UDC: 004.932:519.652
DOI: 10.20535/SRIT.2308-8893.2022.1.10

OVERVIEW OF THE DETECTION AND TRACKING METHODS OF THE LAB ANIMALS

M.A. SHVANDT, V.V. MOROZ

Abstract. This article presents an overview of several most common techniques and approaches for object detection and tracking. Today, the tracking task is a very common problem and it can appear in many aspects of our life. One particular case of using object tracking techniques can appear during a lab animal behavior study. Different experimental conditions and the need of certain data collection can require some special tracking techniques. Thus, a set of general approaches to object tracking techniques were considered, and their functionality and possibilities were tested in a real life experiment. In this paper, their basis and main aspects are presented. The experiment has demonstrated the advantages and disadvantages of the studied methods. Considering this, conclusions and recommendations to their usage cases were made.

Keywords: object tracking, object detection, algorithm, video, frame, image, background, foreground, experiment, color space, thresholding, background estimation, segmentation.

INTRODUCTION

Over the last two decades, strong development of tech and technologies had to their widespread implementation in all areas of human life. One of such technologies is image processing and visual analysis [1]. Lots of processes in the world that surrounds us, including street traffic control, the tasks of terrorism prevention and war operations, need to be monitored, analyzed and, very often, controlled. In most such cases photo and video analysis comes in handy. Object detection and tracking usually play important roles in it. A certain object has to be detected on life video stream or recorded video, and then it is necessary to observe and trace the object's movements and position and, presumably, perform some analysis of these movements.

Object detection and tracking are in fact the key tasks of computer vision, as they allow one to gather consecutive information about the object which later can be analyzed [2]. As one knows most of information a person receives through its eyes, that is why computer vision also play an important role in data analysis. The tasks of computer vision include information acquisition, processing of the acquired information, processed data analysis and useful data acquisition. Computer vision is focused on the processing of two- and three-dimensional images. One of the tasks of 2D-processing is optical flow processing (video processing). It includes three key steps:

1) detection of moving objects;

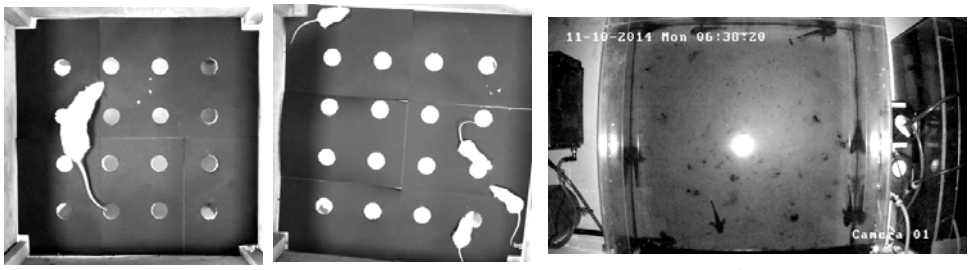
2) object tracking from frame to frame;

3) analysis of an object to determine its characteristics.

In a simple way object tracking can be determined as the task of object trajectory estimation in the image plane.

The detection and tracking methods relay on many features, but the key ones are object shape and video background state. Thus the problem of tracking/detection can be quite challenging due to several factors. For example, there can be a lack of visual information due to the projection of a three-dimensional object on a two-dimensional plane. At the detection phase, the object can be partially occluded or it gets occluded later during tracking process. Also, the tracked object can change its shape or scale, which can also lead to tracking errors. And sometimes its movements can be quite complicated and hard-to-predict. The change of lighting can cause tracking errors, as well as the image noise. Image background can be a major source of difficulties. The easy situation is when there is a static background or it changes very slightly – then it is simple enough to pick out the tracked object. But if the background on each frame changes quite severely, it can also lead to situation when the tracking algorithm fails to pick out the necessary object correctly and loses it. In addition to that the tracking algorithm must be applicable for real-time video processing. In order to solve all these problems, different approaches have been suggested.

In biology there often is a necessity to study the life processes and behavior of lab animals, for example mice or fish. Such studies in this field have been carried out for a long time and they are still of sufficient scientific interest [3–5]. Different lab conditions may require specific approaches for automatic animal behavior examination. Thus the problem of object detection and tracking can be studied well on the particular example (Fig. 1) of such activity study. In the first case (Fig. 1, *a* and 1, *b*) the test environment is represented by a box with circle holes in the bottom (the holes denote the center of the test stand). In this case the task is to track the lab mice, note their movements between holes, time spent in the test stand center and moments when several mice contact with each other. In the second case there is an aquarium (Fig. 1, *c*). The task is quite similar: to track fish movements and notice their contacts. In both cases the camera is placed above the test environment. In order to solve these particular tasks a wide research was carried out to find the most suitable object detection and tracking approaches that could be used separately or combined. Thus various methods were examined, their advantages, disadvantages and algorithmic aspects have been considered. The complete analysis is presented further in this paper.



a *b* *c*

Fig. 1. Lab animals behavior study: *a* — lab rats; *b* — lab mice; *c* — fish

THE PROBLEM OF OBJECT DETECTION

The key task that appears during the object tracking process on video stream is their detection. Some methods require full object detection only on the first frame, some use continuous full detection on each frame.

Segmentation-based detection. *Image segmentation* is a process of digital image division on multiple sets of pixels. This process can also assign special markers to each pixel, so that the pixels with similar markers could have common visual characteristics [6]. For example, such approach can be based on the Watershed Transform [7] and results in the Watershed Algorithm combined with the Distance Transform. Image segmentation allows one to simplify the image analysis. It results in the highlighting of the borders and object itself. Thus, pixels belonging to the same segments are similar by some calculated feature (color, brightness value, etc.), with the rest of elements being significantly different by that feature. The result of such segmentation can be seen on Fig. 2. This approach is easy to use when objects of interest significantly differ from the background by some parameter. But the main problem is that it has low versatility and requires too accurate algorithm parameters setup in each particular case. It also is very sensitive to lighting conditions.

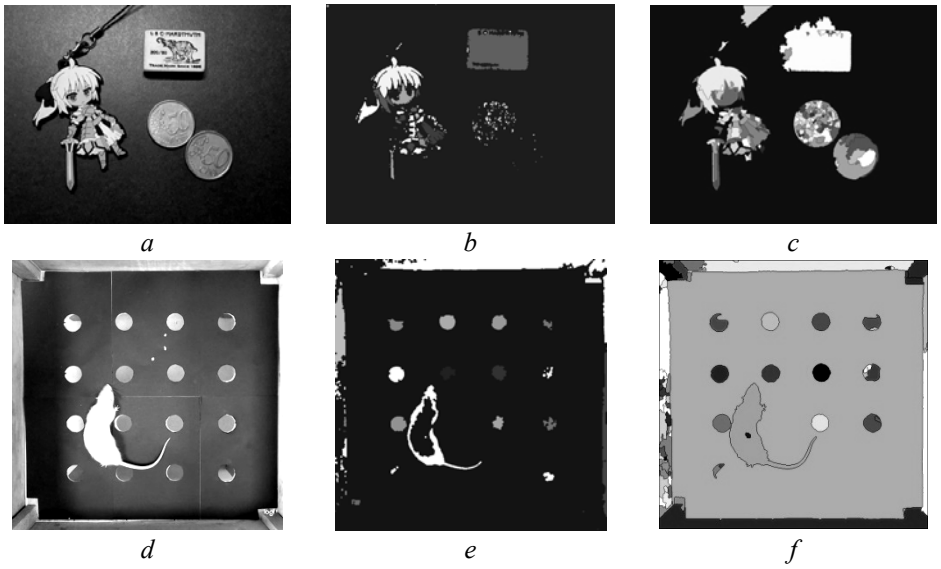


Fig. 2. Image segmentation: *a* — test 1: original image; *b* — test 1: markers; *c* — test 1: segmentation result; *d* — test 2: original image; *e* — test 2: markers; *f* — test 2: segmentation result

Another approach that works with image segments is *Template Matching* [8]. The algorithm compares the given object template with the sub-regions of processed image. To do this it simply slides the template along the image and checks if it matches to some region. The template (or patch) is sliding one pixel at a time (left to right, up to down) [9, 10]. At each location the algorithm calculates a metric that allows one to understand how similar the patch is to that particular area of the source image. For each location T over input image I it stores the metric in the result matrix R. Each cell (x, y) from R contains the match metric. Thus it is possible to find the best match by searching for the highest value (or lower, depending on the type of matching method) in the R matrix.

It is worth noticing, that while the patch must be a rectangle it may be that not the whole area of the rectangle is relevant. In this case the algorithm uses mask to isolate the portion of the patch that should be used to find the match. The mask is a grayscale image that masks the template image and must have the same

dimensions and number of channels. The match $R(x, y)$ can be calculated in several ways:

1. As a Square Difference:

$$R_{sq}(x, y) = \sum_{x', y'} (T(x', y') - I(x + x', y + y'))^2. \quad (1)$$

2. As Normed Square Difference:

$$R_{nsq}(x, y) = R_{sq}(x, y) / \sqrt{\sum_{x', y'} T(x', y')^2 \sum_{x', y'} I(x + x', y + y')^2}. \quad (2)$$

3. As Cross Correlation:

$$R_{ccorr}(x, y) = \sum_{x', y'} (T(x', y') I(x + x', y + y')). \quad (3)$$

4. As Normed Cross Correlation:

$$R_{nccorr}(x, y) = R_{ccorr}(x, y) / \sqrt{\sum_{x', y'} T(x', y')^2 \sum_{x', y'} I(x + x', y + y')^2}. \quad (4)$$

5. As Correlation Coefficient:

$$R_{ccoeff}(x, y) = \sum_{x', y'} (T'(x', y') I'(x + x', y + y')), \quad (5)$$

where

$$T'(x', y') = T(x', y') - \frac{\sum_{x'', y''} T(x'', y'')}{wh},$$

$$I'(x + x', y + y') = I(x + x', y + y') - \frac{\sum_{x'', y''} I(x + x'', y + y'')}{wh}.$$

6. As Normed Correlation Coefficient:

$$R_{nccoeff}(x, y) = R_{ccoeff}(x, y) / \sqrt{\sum_{x', y'} T(x', y')^2 \sum_{x', y'} I(x + x', y + y')^2}. \quad (6)$$

The result of Template matching algorithm is presented on Fig. 3. This approach can be used in case of some scene analysis when camera is static and objects of interest look almost identical, for example, detection of some products on a factory assembly line. But on the other hand, such method does not work stable in case of rotation or scaling and when an object is partially occluded. If the searched objects are scaled the most simple way could be to enlarge the template image as much as possible and that consequently scale it down at each search stage, hoping that at some point the template image will be scaled to the correct size. If the objects are rotated, the easiest way is to create a set of rotated by 1 degree template images and then iteratively check each sample. But both such approaches will deliver poor performance, especially in case of high resolution images. In case when the objects are both scaled and rotated, the performance can get even worse.

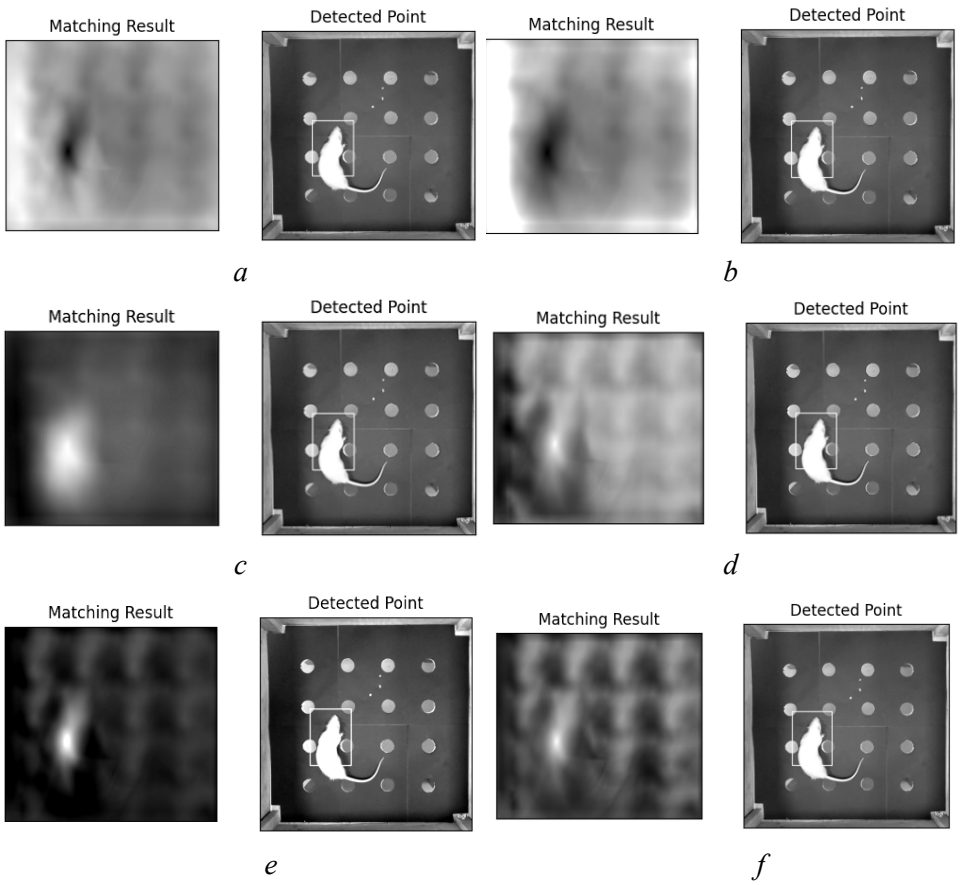


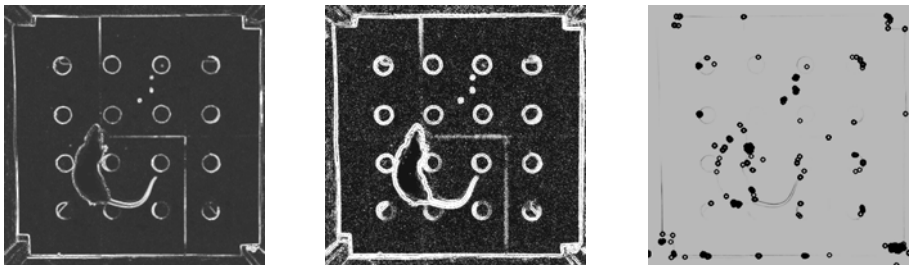
Fig. 3. Template matching: *a* — Square Difference (1); *b* — Normed Square Difference (2); *c* — Cross Correlation (3); *d* — Normed Cross Correlation (4); *e* — Correlation Coefficient (5); *f* — Normed Correlation Coefficient (6)

Feature-based detection. One more way to detect an object on image is to find it by some features. A feature is some element or part that is more distinguished than the other parts/elements, some local image particle. As simple example of such features are corners and borders. The search of an object in this case is based on the comparison of the characteristic features of the processed frame and a template showing the object one is looking for [11]. Local features should be repetitive (stable to change the angle or lighting during the video series), compact (their number should be much less than the total number pixels of the image), unique (each feature must have their own description).

To identify the characteristic features special detectors are used. One of the most common is the Harris (corner) detector (Fig. 4,*c*), which recognizes the features of the type "corner" in the image. As corner detectors are not very sensitive to image scaling, the concept of so-called drops (Blob) was introduced - teardrop-shaped neighborhoods with a special point located in the center. One of the most common blob methods is *LoG* (The Laplacian of Gaussian) [12]. LoG is a filter $LoG(x,y) = \frac{x^2+y^2-2\sigma^2}{\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$ that applies the Gaussian operator

$L(x, y; \sigma) = G(x, y; \sigma) * I(x, y)$, $G(x, y; \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}$, and the Laplace operator $\nabla^2 = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$ to the image, respectively (Fig. 4,*a*):

Here σ the standard deviation, $L(x, y; \sigma)$ is the Gaussian scale-space representation of an image $I(x, y)$, and $*$ is the convolution operator. *DoG* detector $DoG \widehat{=} (G_{\sigma_1} - G_{\sigma_2}) = \frac{1}{\sqrt{2\pi}} \left(\frac{1}{\sigma_1} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{\sigma_2} e^{-\frac{x^2+y^2}{2\sigma_2^2}} \right)$ based on the Gaussian difference $L_1(x, y; \sigma_1) = G_{\sigma_1}(x, y; \sigma_1) I(x, y)$, $L_2(x, y; \sigma_2) = G_{\sigma_2}(x, y; \sigma_2) I(x, y)$; $L_1(x, y; \sigma_1) - L_2(x, y; \sigma_2) = G_{\sigma_1}(x, y; \sigma_1) I(x, y) - G_{\sigma_2}(x, y; \sigma_2) I(x, y) =$ $= (G_{\sigma_1}(x, y; \sigma_1) - G_{\sigma_2}(x, y; \sigma_2)) * I(x, y) = DoG * I(x, y)$. [13] is also common (Fig. 4,*b*). The difference between the two smoothing is as follows.



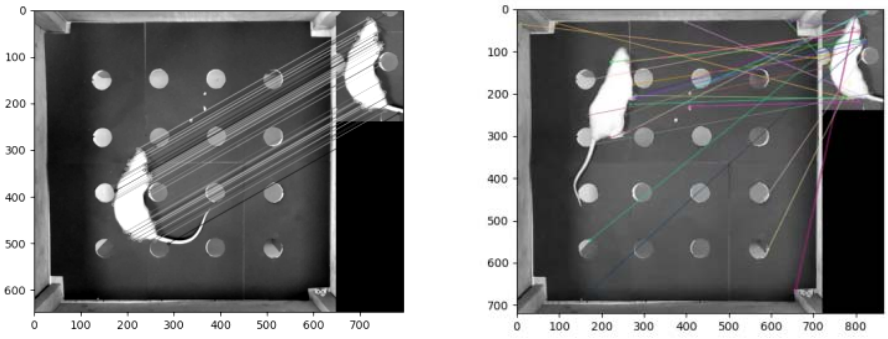
a *b* *c*

Fig. 4. Blob detection using Gaussian filter: *a* — Laplacian of Gaussian (LoG); *b* — Gaussian difference (DoG); *c* — Harris detector

After finding special points, it is necessary to compare them. This task requires a way of compact characteristic features representation. In practical tasks, the *SIFT* (Scale-Invariant feature transform) descriptor [14] and its derivatives, such as *SURF* [15], are considered to be the best methods. Despite being invariant to small turns, scaling of objects and changes in stage lighting, the feature-based approach actually makes it impossible to define an object as instance of some class and it also provide false results in case of object dynamic shape change (Fig. 5).

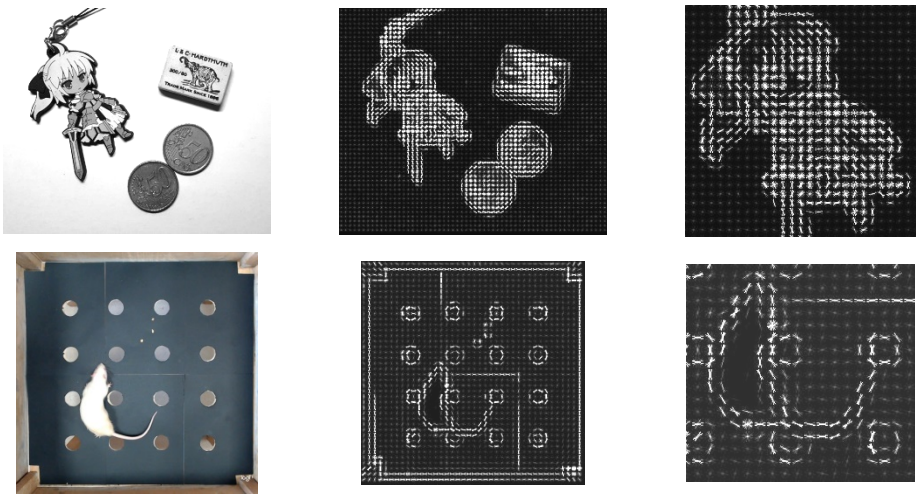
Categorical recognition. Methods for detecting characteristic features are well suited to solve the problem of searching across the database of images [16]. However, in our particular case it is necessary not simply to reveal some object on the frames of a video corresponding to some template, but also to recognize all objects of certain class. The considered problem could be solved by methods of feature detection but at the same time it would be necessary to create a large number of templates and it would take a long time to compare the frames with each of them. The approach that allows us to avoid this is based on the classification of objects, i.e. categorical recognition. It consists of two main elements: the definition of a set of features or descriptors and machine learning of

the classifier. As a set of features the Histogram of Oriented Gradients (*HOG*) or Haar features can be used. The HOG features [17] are based on the calculation of the number of gradient directions in the local areas of the image (Fig. 6).



a *b*

Fig. 5. SIFT feature object detection: *a* — perfect match; *b* — mismatch case



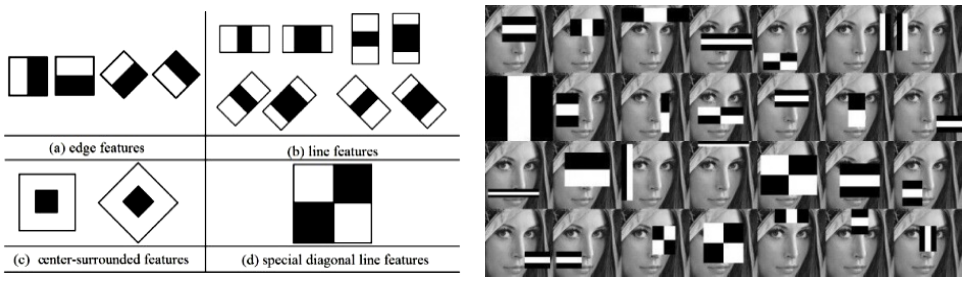
a *b* *c*

Fig. 6. HOG features detection: *a* — input images; *b* — extracted HOG features; *c* — HOG features (magn.)

Haar signs [18] or primitives are rectangles consisting of adjacent areas (see Fig. 7, *a*). These areas get positioned on the image, then the intensity of pixels in the areas is summed, and then the difference between the obtained sums is calculated, which is the value of a certain feature of a certain size, located on the image in a certain way. An example of the use of Haar features is shown on Fig. 7, *b*. The advantage of Haar features is a relatively high computational speed. Machine learning is used to create a class clarifier. The classifier is used to indicate which features belong to the object. Thus for training purpose some base of these features is used.

HOG is calculated on a dense grid of evenly distributed cells (Fig. 6). This method highlights well the objects with multiple details, but in case when the object is mostly a single piece without any significant details, in most cases it will

only highlight the borders, which can be not enough for complete detection. The Haar approach on the other hand is suitable for face detection and recognition. But this approach still requires a pre-trained classifier which sometimes can be problematic and will not work well with objects that tend to change shapes.



a *b*

Fig. 7. Haar features: *a* — Haar features types (source: www.spiedigitallibrary.org); *b* — general representation of training the Haar classifier (source: medium.com)

THE PROBLEM OF OBJECT TRACKING

As mentioned above the process of tracking of moving objects is one of the components of many real-time systems such as observation systems, video analysis and others. The input data of any tracking algorithm is a sequence of images (video frames) $I_1, I_2, ..., I_n$ with an increasing amount of information that needs to be processed and analyzed. The task of tracking is to construct the trajectories of the target objects on the input sequence of frames. If we assume that the position of the object on the image numbered k is denoted by P_k. Then the trajectory of the object is sequence of its positions $P_s, P_{s+1}, ..., P_{s+l-1}$, where s is the number of the first frame in which the object was detected, l is the number of frames in the sequence where the object is observed.

Some methods of object detection allow us to detect the entire object, but usually they are not suitable for continuous work, especially for real-time video processing. In most cases performing the detection "from scratch" for each frame can be very costly in terms of performance and speed, thus the detection process should be optimized in some way, especially if some frames have already been processed and we received some additional information from them. That is way several different object tracking approaches have been introduced. Note that depending on the method of tracking the position of the object can be determined differently (coordinates and size of the sides of the surrounding rectangle, coordinates of the center of mass of the contour, etc.).

Color-based tracking approaches. The idea of simple color-based tracking consists of the following steps [17]: first, the algorithm takes each frame and converts it from RGB to HSV color model. It is necessary because the RGB representation is nor very suitable for selection of some specific color range. It can be performed in the following way [1, 19]: the given R, G, B values are scaled to change the range from $0\ldots255$ to $0\ldots1$. Then one calculates

$$C_{\max} = \max(R', G', B'), \quad C_{\min} = \min(R', G', B'), \quad \Delta = C_{\max} - C_{\min},$$

where R', G', B' are scaled R, G, B values. The Hue (H) can be calculated using the following formula:

$$H = \begin{cases} 0^o & \text{if} \quad \Delta = 0; \\ 60^{\circ} \cdot \left(\dfrac{G' - B'}{\Delta} \right), & \text{if} \quad C_{\max} = R'; \\ 60^{\circ} \cdot \left(\dfrac{B' - R'}{\Delta} + 2 \right), & \text{if} \quad C_{\max} = G'; \\ 60^{\circ} \cdot \left(\dfrac{R' - G'}{\Delta} + 4 \right), & \text{if} \quad C_{\max} = B'; \end{cases}$$

$$H = H + 360, \ \text{if} \ H < 0.$$

The Saturation (S) calculation:

$$S = \begin{cases} 0, & C_{\max} = 0; \\ \dfrac{\Delta}{C_{\max}}, & C_{\max} \neq 0. \end{cases}$$

The Value V calculation:

$$V = C_{\max}.$$

It is worth noticing that before the RGB to HSV conversion a Gaussian blur is applied, as described in [20] to remove noise in order to receive better output. The result is seen on Fig. 8, *a*, *b*. The second step after a successful conversion is the color thresholding. The lower and upper boundaries of the desired color are set in the HSV color space. This allows to filter out the rest of the colors from the image. The thresholding process for an input image I can be described as follows:

$$dst(I) = (lowerB(I)_1 \leq$$

$$\leq src(I)_1 \leq upperB(I)_1) \wedge ... \wedge (lowerB(I)_n \leq src(I)_n \leq upperB(I)_n),$$

where $lowerB(I)_i \leq src(I)_i \leq upperB(I)_i$ stands for the i_{th} input array channel, $i = 1,...,n$. Thus:

- For every element of a single-channel input array:

$$dst(I) = lowerB(I)_1 \leq src(I)_1 \leq upperB(I)_1;$$

- For two-channel arrays:

$$dst(I) = lowerB(I)_1 \leq src(I)_1 \leq upperB(I)_1 \wedge lowerB(I)_2 \leq src(I)_2 \leq upperB(I)_2.$$

The resulting image is a binary image, i.e. all its pixel values are 1 or 0. For the resulting image after thresholding the operations of *erosion* and *dilatation* are applied, as described in [21, 22]. This allows us to get rid of most separated areas that managed to pass the threshold (Fig. 8, *c*, *d*, *f*). The final step is a centroid calculation for each blob using the binary image moments [23]:

$$M_{i,j} = \sum_x \sum_y x^i y^j I(x, y). \tag{7}$$

If one denotes a blob area as M_{00}, then the centroid can be calculated as follows:

$$\{\bar{x};\bar{y}\} = \left\{\frac{M_{10}}{M_{00}};\frac{M_{01}}{M_{00}}\right\}. \quad (8)$$

For each blob its point $\{\bar{x};\bar{y}\}$ can be used as object position on current frame (Fig. 8, *e*).

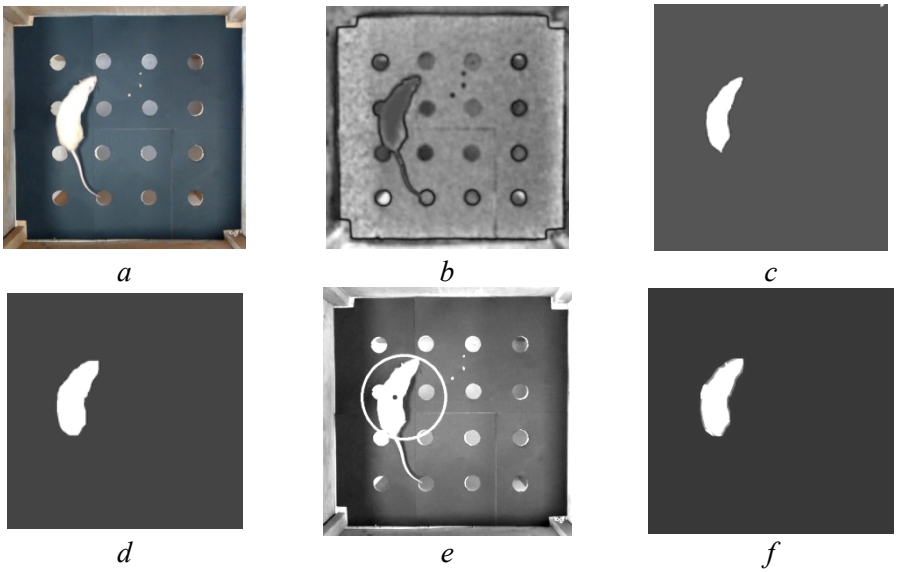


Fig. 8. Color-based tracking: *a* — RGB frame; *b* — HSV frame; *c* — raw binary image; *d* — after erosion & dilatation operations; *e* — detected object; *f* — HSV thresholding result

The advantages of such tracking approach are that in fact the target object gets detected automatically it works well with objects, which change their shape. In addition the overall realization is very simple and it has good performance speed, which makes it suitable for real-time video capturing. But this method has some serious disadvantages. Firstly, it is more of a detection than tracking technics, so if there are several object of interest and they occasionally get occluded, after repeated detection there is no guarantee, that these objects' positions were not messed up (Fig. 9, *a*, *b*). This point requires additional control in addition to tracking technics. Also it requires from the user a manual selection of lower and upper HSV color threshold boundaries, which is not a very easy task by itself, and the tracked objects have to be distinguished from the background by color (Fig. 9, *c*). In addition, this technic will work well mostly only with the colorful images, because the grayscale color space is much more poor for color differentiation, thus it will not be suitable for usage on videos like one on Fig. 1, *c*. An finally, if several tracked objects of one occasionally come very close to each other, they merge into just one object and the algorithm begins treating them as a single object. This fact devalues the accuracy of the method.

a *b* *c*

Fig. 9. Color-based tracking errors: *a* — 4 mice; *b* — 2 mice are merged into a single blob; *c* — identical color range case

Another approach is the background subtraction. The idea of this method is similar to the color tracking algorithm as it also directly separates the tracking object from the background [21, 24]. The main difference is that one requires an image of observed location without any moving objects on it. In case of rat/mice tracking task this location is the test box. This method was implemented and tested in [20]. In brief, it consists of the following steps:

1. The algorithm receives an image of empty observed location, it is converted from RGB to grayscale and cleared from noise with Gaussian or Median filters (Fig. 10, *a*).

2. Each video frame is also converted from RGB to grayscale and cleared from noise (Fig. 10, *b*).

3. For each frame the background subtraction operation is performed on both grayscale frame and grayscale image of an empty box. The main formula is:

$$d_{i,j} = \left| b_{i,j} - f_{i,j} \right|, \quad i = \overline{1,n}, \quad j = \overline{1,m},$$

where $d_{i,j}$ is pixel value of the resulting image (i.e. background subtraction output / image difference), $b_{i,j}$ and $f_{i,j}$ are the pixel values of empty grayscale background image and each grayscale video frame respectively, $i = \overline{1,n}$ and $j = \overline{1,m}$ are the dimensions image/frames. Notice, that these dimensions must be equal for both empty background image and frame for obvious reasons. The result is presented on Fig. 10, *c*.

4. Next step is thresholding [25]: all pixel values, that are higher than some *threshold* are put to 0, the rest is set to 255. The result is a binary image (Fig. 10,*d*), i.e. it is only black and white. If necessary, operations of erosion and dilatation are applied (Fig. 10, *e*).

5. Finally, similarly to color-based tracking, for binary image blobs centroid calculation is performed [23]. It is done using image moments calculation (formulas 7 and 8). The result can be seen on Fig. 10, *f* (circles were detected using Hough transform as the box central area [25]).

This methods has similar to color-based tracking advantages, as it also requires a threshold value, but it is more convenient as it requires only one such value instead of a range. Thus it is more stable. But the main disadvantage is the mandatory existence of the background image. In case of difficulties with

providing such image this algorithm should not be used. The rest of possible problems are also similar to color-based tracking.

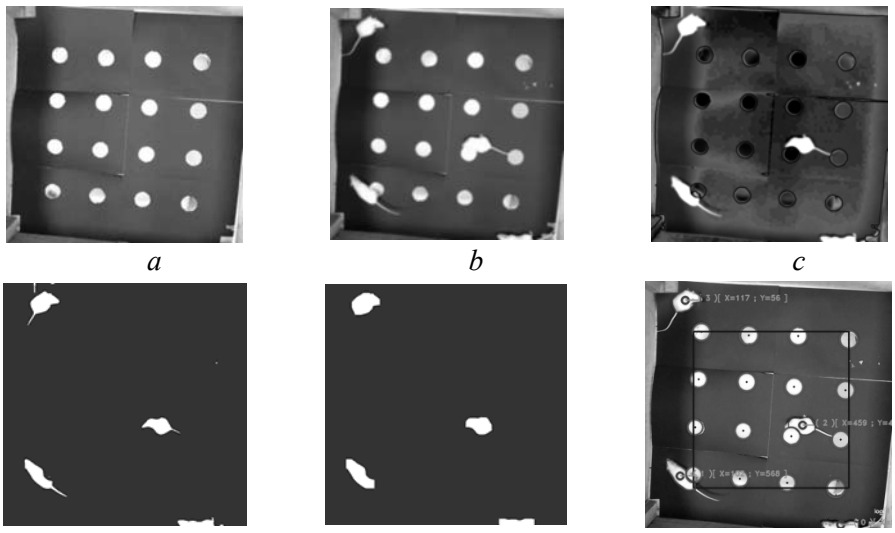


Fig. 10. Background subtraction tracking: *a* — grayscale box image smoothed; *b* — grayscale frame smoothed; *c* — image difference; *d* — binary image; *e* — after erosion/dilatation; *f* — background subtraction result

It can happen that there is no background image provided. Then *background estimation* method comes in handy. It can be used as an addition to background subtraction algorithm. The main idea is that the background image gets calculated from input video. It can be performed using the Approximation Median Algorithm [26, 27]. As it is described in [26], it finds the difference of values of the current pixel's intensity and the median of some recent pixel's intensity. For this task an *n*-size buffer is used, it contains *n* last frames whose pixel values are used for calculating the median value for background image. The main formula for this method can be written as follows:

$$\left|F_{i,j,k} - Med_{i,j,k}\right| > thresh, \quad i = \overline{1, side_1}, \quad j = \overline{1, side_2}, \quad k = \overline{1, frNum},$$

with F representing the current frame and Med being the median of last n frames. For each new frame k for each pixel (i, j) the difference of current frame pixel with pixel of median of last n frames decides whether this value is foreground or background. The median value gets updated for last n recent pixel values.

The described method can be also performed in the following way [28]. Assuming, that the camera is static and most of the time every pixel shows the same piece of the background, every moving object will occlude the background. In this case for the video on can randomly sample n frames. Thus for every pixel, now there are n estimates of the background. As long as a pixel is not occluded by the moving object, more than 50% of the time, the median of the pixel over these n can be a good estimate of the background at that pixel. This process can be repeated this for every pixel and thus it recovers the entire background.

As an alternative, Mixture of Gaussian can be used for background estimation [26, 27]. It uses a Gaussian probability density function to evaluate the pixel

intensity value. This method calculates the difference of values of the current pixel's intensity and cumulative average of the previous values. It means that the algorithm keeps a cumulative average μ_t of the recent pixel values, and if the difference of the current image's pixel and the cumulative pixel values is greater than the product of a constant value c and standard deviation σ, then this difference it is classified as foreground. Thus for each frame t the F_t pixel value can be denoted as foreground pixel, if the following inequality holds:

$$\left|F_t - \mu_t\right| > c\sigma_t, \quad t = \overline{1, fr\,Num},$$

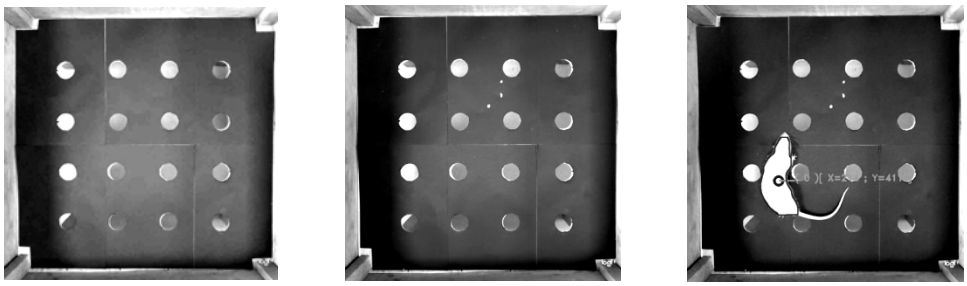
In other case, this value can be classified as background. Also, this algorithm updates background image as the running average using formulas:

$$\mu_{t+1} = \mu_t F_t + (1-\alpha)\mu_t,$$

$$\sigma^2_{t+1} = \alpha(F_t - \mu_t)^2 + (1-\alpha)\sigma^2 t,$$

where α is the learning rate (typically $\alpha = 0{,}05$); F_t is the pixel current value; μ_t is the previous average.

The result of such technics is shown on Fig. 11, 12. In case of mice/rat tracking, there is an empty box image, so it can be compared with the resulting background estimation (Fig. 11). Notice, that three white dots were static on each video frame, thus they managed to pass to estimated background (Fig. 11, *b*). Fig. 12, *a* shows estimated background image, acquired from the corresponding aquarium video. In this particular case no empty aquarium image had been provided, thus this is exactly the case when background estimation can be applied. Also notice, that one fish in the top corners remained static during the whole video, so they were also classified as background and was later missed by the algorithm. This case shows the main drawback of such approach (Fig. 12, *b*).

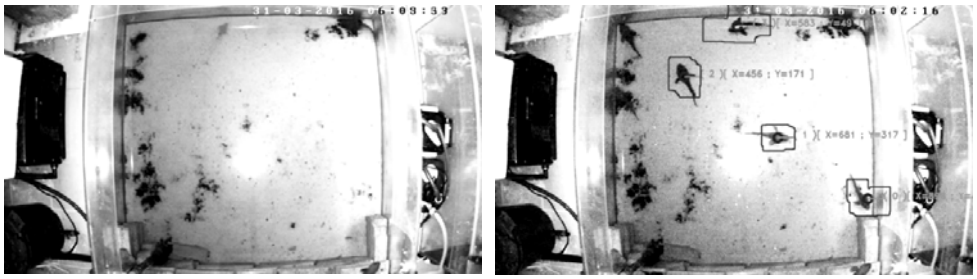


a *b* *c*

Fig. 11. Background estimation: *a* — original empty box image; *b* — estimated box image; *c* — tracking result

`

The positive side of this approach is that when one uses the background subtraction and there is no empty background image, in most cases this technic can compensate this need. But as it is shown on Fig. 12, *a*, if one of the tracked objects remains static during the whole video, it will be classified as a part of the background and thus the tracking method will not be able to detect and track it. It is also effective only if the entire background is static.



a *b*

Fig. 12. Background subtraction operation test with fish: *a* — empty aquarium estimation; *b* — tracking result

Kernel tracking & optical flow. A kernel component is the shape of an object. In the simplest case, the component can be represented by a rectangular or oval shape, in more complex ones by three-dimensional model of the object projected on the plane of the image. The methods of this group are usually used if the motion is determined by a normal displacement, rotation, or affine transformation. Component tracking is an iterative localization procedure based on maximizing some similarity criterion. In practice, it is realized using mean shift and its continuous modification (Continuous Adaptive Mean Shift, CAM Shift).

The idea of the *Mean Shift* [29, 30] is that for each special point (in the general case, for each object) the search window is selected, the center of masses of the intensity distribution (i.e. of the histogram) is calculated. Accordingly, the center of the window is shifted to the center of mass, which is the position of the point on the current frame. Determining the position of the point in the following frames is reduced to the application of the next step of the method of "average shift". The method stops when the center of mass stops shifting (Fig. 13).

Fig. 13. Mean shift tracking

The problem with Mean Shift is that the window (ROI) always has the same size whether the object is very far or very close to the camera, it needs to be adapted during the tracking process. The solution to this is CAM Shift (Continuously Adaptive Mean Shift) [31]. This approach applies the Mean Shift first, then once Mean Shift converges, it updates the size of the window as

$$s = 2\sqrt{\frac{M_{00}}{256}}.$$

CAM Shift also calculates the orientation of the best fitting ellipse to it. It applies the Mean Shift with new scaled search window again and previous window location. This process continues until the required accuracy is met (Fig. 14). This approach shows fine work speed and is more stable than Mean Shift, but the

main problem with CAM Shift is that it is connected to color range, thus it is sensitive to lighting conditions and can fail with objects that change their shape.

Fig. 14. CAM Shift tracking

Optical flow estimation can be used as the alternative to all previous methods [32, 33]. Optical flow itself can be described as a trace of visible object movement between two consecutive frames [34]. It can be caused by moving object itself or by camera movement and it is represented by 2D vector field where each vector is a displacement vector showing the movement of points from first frame to second. There are several applications, where optical flow can be used, especially motion detection, or video stabilization.

There are several assumptions that optical flow works with [33, 34]: firstly, the pixel intensities of an object do not change between consecutive frames, and secondly, pixels in neighborhood must have similar motion. Let $I(x,y,t)$ be a pixel from the first frame (t is time), and it gets moved by distance (dx,dy) in the next frame taken after dt time. Assuming, that the pixel intensity does not change, the following holds:

$$I\left(x,y,t\right)=I\left(x+dx,y+dy,t+dt\right).$$

By using the Taylor series approximation of right-hand side, removing common terms and dividing by dt one gets the following equation:

$$f_x u+f_y v+f_t=0$$

$$\text{with } f_x=\frac{\partial f}{\partial x};\ f_y=\frac{\partial f}{\partial y};\ u=\frac{dx}{dt};\ v=\frac{dy}{dt}. \tag{9}$$

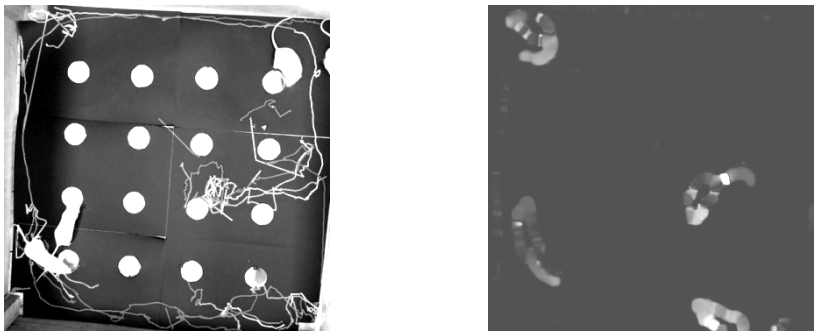
The equation (9) is called *Optical Flow* equation, where f_x, f_y are image gradients, which can be found, and f_t is the gradient along time. The u and v components are unknown and thus equation (9) cannot be solved with two unknown variables. There are several solutions to this problem. One of them is Lucas-Kanade method [32, 34]. The Lucas–Kanade approach uses the 3×3 patch around the point, so that 9 points have the same motion. It is possible to calculate f_x, f_y, f_t for these 9 points, thus there appears a task to solve 9 equations with two unknown variables which is over-determined. It can be solved with least square fit method:

$$\begin{bmatrix}u\\v\end{bmatrix}=\begin{bmatrix}\sum\limits_i f_{x_i}^2 & \sum\limits_i f_{x_i}f_{y_i}\\ \sum\limits_i f_{x_i}f_{y_i} & \sum\limits_i f_{y_i}^2\end{bmatrix}^{-1}\begin{bmatrix}-\sum\limits_i f_{x_i}f_{t_i}\\ -\sum\limits_i f_{y_i}f_{t_i}\end{bmatrix}.$$

It is worth noticing, that the inverse matrix is similar to Harris corner detector, as corners are better points to be tracked. Also, as it can be seen, this approach allows to detect only small motions, but no the big ones. In order to solve this problem the pyramids are used: when going up in the pyramid, small motions

are removed and large motions become small motions. Thus when one applies Lucas–Kanade there, one gets optical flow along with the scale. The result is presented on Fig. 15, *a*. As Lucas–Kanade method computes optical flow for a sparse feature set (*sparse optical flow*), using, for example, Shi-Tomasi corner detection technic, another approach, based on the Gunner Farneback's algorithm (*dense optical flow*) [34, 35] computes the optical flow for all the points in the frame (Fig. 15, *b*). For vectors (u,v) it is possible to find their magnitude and direction. Thus it allows us to trace the moving object and its movement directions (color shows the direction).

Both Lucas–Kanade and Farneback's algorithms perform well in case of a static background. They also do not require any manual object selection, the object of interest can be found by its motion. But in case of object occlusion redetection is required, this fact makes these technics suitable mostly only for laboratory conditions, like in this particular case (Fig. 15). They also perform fine in case of object's shape change.



a *b*

Fig. 15. Optical flow: *a* — Lucas–Kanade method; *b* — Farneback's method

Point tracking methods. In such approaches, it is assumed that the position of the object is determined by the location of a set of characteristic points. The same object in consecutive frames is represented by sets of corresponding pairs of points. This group of methods is divided into two subgroups:

- Deterministic methods [36] use qualitative heuristics of motion (a small change in velocity, the invariance of the distance in three-dimensional space between a pair of points belonging to object), in essence, the task is reduced to minimizing the function correspondence of sets of points. Methods based on the calculation of dense and sparse optical flux, as well as methods of matching key point descriptors are typical representatives of deterministic methods.
- Probabilistic methods use an approach based on the concept of state space. It is believed that a moving object has a certain internal state, which is measured on to each frame. To estimate the next state of the object, it is necessary to generalize as much as possible the received measurements, that is, to determine the new state provided that the set is obtained measurements for states on previous frames. Typical examples of such methods are methods based on the Kalman filter [37, 38] or Particle filter [39].

The Kalman filter is used to track single objects in noisy images. Each state of the system can be described by a vector of its parameters. By some influence the system passes from one state to another. The set of all states of the system and transitions form a model. There is a concept of observation data vector. This is

a set of system parameters that we can extract from the observation of behavior of the system. In most cases, the dimension of the vector states of the system exceeds the dimension of the observation data vector. In this case, the Kalman filter is able to estimate with a certain probability the complete internal state of the system.

The Kalman filter works with time-discrete linear dynamical systems. Such systems are modeled by Markov chains with the help of linear operators and terms with normal distribution. At each discrete moment of time, the linear operator acts on the state and translates it into another state, adding some random variable in the form of normal noise and, in the general case, a control vector that simulates the influence of the control signal. Mathematical model of this process in matrix form:

$$x_k = F_k x_{k-1} + B_k u_k + w_k,$$

$$z_k = H_k x_k + v_k,$$

where F_k is a $n \times n$ matrix that describes how the state changes systems in transition from $k-1$ to k without control; B_k is a $n \times l$ matrix that describes how the control effect u_k changes state from $k-1$ to k, l is the dimension of the control effect; H_k is a $c \times l$ matrix that describes how the state x_k is transformed into an observation z_k, c is the dimension of the observation vector; w_k, v_k are arbitrary values representing the normally distributed noise when measuring the state c by the corresponding covariance matrices $Q_k, R_k, w_k \sim N(0, Q_k)$, $v_k \sim N(0, R_k)$.

The algorithm consists of two repeating phases: extrapolation phase and correction phase. During the operation of the first phase, a prediction of values of the state variables takes place (extrapolation) based on state estimation on the previous step, as well as their uncertainty. This assessment often also called *a priori* because it is given to perform any measurements and is based on mathematical model only. The second phase is responsible for refining the result of extrapolation using the appropriate measurements, possibly obtained with some error. This assessment is called *a posteriori*.

In the classical operation of the algorithm, these phases alternate, i.e. the prediction happens in relation to the results of adjustment with past iteration, and the adjustment specifies the result of the extrapolation phase. However, in some cases, the correction phase may be missed and the prediction will be based on an unspecified estimate. This situation can occur if for some reason we do not have information from the measuring sensors at this stage. To understand further processes, it is necessary to enter the following notation:

x_k — the actual state of the system at the time k;

$\hat{x}_k$ — estimated state at time k;

$\hat{x}_k^-$ — predicted system state at time k;

P_k — estimated matrix of error covariance of condition measurement;

P_k^- — predicted matrix of error covariance of condition measurement.

T a b l e 1. Kalman Filter algorithm

Extrapolation		Correction
1) State extrapolation: $\hat{x}_k^- = F_k \hat{x}_{k-1}^- + B_k u_k$	$\rightarrow$	1) Kalman amplification: $K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$
2) Covariance matrix extrapolation: $P_k^- = F_k P_{k-1} F_k^T + Q_k$	$\leftarrow$	2) State vector correction: $\hat{x}_k = \hat{x}_k^- + K_k (z_k - H_k \hat{x}_k^-)$
		3) Covariance matrix calculation: $P_k = (I - K_k H_k) P_k^-$

Interactive tracking. The general idea of the set of suggested approaches is the motion and appearance models [40]. As one remembers, the task of tracking is to detect an object in the current frame given this object was successfully detected and tracked in all (nearly all) previous frames. If the object was tracked up until current frame, it means that it has been moving, i.e. the parameters of the motion model are known. This term means that object's location and the velocity (speed and motion direction) in previous frames are also known. If there is no other information on the object, it can be possible to estimate its new location based on the currently existing motion model and thus one can get close to the object real position.

Thus if the object is simple and its appearance did not change too much, it is possible to use some simple template as an appearance model and look for it. But as the object appearance can change pretty much, the model can be represented as a classifier that is trained during the whole tracking process. The main task for the classifier is to classify a rectangular region of interest (ROI) of an image as either an object or background. In order to do this, it takes as the input an image patch and returns an estimation value in range [0, 1]. This value is the probability that the image patch contains the object. As one can see here the binary classification is used, thus if the estimation score is 0, it means that the classifier thinks that the image patch is the background, and if the score is 1, it says that the patch is the object. The training (learning) is performed during the tracking process, as the classifier "learns" to detect the object. This approach is similar to the work of the neural networks, but is this particular case the training set is quite small, as it is just the set of video frames.

There are several interactive training methods [40–44], that uses this methodology. First group includes BOOSTING, MIL, KCF trackers. The BOOSTING tracker is based on the AdaBoost algorithm and uses HAAR cascade based face detector. The user should provide the initial bounding box, that is used as a positive example for the object and many other image patches outside this box are treated as the background. Also, this algorithm cannot detect the tracking failure. MIL (Multiple Instance Learning) is based on the same idea, but instead of considering only the current location of the object as a positive example, it looks in a small neighborhood around the current location to estimate several potential positive examples. The KCF (Kernelized Correlation Filters) tracker also supports the ideas from BOOSTING and MIL. The difference is, that this tracker uses the fact that the multiple positive samples used in the MIL tracker have large overlapping regions. The fact of overlapping is used for performance enhancement. This method reports a tracking failure and can recover from partial occlusion.

The experiment had shown that both BOOSTING (Fig. 16) and MIL (Fig. 17, *a*) tracker had shown similar performance and tracking quality. The main problem was that in current condition they began failing and losing the objects (Fig. 17, *b*). The KCF tracker indeed had shown much faster frame processing due to its technic of usage of the overlapping regions. But it also resulted in much worse tracker quality – the tracker tends to loose objects very quickly.

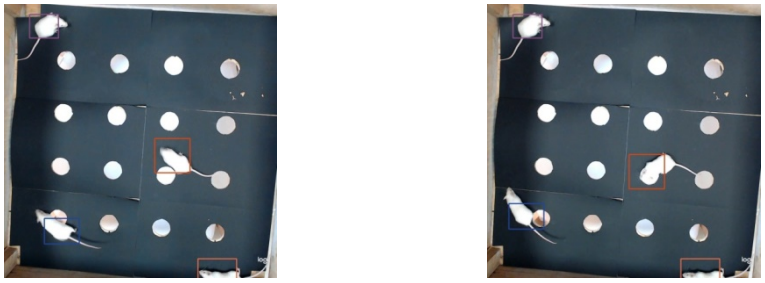
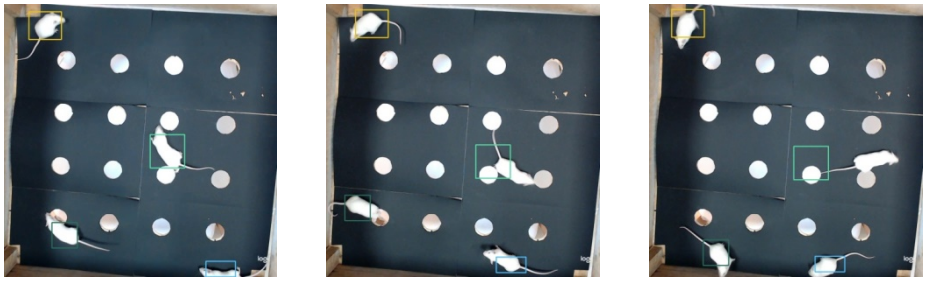


Fig. 16. BOOSTING tracker



a *b*

Fig. 17. MIL tracker: *a* — successful tracking; *b* — tracker failure

The other set of trackers include TLD, MEDIANFLOW, MOSSE and CSRT. The TLD (Tracking, learning, and detection) as its name suggests separates the tracking process into three subtasks, i.e. tracking, learning, detection [40]. According to its creators, the algorithm tracks the object from frame to frame, while its detector localizes all object's appearances that have been found so far and performs tracker's self-correction if required. During the learning process the algorithm estimates errors of the tracker's object detector and then updates it in order to avoid them further on. This results in tracker jumping around, which one hand, in case of sudden occlusions allows the tracker to return back to initial object. But on the other hand as result of such jumps quite often TLD tends to lose its target and focus on another object. Thus despite this tracker performs fine under occlusion over multiple frames or scale changes, it provides lots of false positives results, which making it almost unusable. The testing of TLD is shown on Fig. 18.

The MEDIANFLOW tracker [40] follows the object in both forward and backward directions and estimates the divergence between object's two trajectories. Thus it calculates forward-backward error and tries to minimize it. This technic allows to detect tracking failures and keep a more or less stable trajectory. The test has shown (and it matches the earlier results [40]), that this

tracker works well only with predictable and small movements with no occlusions. But in case of lab animals which tend to move unpredictably it fails almost immediately (Fig. 19).

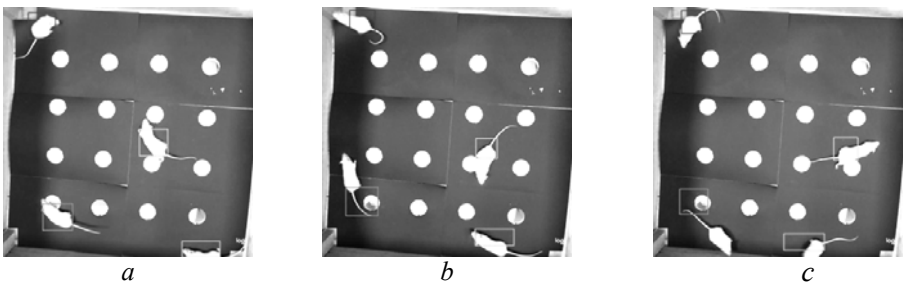
a *b*

Fig. 18. TLD tracker: *a* — successful tracking; *b* — tracker jumps to other object

a *b*

Fig. 19. MEDIANFLOW tracker: *a* — successful tracking; *b* — tracker failure due to chaotic mice movements

The final two trackers are MOSSE (Minimum Output Sum of Squared Error) and CSRT (Channel and Spatial Reliability Tracker). The MOSSE tracker is based on the calculation of adaptive correlation, as it produces stable correlation filters when initialized using a single frame. This tracker can operate fast at very high framerates, and it couples fine with lighting, scale, pose changes and non-rigid deformations. But its overall performance is lower than learning-based trackers, for example, like MIL or KCF. The CSRT tracker uses the spatial reliability map for adjusting the filter support to the part of the selected region from the frame for tracking [1]. This allows to resolve situations with enlarging and localization of the selected, thus it can track fine the non-rectangular regions or objects. But in current static background conditions and unpredictable movements it also tends to loose objects (Fig. 20).



a *b* *c*

Fig. 20. CSRT tracker: *a* — successful tracking; *b* — tracker begins to lose objects; *c* — tracker failure: object lost

CONCLUSIONS

As the experiment results show, the interactive trackers may perform well under certain conditions, but in this particular case of lab animal tracking none of them can track all objects up to the very end of the video. The main advantage of them is that they usually do not require any additional data, like empty background image and do not need to estimate it (as minor preprocessing, some noise reduction or contrast change can be applied). Also, some of them can resolve minor occlusion situations. But all of these methods require manual object selection and none of them managed to demonstrate any stable work, which can say that they are not completely suitable for the current task in its original form.

The more simple approaches, like background subtraction (with or without background estimation) in case of static location can perform quite well, as they detect the objects automatically and their computational complexity is not high. Kernel tracking approaches, as well as optical flow and point tracking can provide some additional information about object motion, which can be used in combination with the simple/interactive tracking methods for performance improvement and additional motion data acquisition.

Some of object detection methods can also serve as some addition to the tracking methods during the tracking process itself. Different use cases can require different detection technics. For example, in case of lab mice behavior observations with specific environment conditions (a box with the dark floor as a test stand) an automatic detection by color can be applied, but in case of some more complex environment a combination of several detection approaches may be required. The detailed comparison of tested object detection and tracking approaches is presented on Table 2, 3.

The testing results also allow one to assume that for lab mice activity study most likely background subtraction in combination with image segmentation and interactive tracker can be used. Fish tracking may require some interactive tracker combined with optical flow and image segmentation. Our further research will include the usage of the composition of interactive trackers and the simple approaches. The idea is to use the positive sides of both sets of methods to compensate each other's disadvantages. Also, the neural network based tracking is planned to be applied in attempt to create a completely stable tracker.

Table 2. Object detection approaches performance comparison

Method	Advantages	Disadvantages / Features
Image segmentation	Easy to use when the objects of interest significantly differ from the background by some parameter Can be used as an addition to other methods to highlight the main image parts	Low versatility Requires too accurate algorithm parameters setup in each particular case Sensitive to lighting conditions
Template Matching	Useful for scene analysis when camera is static and objects of interest look almost identical Can be used for detection of some products on a factory assembly line	Can fail in case of rotation, scaling or partial occlusions If rotation/scaling take place, additional search steps will be required
Feature-based detection	Invariant to minor turns, scaling of objects and changes in stage lighting Suitable for rough object search	Requires a way of compact characteristic features representation Impossible to define an object as instance of some class Provides false results in case of object dynamic shape change

Continued Table 2

Method	Advantages	Disadvantages / Features
Categorical recognition	HOG highlights well the objects with multiple details Haar approach is more suitable for face detection and recognition	When the object is mostly a single piece without any significant details, HOG will only highlight the borders Both approaches still require a pretrained classifier Fails when objects tend to change shape

Table 3. Object tracking approaches performance comparison

Method	Advantages	Disadvantages / Features
Color-based tracking	Object gets detected automatically Simple realization Good performance speed Works well with objects, which change their shape	Cannot handle occlusions Cannot handle object 'merging' Requires additional algorithms for centroid tracking Requires manual color threshold setup Unstable if colors are too similar
Background subtraction	Same benefits as color tracking Requires only one threshold value instead of a range	Mandatory existence of the background image Other possible problems are similar to those ones from color-based tracking
Background estimation	Useful if no empty background i mage provided Can be used as addition to Background subtraction	Objects of interest that do not move actively can be classified as background Effective only if the entire background is static
Kernel tracking (Mean Shift/CAM Shift)	Shows fine work speed CAM Shift ROI can adjust its size during the process	The Mean Shift ROI has fixed size CAM Shift is connected to color range CAM Shift is sensitive to lighting conditions and object shape changes
Optical flow	Good work in case of static background The object of interest can be found by its motion automatically Performs fine in case of object's shape change	In case of object occlusion redetection is required Object gets lost when its movements are getting slower Cannot handle 'object merging' problem
Point tracking (Kalman filter)	Good performance when tracking single objects on noisy images	Has complicated computations and implementation Not good at handling object merging/ occlusions
Interactive tracking (BOOSTING / MIL / KCF)	Fine work speed (BOOSTING/ MIL) Best work speed (KCF) Suitable for non-static background (moving camera)	KCF tends to loose objects more often than MIL/BOOSTING Not good at handling object merging/occlusions
Interactive tracking (TLD)	Can handle object occlusions/ merging Works with non-static background (moving camera)	Provide too many false positives, tends to loose object of interest
Interactive tracking (MEDIANFLOW / MOSSE / CSRT)	CSRT has fine work speed at high framerates CSRT handles lighting, scale, pose changes Suitable for non-static background (moving camera)	MEDIANFLOW works well only with predictable and small movements with no occlusions Methods works well mostly with predictable object movements Methods cannot handle occlusions

ACKNOWLEDGEMENTS

Special thanks for the research assistance and provided test videos and images of lab animals to Faculty of Biology of Odesa I.I. Mechnikov National University.

REFERENCES

1. A.R. Smith, “Color Gamut Transform Pairs”, in *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pp. 12–19, 1978. doi:10.1145/800248.807361.
2. W.K. Pratt, *Digital Image Processing*; 4th edition. Wiley-Interscience, A John Wiley & Sons, Inc., Publication, 2007, 807 p.
3. X. Zhao, S. Yan, and Q. Gao, “An Algorithm for Tracking Multiple Fish Based on Biological Water Quality Monitoring”, *IEEE Access*, vol. 7, pp. 15018–15026, January 2019. doi:10.1109/ACCESS.2019.2895072.
4. J. Delcourt, M. Denoel, M. Ylieff, and P. Poncin, “Video multitracking of fish behaviour: A synthesis and future perspectives”, *Fish and Fisheries*, vol. 14, no. 2, pp. 186–204, June 2013. doi:10.1111/j.1467-2979.2012.00462.x.
5. H.E.-D. Mohamed et al. “MSR-YOLO: Method to Enhance Fish Detection and Tracking”, *The 11th International Conference on Ambient Systems, Networks and Technologies (ANT), April 6–9, 2020, Warsaw, Poland*.
6. P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, “Contour Detectionand Hierarchical Image Segmentation”, *IEEE TPAMI*, vol. 33, no. 5, pp. 898–916, 2011. doi: 10.1109/TPAMI.2010.161.
7. S. Beucher, “The Watershed Transformation Applied to Image Segmentation”, *Scanning microscopy*, vol. 6, July 2000, 26 p.
8. R. Brunelli, *Template Matching Techniques in Computer Vision. Theory and Practice*. Wiley, 2009, 339 p.
9. Dr. A.S. Khedher, Dr. A.M. Alkababji, and O. Hadi, “Improving the Reliability of Object Recognition Based On Template Matching”, *AL Rafdain Engineering Journal, Computer Science*, pp. 81–88, 2015.
10. *Template Matching*. Available: https://docs.opencv.org/3.4/de/da9tutorial_template_matching.html
11. D.G. Lowe, “Object recognition from local scale-invariant features”, *International Conference on Computer Vision – ICCV, 1999*, vol. 2, pp. 1150–1157.
12. H. Kong, H.C. Akakin, and S.E. Sarma, “A Generalized Laplacian of Gaussian Filter for Blob Detection and Its Applications”, *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1719–1733, January 2013. doi: 10.1109/TSMCB.2012.2228639.
13. L. Assirati, N.R. Silva, L. Berton, A.A. Lopes, and O.M. Bruno, “Performing edge detection by Difference of Gaussians using q-Gaussian kernels”, *2nd International Conference on Mathematical Modeling in Physical Sciences 2013, Journal of Physics*, *Conference Series 490(2014) 012020*, IOP Publishing, 2014, 4 p. doi: 10.1088/1742-6596/490/1/012020.
14. T. Lindeberg, “Scale Invariant Feature Transform”, *Scholarpedia*, vol. 7, no. 5: 10491, 2012, 17 p. doi: 10.4249/scholarpedia.10491.
15. H. Bay, T. Tuytelaars, and L.V. Gool, “SURF: Speeded up robust features”, in *Proceedings of the 9th European conference on Computer Vision*, vol. 1, 2006, 14 p. doi: 10.1007/11744023_32.
16. N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection”, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR ’05), San Diego, United States*, vol. 1, pp. 886–893, June 2005. doi: 0.1109/CVPR.2005.177.
17. T. Kobayashi, A. Hidaka, and T. Kurita, “Selection of Histograms of Oriented Gradients Features for Pedestrian Detection”, *Neural Information Processing, 14th International Conference, ICONIP 2007, Kitakyushu, Japan*, *November 13–16, 2007*, Revised Selected Papers, Part II, pp. 598–607. doi: 10.1007/978-3-540-69162-4_62.
18. P. Viola and M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features”, *IEEE Conference on Computer Vision Pattern Recognition*, vol. 1, 2001. doi: I-511.10.1109/CVPR.2001.990517.
19. H. Hunud, A. Kadouf, and Y.M. Mustafah, “Colour-based Object Detection and Tracking for Autonomous Quadrotor UAV”, *2013 IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 53, 2013, 9 p. doi:10.1088/1757-899X/53/1/012086.

20. V.V. Moroz and M.A. Shvandt, "Study of movement and behavior of laboratory animals by methods of object detection and tracking", *Herald of the National Technical University "KhPI", Series of "Informatics and Modeling",* Kharkiv: NTU "KhPI", Kharkiv, vol. 13, no. 1338, pp. 93–103, 2019. doi: 10.20998/2411-0558.2019.13.09.
21. A.M. Raid, W.M. Khedr, M.A. El-dosuky, and Mona Aoud, "Image restoration based on morphological operations", *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, vol. 4, no. 3, 2014, pp. 9–21. doi: 10.5121/ijcseit.2014.4302.
22. S. Ravi and A.M. Khan, "Morphological Operations for Image Processing: Understanding and its Applications", in *NCVSComs-13 Conference Proceedings*, 2015, pp. 17–19.
23. Y. Zhang, "Pathological Brain Detection based on wavelet entropy and Hu moment invariants", *Bio-Medical Materials and Engineering*, no. 26, pp. 1283–1290, 2015.
24. P. Joshi, D.M. Escrivá, and V. Godoy, *OpenCV By Example*. Birmingham: Packt Publishing Ltd, 2016, 297 p.
25. M. Nixon and A. Aguado, *Feature Extraction & Image Processing for Computer Vision*; 3d ed. London: Elsevier Ltd, 2012, 623 p.
26. S.-C.S. Cheung and C. Kamath, "Robust Background Subtraction with Foreground Validation for Urban Traffic Video", *EURASIP Journal on Applied Signal Processing*, vol. 14, pp. 2330–2340, Hindawi Publishing Corporation, 2005.
27. M.A. Alawi, O.O. Khalifa, and M.D.R. Islam, "Performance Comparison of Background Estimation Algorithms for Detecting Moving Vehicle", *World Applied Sciences Journal 21 (Mathematical Applications in Engineering)*, IDOSI Publications, 2013, pp. 109–114. doi: 10.5829/idosi.wasj.2013.21.mae.99934.
28. S. Mallick, *Simple Background Estimation in Videos using OpenCV (C++/Python)*. 2019. Available: https://learnopencv.com/simple-background-estimation-in-videos-using-opencv-c-python/.
29. Y. Cheng, "Mean Shift, Mode Seeking, and Clustering", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, August 1995.
30. D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, May 2002. doi: 10.1109/34.1000236.
31. G. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface", *Archived 2012-04-17 at the Wayback Machine, Intel Technology Journal*, no. Q2, 1998.
32. B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision", in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pp. 674–679.
33. A. Radgui, C. Demonceaux, E. Mouaddib, D. Aboutajdine, and M. Rziza, "An adapted Lucas–Kanade's method for optical flow estimation in catadioptric images", *The 8th Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras - OMNIVIS, Marseille: France, Marseille, 2008*, 12 p.
34. *Optical Flow*. Available: https://docs.opencv.org/3.4/d4/dee/tutorial_optical_ flow.html.
35. G. Farneback, "Two-Frame Motion Estimation Based on Polynomial Expansion", *Lecture Notes in Computer Science*, 8 p., 2003.
36. C. Veenman, M. Reinders, and E. Backer, "Resolving motion correspondence for densely moving points", *IEEE* Transactions on *Pattern Analysis Machine Intelligence*, vol. 23, no. 1, pp. 54–72, 2001. doi: 10.1109/34.899946.
37. A. Salarpour, A. Salarpour, M. Fathi, and M.H. Dezfoulian, "Vehicle tracking using Kalman filter and features", *Signal and Image Processing: An International Journal (SIPIJ)*, vol. 2, no. 2, pp. 45–67, 2011. doi: 10.5121/sipij.2011.2201.
38. S. Dan, Zh. Baojun, and T. Linbo, "A Tracking Algorithm Based on SIFT and Kalman Filter", in *Proceedings The 2nd International Conference on Computer Application and System Modeling*, 2012, pp. 1563–1566.
39. F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.J. Nordlund, "Particle Filters for Positioning, Navigation and Tracking", *IEEE Transactions on Signal Processing*, vol. 2, no. 2, pp. 425–437, 2002.

40. S. Mallick, *Object Tracking using OpenCV (C++/Python)*. 2017. Available: https://learnopencv.com/object-tracking-using-opencv-cpp-python/.
41. Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-Backward Error: Automatic Detection of Tracking Failures", in *Proceedings of International Conference on Pattern Recognition, 23-26 August, 2010, Istambul, Turkey*, 4 p. doi: 10.1109/ICPR.2010.675.
42. Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection", in *Proceedings of IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 1, 2010, 14 p.
43. S. Zhou, Y. Peng, K. Gong, and L. Shu, "An Improved TLD Tracking Algorithm for Fast-moving Object", *International Conference on Computer Science, Electronics and Communication Engineering (CSECE 2018), Advances in Computer Science Research*, vol. 80, pp. 69–73.
44. D.S. Bolme, J.R. Beveridge, B.A. Draper, and Y.M. Lui, "Visual Object Tracking using Adaptive Correlation Filters", *CVPR*, 2010, 10 p. doi: 10.1109/CVPR.2010.5539960.

Received 07.07.2021

INFORMATION ON THE ARTICLE

Maksym A. Shvandt, ORCID: 0000-0002-4580-3961, Odesa I.I. Mechnikov National University, Ukraine, e-mail: maxim.shvandt@gmail.com

Volodymyr V. Moroz, ORCID: 0000-0002-3240-4590, Odesa I.I. Mechnikov National University, Ukraine, e-mail: v.moroz@onu.edu.ua

ОГЛЯД МЕТОДІВ ВИЯВЛЕННЯ ТА ВІДСТЕЖЕННЯ ЛАБОРАТОРНИХ ТВАРИН / М.А. Швандт, В.В. Мороз

Анотація. Подано огляд та аналіз кількох найпоширеніших методів та алгоритмів виявлення і відстеження об'єктів. Окремий випадок використання техніки відстеження об'єктів може виникнути під час лабораторного дослідження поведінки тварин. Різні експериментальні умови та необхідність збирання певних корисних даних можуть потребувати спеціальних методів відстеження. Тому розглянуто набір загальних підходів до відстеження об'єктів, а їх функціональність та можливості перевірено в реальному експерименті. Наведено їх основу та базові аспекти. Експеримент продемонстрував переваги та недоліки досліджуваних методів. Зроблено висновки та рекомендації щодо випадків їх використання.

Ключові слова: відстеження (трекінг) об'єктів, детектування об'єктів, алгоритм, відео, кадр, зображення, задній план, передній план, експеримент, кольоровий простір, порогове значення, обчислення заднього плану, сегментація.

ОБЗОР МЕТОДОВ ОБНАРУЖЕНИЯ И ОТСЛЕЖИВАНИЯ ЛАБОРАТОРНЫХ ЖИВОТНЫХ / М.А. Швандт, В.В. Мороз

Аннотация. Представлены обзор и анализ нескольких распространенных методов и алгоритмов обнаружения и отслеживания объектов. Частный случай использования методики отслеживания объектов может возникнуть во время лабораторного исследования поведения животных. Различные экспериментальные условия и необходимость сбора определенных полезных данных могут потребовать специальных методов отслеживания. Поэтому рассмотрен набор общих подходов к отслеживанию объектов, а их функциональность и возможности проверены в ходе реального эксперимента. Представлены их основа и базовые аспекты. Эксперимент продемонстрировал преимущества и недостатки исследуемых методов. Сделаны выводы и рекомендации по поводу случаев их использования.

Ключевые слова: отслеживание (трекинг) объектов, обнаружение объектов, алгоритм, видео, кадр, изображение, задний план, передний план, эксперимент, цветовое пространство, пороговое значение, вычисление заднего плана, сегментация.