching"* the background model if for any Gaussian component of the mixture the following inequality holds:

$$\sqrt{\left(x_s - \mu_{\mathrm{b},k}\right)^{\mathrm{T}} \Sigma_{\mathrm{b},k}^{-1} \left(x_s - \mu_{\mathrm{b},k}\right)} < T \ , \tag{12}$$

where threshold T typically takes values on the $[2.5; 3.5]$ range, and we used a constant $T = 2.5$ value in our experiments. The output of the preliminary segmentation of the Fig. 4 input is demonstrated in Fig. 6a, where black color represents the pixels classified as background ($\omega_s = \mathrm{b}$), and white pixels denote the foreground pixels ($\omega_s = \mathrm{f}$). We can observe that significant part of the right wing of the left airplane has been classified as background, since it matches the color of the clouds. This issue will be addressed in the next section by using local appearance models.

Situations might occur when two or more objects are so close to each other that their rectangle regions overlap, i.e. parts of one object becomes visible in the region of another object. Since the above described segmentation step is pixel-based and multi-modal, in such situations the overlapping part of the other object will be extracted as separate blob (will be classified as foreground, but as a separate region), and its remainder parts will become a different blob, extracted from another - neighboring - rectangular area. However, this causes no problems, since the output of the segmentation is a frame with white masks where it is not important whether a mask is a result of separate smaller blobs, or was detected as a single larger blob. Fig. 5 presents such a situation.

2.2.2 MRF Segmentation

The preliminary segmentation process produces initial object silhouettes which might be broken or some parts of the objects be misclassified as background. In the next step we refine these silhouettes, and we follow a Bayesian approach for the classification of background and foreground pixels. For this step we need statistical information about the a priori and conditional probabilities of the two classes and the observable pixel values. In addition, we use a MRF to model the spatial interaction constraint of the neighboring pixels.

In our case the appearance of a foreground object is modeled by a MoG with the following conditional probability function:

$$p_{\mathrm{f}}(s) = \sum_{k=1}^{M_{\mathrm{f}}} \omega_{\mathrm{f},k} \cdot \mathcal{N}\left(x_s | \mu_{\mathrm{f},k}, \varSigma_{\mathrm{f},k}\right) \ , \tag{13}$$

where we use smaller number of components in the mixture than we used in the global background model, i.e. $M_{\mathrm{f}} < M_{\mathrm{b}}$. To estimate the model parameters, we use the pixels within the b_t^k bounding box, which were classified as foreground in the preliminary segmentation step (highlighted in Fig. 6b). The parameters of the MoG are obtained again by EM.[21]

According to the MRF model, the optimal labeling $\widehat{\mathbf{\Omega}}$ is be expressed as

$$\widehat{\mathbf{\Omega}} = \operatorname*{argmin}_{\mathbf{\Omega}} \sum_{s \in \mathbf{S}} - \log P\left(x_s | \omega_s\right) + \sum_{r,s \in \mathbf{S}} V\left(\omega_r, \omega_s\right) \ , \tag{14}$$

where the spatial constraint is realized by the $V(\omega_r, \omega_s)$ function (also known as smoothing term). Here we use $V(\cdot,\cdot)$ to penalize those pixels whose class labels differ from those of their neighboring pixels. In our model $V(\omega_r, \omega_s) = 0$ if ω_r and ω_s are not neighbors, otherwise

$$V\left(\omega_r, \omega_s\right) = \begin{cases} 0 & \text{if} \quad \omega_r = \omega_s \ , \\ \beta & \text{if} \quad \omega_r \neq \omega_s \ , \end{cases} \tag{15}$$

where $\beta > 0$ is a penalizing constant.

Here we utilize the conditional probability functions of the two classes (background and foreground respectively) at a given pixel s, defined in Eq. 7 and Eq. 13. Finally, for solving the MRF problem we optimize our Eq. 14 energy function using a graph cuts based optimization algorithm,[22,23,24,25] which gives a good sub-optimal solution in a few iteration of steps. Fig. 6c shows the final result obtained by MRF segmentation. We can observe that the number of falsely classified pixels significantly decreased, e.g. the wings of the left airplane.

Fig. 7 presents final outputs of the full presented object extraction approach above, compared to classical approaches[26] which are less robust against background noise (e.g. clouds, vapor trails might be classified as foreground).

2.3 Feature Extraction

As later described, the recognition step of the presented approach is not real time, thus in our implementation the queries against the shape dataset run in parallel with the main processing thread, updating the current object class after each step. A lower number of updates would generally mean lower recognition precision, thus, to alleviate this issue, we try to make the recognition itself more robust. We accomplish this by using a combination of two features - shape and texture - and we show that this solution helps in keeping (or even improving) the recognition accuracy even in the case of lower update frequencies. This in turn reduces the overall computational requirements of the whole process.

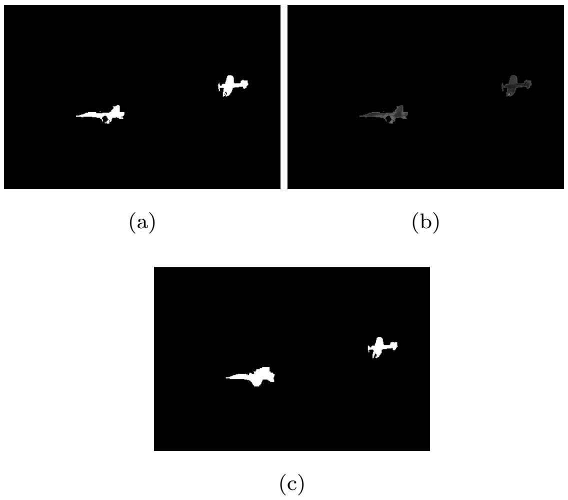
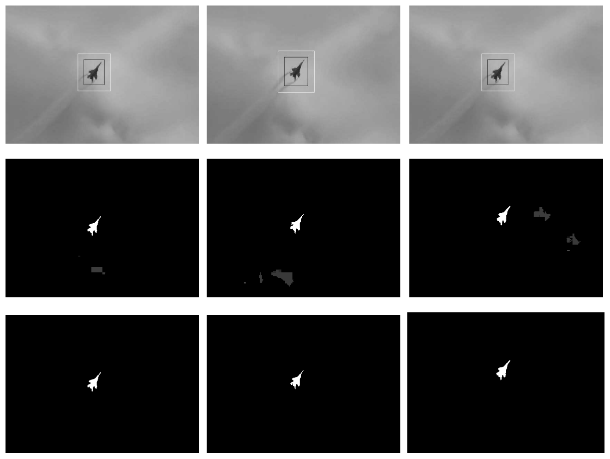


Fig. 6: Foreground-background separation process of Fig. 4. (a) Preliminary segmentation result. (b) Pixel values used for estimating the foreground models. (c) Final result.

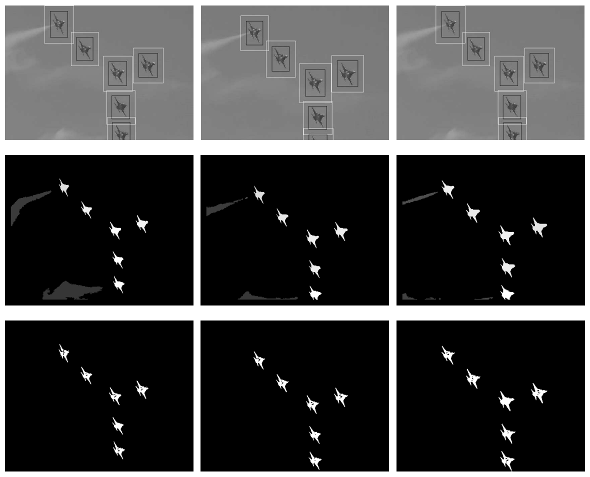
Shape features have been extracted and compared with a variety of methods in the literature, including Hidden Markov Models, Scale Invariant Feature points, tangent/turning functions,[27] curvature maps, shock graphs, Fourier descriptors,[28] polar coordinates,[29] edge based approaches like Chamfer distance methods[30] (based on using small shape fragments), and so on. They all have their benefits and drawbacks, regarding computational complexity, precision capabilities, implementation issues, robustness and scalability. Overall, such methods convert some high level description into a distance based comparison, using some kind of chain code shape representation, incorporating rotation and scale invariance at the high level.

In our solution we intended to use a shape description with a low computational complexity feature extraction step. Thus, we use a simple blob shape extraction, going over the obtained boundary points of the objects and storing them as raw contour information. Also, we use a simple filtering step to drop erroneous/noisy contour pixels (by a median-like filtering of the contour with a small neighborhood window), then using a scale and rotation invariant turning function representation for describing the contours, which will also be the basis for comparison (Sec. 3). For such a fairly simple method to work reliably, we need the outputs of the above presented robust foreground and object extraction step. Fig. 8 shows some examples for this kind of object contour representations.

Texture features can also be extracted by a variety of methods, including LBP,[31] Gabor filters,[32] etc. To have a more complex texture representation with a tested method with known average performance, we decided to use the homogeneous texture descriptor (HTD)[33] from among the MPEG-7 descriptors. The HTD is a well-established and known standard texture descriptor, with known properties, extensive available tests and literature, and it is also lightweight. Ref. 34 provides a detailed analysis and evaluation of the HTD



(a)



(b)

Fig. 7: Samples for robustness against background noise (clouds, vapor trails, etc. For both (a) and (b) 1st row: input frame with background/foreground calculation regions (see Fig. 4); 2nd row: classical foreground masks; 3rd row: presented approach.

against other features, as effective way to describe object segmentation and image and video contents. The descriptor is computed by first filtering the image with a bank of orientation and scale sensitive Gabor filters, and computing the mean and standard deviation of the filtered outputs in the frequency domain. Fig. 9 shows extracted shape examples from the previously obtained foregrounds and the respective regions that will be used for texture feature extraction.

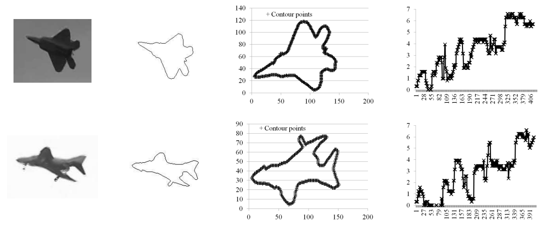


Fig. 8: From left to right: input frame object region; object outline; internal contour representation; turning function representation.

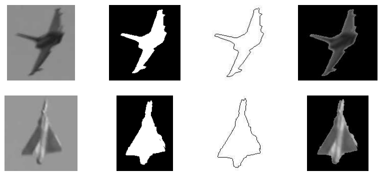


Fig. 9: From left to right: section of the input frames with object; extracted object blob; extracted object contour; region of the original frame with texture.

3 Target recognition

The goal of the recognition is to classify targets into one of the known classes during their observation, in a continuously updated process, regardless of the orientation, view or size of the target. Generic object recognition methods are usually able to classify objects from the same view that they have been trained with. We do not want to have this limitation, thus we build an index of target classes, using various recordings containing targets moving freely, and the classification process is considered as a result produced for a query using this pre-built index. This approach enables us to quickly extend the recognizable classes, and to easily add more samples for a single class into the index, with the goal of being able to recognize the same target class from as many views as possible.

For comparing object shapes, we use the mentioned turning function representation,[35] mostly for speed and efficiency, where the object boundary is represented by a 2D function describing the direction of the tangents of the curve along the objects' contours. To compare two such representations, the minimum distance of all shifted (by t) and rotated (by θ) versions of the turning functions are produced (for rotation and scale invariant comparison), i.e.

$$d_s^2(t_1, t_2) = \underset{t,\theta}{\operatorname{argmin}} \left(\sum_k |t_1(k+t) - t_2(k) + \theta|^2 \right) \ . \tag{16}$$

The turning function comparison we use is based on the rotation- and scale-invariant method from Ref. 35, with the addition of a smoothing step to filter outlier points and sudden irregularities along a contour.

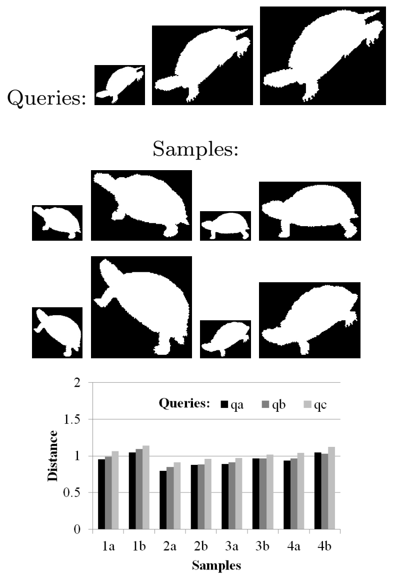


Fig. 10: Visual samples for rotation- and scale-related properties of the used turning function-based calculations. The top row shows the used 3 queries (one query image and its 2 scaled versions, represented by qa, qb, qc in the graph), while the samples are 4 objects with differences in shape, scale and orientation. For each sample, the qa, qb, qc bars are very similar, showing that the scaled and rotated samples are at the same distance from the scaled queries.

Scale- and rotation-invariance of the function is also visualized in Fig. 10, where a query and its 2 scaled versions (top row, noted by qa, qb and qc in the bottom graph) are used as a basis for comparison against 4 other shapes which differ in orientation and in shape as well (differences at the head, leg and tail regions) and a scaled version of these 4 shapes (thus 8 in total). Then, the graph shows the distance of each qa, qb, qc query against the rotated and scaled samples, showing that the bars in group are very similar to each other, meaning that the differences of the scaled queries are very similar against all the rotated and scaled samples.

Texture features are compared by the HTD distance metric (from the MPEG-7 reference), comparing the local means and deviations obtained from the descriptor in the frequency domain. Distance is calculated as the minimum of L_1 norms of shifted texture descriptor vectors (v_1 and v_2) for rotation invariance[33] as

$$d_t(v_1, v_2) = min\left(\sum_k |v_{1,m\cdot f}(k) - v_2(k)|\right) , \tag{17}$$

where $f = 30°$ is the angular division, $m = 1, \ldots, 5$, and $v_{1,m\cdot f}$ is the shifted v_1 vector. Experiments[33] have shown an average accuracy of 77% for this descriptor (tested on the Brodatz texture dataset).

Since the class recognition step is not realtime, queries against the indexed shape database are run at lower frequencies than the input video framerate, and classification is done by building a continuous probability statistics of the results. As Fig. 1 shows, while processing of input frames is continuous (branch A), querying against the index runs in parallel (branch B), with reduced speed, updating the retrieval statistics after each cycle. In the following section dealing with evaluation of the recognition, we include details on how the combined recognition performs against changing query frequencies (i.e. how often queries are run against the index).

For recognition, an index is built from shape and texture features of a dataset of real life videos, mostly from public recordings from air shows. The index structure is based on BK-trees.[36] Such index trees 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.

The class recognition step is part of the main algorithm, but runs as a parallel process, using the outputs of the object extraction and feature extraction steps (branch B in Fig.1). An important part of this process is the class candidate update, in which the recognition results are updated based on the statistics of the most probable results, and the best candidates are continuously refined based on the frequency of the result probabilities:

$$\begin{aligned} P_s(c_i)=P(c_i\in C_s)&=\frac{\alpha\cdot nC_s}{\alpha\cdot nC_s+\beta\cdot nC_t}\ ,\\ P_t(c_i)=P(c_i\in C_t)&=\frac{\beta\cdot nC_t}{\alpha\cdot nC_s+\beta\cdot nC_t}\ , \end{aligned} \tag{18}$$

where P_s and P_t are the probabilities of object c_i belonging to a specific shape (C_s) and texture (C_t) class respectively, α and β are weights that influence how the shape and texture information is included in the retrieval (typically shape has higher weight) and nC_s, nC_t is how many times the object has been classified in to the same specific class during the observation of the object during the full input video (or camera stream), i.e.:

$$nC_s(c_i)=\sum_{k=1}^{i-1}P(c_k\in C_s(c_i))\ , \tag{19}$$

where c_i is the currently evaluated target, and the sum gathers all the instances where this target has belonged to the same C_s class throughout the observation period.

The probability of object c belonging to a specific class C identified by the respective C_s,C_t shape and texture classes will be the class into which it bas been observed to belong the most frequently during the

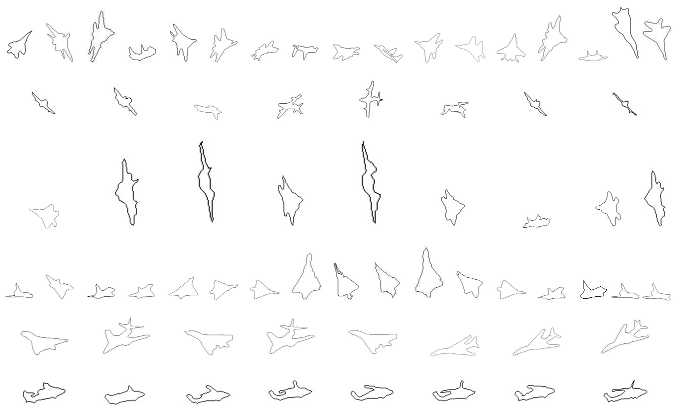


Fig. 11: Example for shapes from dataset classes (each line showing examples from a different class). Each class contains objects with different scale and orientation. Also a good example to show why such shape classes could not be easily learned by traditional classifiers.

Tab. 1: Lengths of query videos used in the test process.

query video	1	2	3	4	5	6	7	8	9	10
video length (frames)	322	134	67	55	100	132	130	143	75	220

observation period:

$$P(c \in C) = \max\{P_s, P_t\} \quad . \tag{20}$$

3.1 Dataset and evaluation

For evaluation we used a dataset gathered from public real life air show videos. The dataset contains 26 classes of planes, and the shapes and textures were extracted automatically with the above presented methods. For each dataset video, objects have been extracted from all frames, which results in classes including a very high variation in scale and orientation of the specific targets (e.g. Fig.11 shows examples from classes to illustrate typical intra-class diversity).

Each class has been manually labeled, and indexes have been built from the extracted shape and texture data. The indexing process for the such obtained approx. 8500 objects takes approximately 2 minutes on a 2.8GHz Intel Core$^{\mathrm{TM}}$i7 CPU, but this is done off-line, and the produced indexes are used during the actual processing (see Fig.1).

To evaluate the retrieval/classification, we used videos not part of the dataset for testing, that contained objects from classes of the dataset. Table 1 show the length (in frames) of the used testing videos.

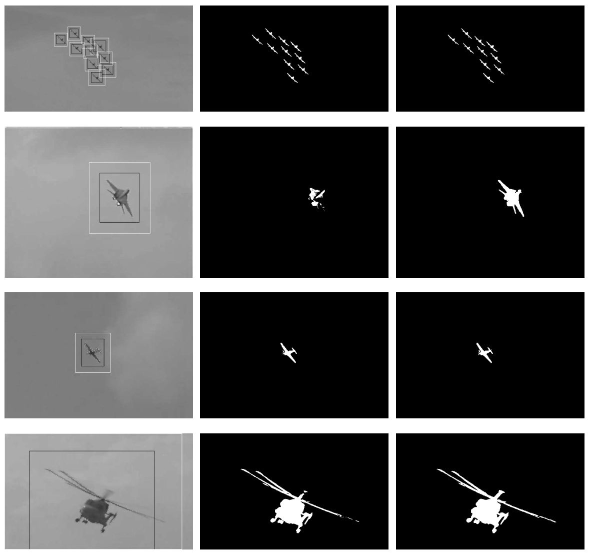


Fig. 12: Example processing outputs for detection of objects. For each example, left to right: input frame with regions for background modeling; raw foreground masks; final filtered object masks.

Regarding the the object extraction discussed in Sec. 2.2 we used the following settings. The number of components of the MoG background and foreground were set to $M_\mathrm{b} = 6$ and $M_\mathrm{f} = 3$, the temporal window had a radius $r = 5$. In the MRF model we used $\beta = 1/2$, and to optimize the energy of the model we used Szeliski et al.'s MRF minimization.[25] Examples of generated object masks are presented in Fig.12, containing samples for multiple objects and challenging contents (clouds, illumination changes, etc.). Sources for the object extraction and some video examples are available online.[37]

For evaluating the recognition rates of the proposed scheme, queries with different frequencies (i.e. every 5^{th}, 10^{th}, 15^{th} and 20^{th} frames) were performed on each of the query videos, with a total of 1118 retrievals, producing the results in Fig. 13 and 14.

The reason for testing the target classification process with varying query frequencies is that since the querying process is not realtime, recognition runs as a parallel process (see Fig. 1), and to reduce the overall computational requirements, it would be desirable to run a target query process less frequently. The goal of the tests is to show that by using shape and texture fused recognition, reducing the query frequency does not hinder the classification process, because errors eventually caused by the less frequent update of the statistics are balanced by the more robust classification using the combined features.

Fig. 13 contains results showing recognition rate data for the 10 query videos, for 5 different query frequencies. Fig. 13a and Fig. 13b show recognition rates for shape only evaluation (only shape based queries were performed) and combined shape+texture evaluation. These graphs show that the combined recognition is less dependent on the query frequency as when only using shape, moreover, the combined feature rate is higher

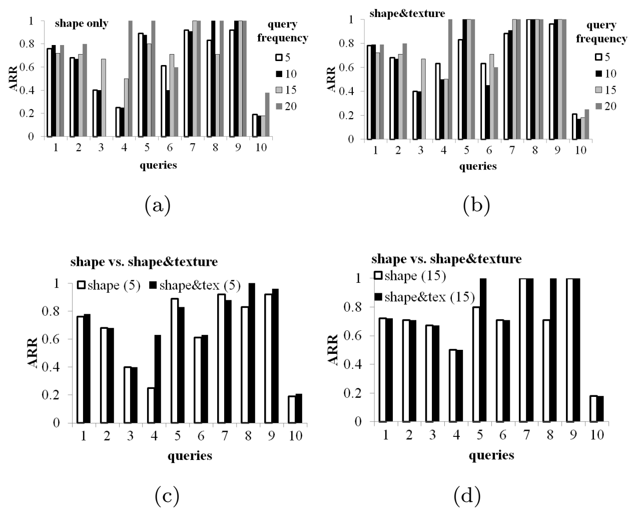


Fig. 13: Evaluation graphs for recognition rates. (a) and (b) show shape and combined shape+texture accumulated recognition rates (ARR) over 10 query videos and 4 different query frequency settings. (c) and (d) show shape vs. combined shape+texture recognition rates in 2 selected frequency settings (every 5 and 15 frames).

for most of the queries. Fig. 13c and Fig. 13d show comparisons of rates between shape only and combined shape+texture queries for two separate query frequencies (every 5 and 15 frames respectively). The graphs show that typically the combined recognition rates are as good, or better than the shape only rates. This is an important feature, since high similarity in texture - which is very typical for planes - should cause a decrease in rates, but the weighted query scheme and the continuously updated class probabilities provide a balanced solution for these problems.

Fig. 14 shows averaged recognition rates, where - for each query frequency - the rate averages were computed over all the 10 query videos, for both the shape only and the combined shape+texture queries, which represents hundreds of averaged queries for each video. This graph shows that, on the average, less frequent querying introduces less errors/noise in the recognition process, but also, that a combined shape+texture recognition scheme is generally better suited for such classification tasks, than relying on shape information alone.

Fig.15 shows accumulated recognition rates (average precision values) for different queries and query frequencies, also showing the first best matches besides the recognized category. These graphs present in a visual form lines of so called confusion matrices, showing how the recognition rates are distributed among the known classes (i.e. which classes and in what percentage are given as results for a query). As described earlier, the recognized class is produced as the best performing category, i.e. the result with the currently highest precision rate (these are the bottom parts of all columns in the figure). The figure's columns also show the other top guesses, which follow the top result. Connected to this figure, Table 2 visually shows the

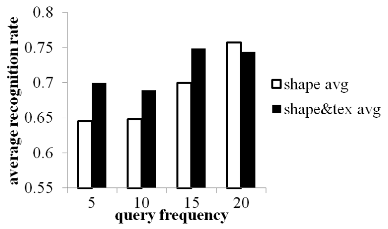
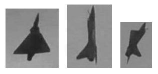
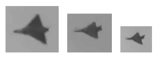


Fig. 14: Averaged recognition rates (over all query videos) for shape only and combined queries, for different query frequencies.

Tab. 2: Samples from query video frames and samples from the best matches at the end of the query video (according to the accumulated recognition rates).

query	input frames	best matches		
q1		✓ 79%	17%	11%
q2		✓ 71%	29%	
q6		✓ 64%	27%	9%
q7		✓ 71%	14%	14%

results for different queries, where frames are sampled from the query and test videos. The “input frames” column shows samples from the query video, while the “best matches” column show the recognized result class and samples from other classes which are close to the result. Here we can see that even the second and third ranked classes are visually close to the correct response (which is the cause for lower precision values).

Connected to the previous figure, Fig. 16 shows the confusion matrix for all classes. Here, 10-15 frames-long queries were used for each class which were not part of the indexed dataset, and recognition rates were recorded for each query, which are shown color coded in the figure. We must note here, that - as described above - recognition rates depend on a number of factors, e.g. length of the query, or number of examples that were included in the dataset index. As a general rule, recognition rates can be improved by observing the target for longer time periods, and if needed, by extending the dataset with the new sample and regenerating the index.

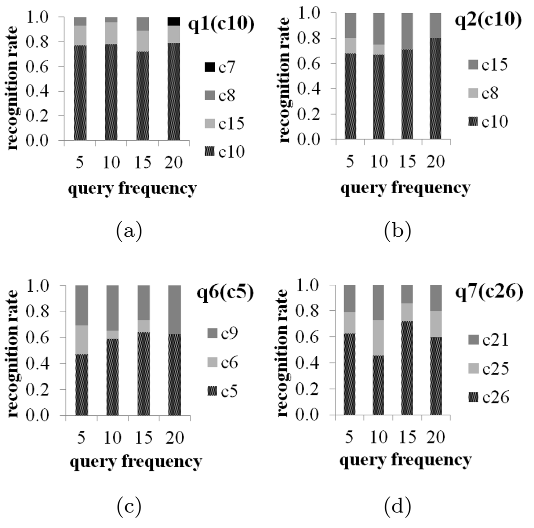


Fig. 15: Evaluation graphs showing accumulated recognition rates for different queries (1, 2, 6 and 7) for different query frequencies. The columns show that for a given query frequency which are the first 3 best matches during the recognition process. For example, a 0.8 recognition rate means that during the entire recognition process, the given specific class was given as the recognition result in 80% of the queries.

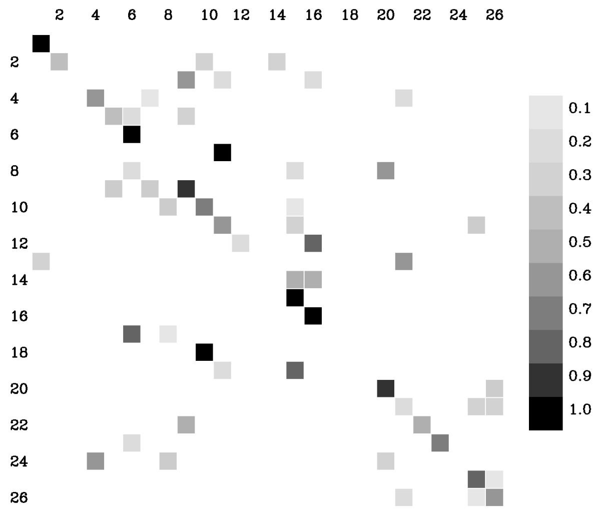


Fig. 16: Confusion matrix of the used classes, showing color coded values of recognition rates, using queries outside of the indexed dataset. Numbers along the axes represent class numbers.

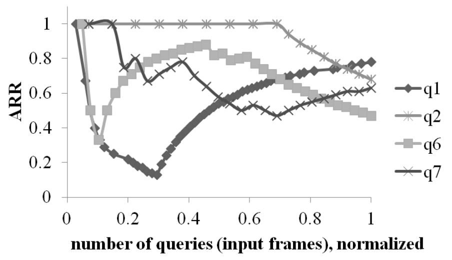


Fig. 17: Accumulated recognition rates vs. the length of the recognition process. The horizontal axis represents the length of the input videos (i.e. the numbers of different queries) on which recognition is being preformed, all normalized to 1 for easier visualization.

Tab. 3: Recognition rates from other works dealing with classification of synthetic or manually segmented shape classes.

Ref. 38 - avg. precisions for:		Ref. 28 - avg. precisions for:		Ref. 29 - avg. precisions for diff. features:	
different noise levels:	0.92	99 shapes	0.9	MI	0.69
30 degree tilt for 9 slant levels:	0.8	216 shapes	0.97	FD	0.57
60 degree tilt for 9 slant levels:	0.75	1045 shapes	0.84	UNL	0.71
90 degree tilt for 9 slant levels:	0.69	24 shape classes	0.9	UNL-F	0.98

Fig.17 (in connection with Fig.15) shows the evolution of recognition rates in relation to the number of queries, i.e. how many times a target is tried to be recognized. This in practice translates to the length of the observation period of a target, during which recognition (query-retrieval step) is performed with a specific frequency (as described earlier). As previous data also shows, the average recognition rates tend to converge around 75% in time (in the figure the length of the observation period is normalized, for better visualization).

Other works in the field of shape recognition typically deal with already available datasets containing clear contours and no noise, or synthetically added noise/distortions. However, as a comparison with other approaches for shape recognition and retrieval, we have included some other results in Table 3. Our average recognition rate of 75% (Fig. 14) is amongst the averages of other approaches. The best advantage of the presented approach is that it produces the presented results on real data, including all steps from background modeling, shape extraction and recognition, and it is easily expandable e.g. by adding a tracking algorithm on top of the produced results.

4 Conclusions

This paper presents a flying target detection method based on corner and edge detection combined with robust background modeling, serving as the basis for a target recognition scheme which uses a fusion of shape and texture features for indexing and classifying the extracted objects. The methods' goals are to be lightweight and robust, providing solutions suitable for application in realtime visual systems for defensive surveillance scenarios as a basis for passive sensors. The detected object contours, main corner and boundary points and object features can be used for target recognition and tracking. Sample sources for object extraction, and video examples are available online.[37] Future work includes adaptation of the methods for ground object detection and tracking for more generic object classes, and adaptation of the presented algorithms for smart cameras in order to provide an integrated solution.

Acknowledgments

The work has been partially supported by grant nr. 83438 of the Hungarian Scientific Research Fund (OTKA).

References

1. J. W. Lu, Y. J. He, and H. Y. Li, "Detecting small target of ship at sea by infrared image," in *Proceedings of IEEE Intl. Conf. on Automation Science and Engineering*, 165–169 (2006).
2. S. Zheng, J. Liu, and J. W. Tian, "An SVM-based small target segmentation and clustering approach," in *Proceedings of Intl. Conference on Machine Learning and Cybernetics*, **6**, 3318–3323 (2004).
3. H. Deng and J. Liu, "Infrared small target detection based on the self-information map," *Infrared Physics & Technology* **54**(2), 100–107 (2011).
4. L. Itti, C. Gold, and C. Koch, "Visual attention and target detection in cluttered natural scenes," *Optical Engineering* **40**(9), 1784–1793 (2001).
5. X. Lia, G. Chena, E. Blaschb, and K. Phamc, "Detecting missile-like flying target from a distance in sequence images," in *Proceedings of Signal Processing, Sensor Fusion, and Target Recognition XVII*, **6968**, SPIE (2008).
6. S. Zhang, W. Liu, and X. Xue, "Research on tracking approach to low-flying weak small target near the sea," in *Proceedings of Optical Information Processing*, **6027**, 962–968, SPIE (2006).
7. C. Bibby and I. Reid, "Visual tracking at sea," in *Proceedings of IEEE Intl Conf on Robotics and Automation*, (2005).
8. T. L. Wenga, Y. Y. Wang, Z. Y. Ho, and Y. N. Sun, "Weather-adaptive flying target detection and tracking from infrared video sequences," *Expert Systems with Applications* **37**(2), 1666–1675 (2010).
9. A. L. Chan, "A robust target tracking algorithm for FLIR imagery," in *Proceedings of Automatic Target Recognition XX, at SPIE Defense, Security, and Sensing*, **7696**, 769603–1–11, SPIE (2010).

10. Y. Wang, W. Yao, Y. Song, N. Sang, and T. Zhang, "Multi-class target recognition based on adaptive feature selection," in *Proceedings of Automatic Target Recognition XX, at SPIE Defense, Security, and Sensing*, **7696**, 769609–1–9, SPIE (2010).

11. E. Blasch and B. Kahler, "Multiresolution EO/IR target tracking and identification," in *Proceedings of Intl. Conf. Information Fusion*, **1**, 275–282 (2005).

12. H. Noor, S. H. Mirza, Y. Sheikh, A. Jain, and M. Shah, "Model generation for video-based object recognition," in *Proceedings of ACM Intl. Conf. on Multimedia*, 715–719 (2006).

13. L. Kovács and T. Szirányi, "Recognition of hidden pattern with background," in *Proceedings of Signal and Data Processing of Small Targets, at SPIE Optics and Photonics: Optical Engineering and Applications*, **6699**, 669906–1–8, SPIE (2007).

14. A. Kovács, L. Utasi, Á. Kovács, and T. Szirányi, "Shape and texture fused recognition of flying targets," in *Proceedings of Signal Processing, Sensor Fusion, and Target Recognition XX, at SPIE Defense, Security and Sensing*, **8050**, 80501E–1–12, SPIE (2011).

15. C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, 147–151 (1988).

16. A. Kovács and T. Szirányi, "Harris function based active contour external force for image segmentation," *Pattern Recognition Letters* **33**(9), 1180–1187 (2012).

17. N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Tr. on System, Man an Cybernetics* **9**(1), 62–66 (1979).

18. J. Canny, "A computational approach to edge detection," *IEEE Tr. on Pattern Analysis and Machine Intelligence* **8**(6), 679–698 (1986).

19. K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Proceedings of The 7th IEEE International Conference on Computer Vision*, **1**, 255–261.

20. C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(8), 747–757 (2000).

21. A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)* **39**(1), 1–38 (1977).

22. Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *In IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(11), 1222–1239 (2001).

23. V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?," *In IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**(2), 147–159 (2004).

24. Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *In IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**(9),

1124–1137 (2004).

25. R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for Markov random fields," in *Proceedings of The 9th European Conference on Computer Vision*, 16–29 (2006). `http://vision.middlebury.edu/MRF/`.

26. L. Kovács and Á. Utasi, "Shape and motion fused multiple flying target recognition and tracking," in *Proceedings of Automatic Target Recognition XX, at SPIE Defense, Security and Sensing*, **7696**, 769605–1–12, SPIE (2010).

27. L. J. Latecki and R. Lakamper, "Application of planar shape comparison to object retrieval in image databases," *Intl. Journal of Information Sciences* **35**(1), 15–29 (2002).

28. W. T. Wong, F. Y. Shih, and J. Liu, "Shape-based image retrieval using support vector machines, Fourier descriptors and self-organizing maps," *Intl. Journal of Information Sciences* **177**(8), 1878–1891 (2007).

29. D. Frejlichowski, "An algorithm for binary contour objects representation and recognition," in *Proceedings of ICIAR, Lecture Notes in Computer Science*, 537–546 (2008).

30. J. Shotton, A. Blake, and R. Cipolla, "Multi-scale categorical object recognition using contour fragments," *IEEE Tr. on Pattern Analysis and Machine Intelligence* **30**(7), 1270–1281 (2008).

31. T. Ojala, M. Pietikäinen, and T. Maënpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Tr. on Pattern Analysis and Machine Intelligence* **24**(7), 971–987 (2002).

32. A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Pattern Recognition* **24**(12), 1167–1186 (1991).

33. B. S. Manjunath, J. R. Ohm, V. V. Vasudevan, and A. Yamada, "Color and texture descriptors," *IEEE Trans. on Circuits and Systems for Video Technology* **11**(6), 703–715 (2001).

34. Y. M. Ro, M. Kim, H. K. Kang, B. S. Manjunath, and J. Kim, "MPEG-7 homogeneous texture descriptor," *ETRI (Electronics and Telecommunications Research Institute) Journal* **23**(2), 41–51 (2001).

35. E. W. Arkin, P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell, "An efficiently computable metric for comparing polygonal shapes," *IEEE Tr. on Pattern Analysis and Machine Intelligence* **13**(3), 209–216 (1991).

36. W. A. Burkhard and R. M. Keller, "Some approaches to best-match file searching," *Communications of the ACM* **16**(4), 230–236 (1973).

37. Á. Utasi, "Object extraction utility." `http://web.eee.sztaki.hu/~ucu/sw/`.

38. M. Bicego and V. Murino, "Investigating Hidden Markov Models' capabilities in 2D shape classification," *IEEE Tr. on Pattern Recognition and Machine Intelligence* **26**(2), 281–286 (2004).

Levente Kovács: Senior research fellow at the Computer and Automation Research Institute of the Hungarian Academy of Sciences, Budapest, Hungary. MSc in IT (2002), PhD in image processing and graphics (2007) from the University of Pannonia, Hungary. Main research areas: image/video feature extraction&indexing, annotation, event detection, non-photorealistic rendering, video restoration.

Andrea Kovács: PhD student and researcher at the Computer and Automation Research Institute of the Hungarian Academy of Sciences, Budapest, Hungary. MSc degree in Electrical Engineering from the Technical University of Budapest (2008). Research interests include image and video processing for remote sensing, shape analysis, boundary extraction, active contours.

Ákos Utasi: Research fellow at the Computer and Automation Research Institute of the Hungarian Academy of Sciences, Budapest, Hungary. MSc in CS (2005), PhD in visual surveillance (2012) from the University of Pannonia, Hungary. His research interests include visual surveillance, motion detection, event detection, and action recognition.

Tamás Szirányi: Ph.D. and D.Sci. degrees in 1991 and 2001, by the Hungarian Academy of Sciences. He was appointed to a Full Professor position in 2001at Pannon University, Veszprém, Hungary, and, in 2004, at the Péter Pázmány Catholic University, Budapest. He leads the Distributed Events Analysis Research Laboratory at the Computer and Automation Research Institute, Budapest. Research areas include machine perception, stochastic optimization, remote sensing, surveillance systems for multiview camera systems, intelligent networked sensor systems, graph based clustering, film restoration. Founder and past president (1997-2002) of the Hungarian Image Processing and Pattern Recognition Society, Associate Editor of IEEE Tr. Image Processing (2003-2009), A.E. of Digital Signal Processing since 2012. Honored with the Master Professor (2001), and ProScientia (2011) awards. Senior member of IEEE and a Fellow of the IAPR and the Hungarian Academy of Engineering. Has more than 200 publications including 40 in major scientific journals.