http://dx.doi.org/10.5755/j01.eie.26.1.25311 ELEKTRONIKA IR ELEKTROTECHNIKA, ISSN 1392-1215, VOL. 26, NO. 1, 2020

A New Object Tracking Framework for Interest Point Based Feature Extraction Algorithms

Zafer Guler[1, *], **Ahmet Cinar**[2], **Erdal Ozbay**[2]
[1]Department of Computer Engineering, Sivas University of Science and Technology
58000 Sivas, Turkey
[2]Department of Computer Engineering, Firat University,
23200 Elazig, Turkey
zaferguler@sivas.edu.tr

***Abstract*—This paper presents a novel object tracking framework for interest point based feature extracting algorithms. The proposed framework uses the feature extracting algorithm without making any changes and it relies on outlier detection, object modelling, and object tracking. At first, the keypoints are extracted by using a feature extraction algorithm. Then, incorrect keypoint matches are detected by the DBScan algorithm. The second step of our tracking framework is object modelling. The object model is defined as a bounding box. The box model has six points and each of these points has its own Gaussian model. Finally, the Gaussian model is performed for object tracking. In object tracking, the old five values are retained to detect incorrect position information. Thus, while the object movements are softened, the instant deviations are eliminated also. Our interest point based object tracking framework (IPBOT) works with any interest point based feature extracting algorithm. Thus, a new algorithm can be added to the object tracking framework with a short integration process. The experiment results show that the proposed tracker significantly improves the success rate of the object tracking.**

***Index Terms*—Feature extraction; Object tracking; SIFT; SURF.**

I. INTRODUCTION

Every object tracking application requires an object detection method [1]. There are two common models for object detection [2]. The first one is to use single frame information. The second is to use combined information, which is computed from multiple frames [2]. There are a lot of methods published in the literature for object detection. These methods are classified into four main categories [2], [3]:

– Point detectors: Speeded-Up Robust Features (SURF) [4], Scale-Invariant Feature Transform (SIFT) [5], and Kanade-Lucas-Tomasi (KLT) detectors [6];

– Segmentation: active contours [7], [8], graph-cut [9], and mean shift [10];

– Background modelling: Mixture of Gaussian [11] and Dynamic Background model [12];

– Supervised classification: Support Vector Machine [13], Neural Networks [14], and Deep Learning [15].

In 2004, the Scale Invariant Feature Transform (SIFT) was published by Lowe to find distinctive invariant features [5]. The SIFT algorithm basically consists of 3 stages. These are keypoint detection, descriptor calculation, and feature matching. During the descriptor establishing stage, SIFT uses a 128-dimensional vector to identify the keypoint. This high dimensional vector causes performance issues and makes the SIFT algorithm run slowly [16]. For solving this problem, the Principal Component Analysis based SIFT (PCA-SIFT) algorithm was proposed in 2004 by Ke and Sukthankar [17]. In PCA-SIFT analysis, the Principal Component Analysis method is used for each keypoint definition. Thus, typically, in order to decrease the high dimensional requirement in the SIFT algorithm, PCA-SIFT is applied. PCA-SIFT is faster than SIFT, but SIFT is more distinctive than PCA-SIFT [18]. The Speed-Up Robust Feature Detector (SURF) developed by Bay is, basically, similar to the SIFT algorithm, but each step of the algorithm is improved [4].

SIFT algorithm and its variants are used in the object tracking applications also. With the algorithm developed by Zhou [19], SIFT keypoints are integrated with the mean shift algorithm. In the proposed approach, the similarity criterion between two neighboring frames is determined by color information and SIFT features [19]. Miao proposed a SURF based object tracking application [20]. In that study, a search space is incrementally estimated for increasing the reusability of the tracked interest point. For computing descriptor, they used online boosting and a classifier based descriptor.

In this paper, we developed an object tracking framework based on for feature extraction algorithms, such as SIFT and its variants.

II. LITERATURE REVIEW

A. SIFT Algorithm

SIFT algorithm achieves robust results against scaling and rotation invariance. However, it requires a high computing capacity. For this reason, SIFT algorithm cannot achieve successful results in real-time systems [5], [21].

Manuscript received 8 March, 2019; accepted 20 October, 2019.
This research was funded by a grant (No. MF16.61) from the Firat University Scientific Research Projects' Unit.

The first step to detect the corresponding points is a convolution operation. The convolution operation is performed between Gaussian filter with different scales of views, and the difference of Gaussian with adjacent images is computed. This process is shown in Fig. 1. The corresponding point, called the keypoint, is defined as the local minimum and maximum between the difference of Gaussian (DOG) scales. Each pixel in the DOG image is compared to the neighbours in the same and adjacent scales. If the pixel is local minimum or maximum, this point is selected as the candidate point [19]. For each candidate point, the four steps are performed as follows.

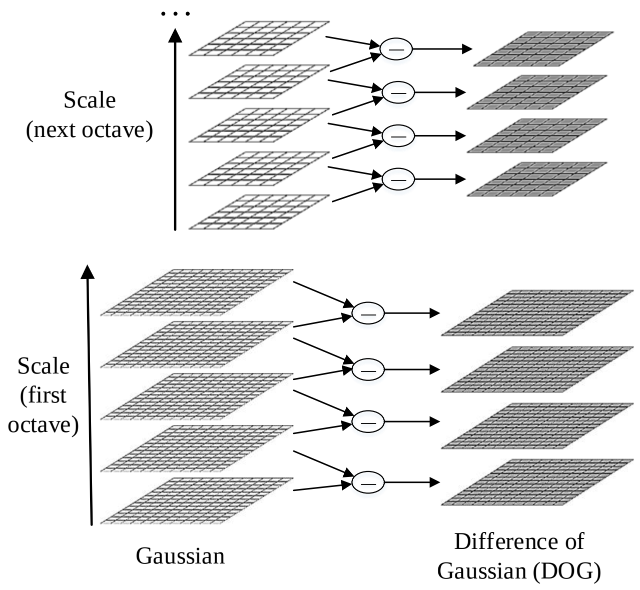

Fig. 1. Computation of the difference of Gaussian [5].

Stage 1. Scale space extreme detection: The scale space is formed by applying the Gaussian filter of images in different scales, and it is defined as the function $L(x, y, \sigma)$. This function is calculated from the convolution of Gaussian $G(x, y, \sigma)$ and input image $I(x, y)$ [5]. To increase the application speed, SIFT uses the DOG instead of Gaussian. Convolved image $D(x, y, \sigma)$, can be computed with (1)

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y), \qquad (1)$$

where k is a constant factor and $*$ is a convolution operation.

After these operations, candidate interest points are selected as the minimum and maximum of DOG [5].

Stage 2. Update location: The location information of each candidate is updated by the color values using the neighbouring pixels [5].

Stage 3. Keypoint refinement and filtering: DOG operation gives a strong response along the edge. Therefore, for stability, low contrast candidates along the edge are eliminated [5].

Stage 4. Keypoint descriptor calculation: Gradients are calculated for each of the remaining interesting points [5]. These gradients are very useful to find the local change in shape distortion and illumination [19].

The SIFT algorithm can detect a greater number of interest points. It is more resident to image deformations also [1]. For real time application, SIFT is comparatively slow. For that reason, SURF algorithm was introduced by Bay, Tuytelaars, and Van Gool [4]. SURF algorithm was inspired by the SIFT algorithm and can be used for object recognition, classification, and registration [16].

B. SURF Algorithm

SIFT and SURF algorithms have similar steps. However, the implementation details in each step are different. The main difference is that SURF algorithm is relatively more efficient than SIFT algorithm [1]. Furthermore, it is suitable for real time applications. The steps required to find the point of interest in SURF algorithm are given in [22].

1. *Calculation of image integral*. The purpose of calculating the image integral is to facilitate the box filter. Image integral is calculated by summing the pixel intensities cumulatively [22].
2. *Box filtering*. A box filter is an effective way of approaching a Hessian Matrix for a given pixel value. In SURF algorithm, Hessian calculation is used for the computing of the interest point [22].
3. *Scale space generation*. With this process, SURF algorithm achieves scale invariance. The same operation is performed in SIFT algorithm, but the calculation in the SURF algorithm is partially different [16]. In SURF algorithm, each filter is applied to the same integral image. This procedure ensures that DOG creation process is achieved using fewer computational cost [22].
4. *Interest point searching*. When addressing a single pixel used in the search process, a 3x3x3 neighbourhood is used to determine, whether this pixel is or not a local maximum. If the central pixel in the search area has the highest intensity value, it is marked as a local maximum. If the central pixel value is greater than the threshold and it is a local maximum, then this pixel is marked as the interest point [22].
5. After each step mentioned above, it is necessary to define a descriptor for each interest point. The steps used to create a descriptor in SURF algorithm are given below.
6. *Orientation*. The main purpose of the orientation step is to provide a directional value for each feature. The orientation value of the features is calculated by the use of the surrounding area of the interest point. Thus, the rotation invariance in SURF algorithm is provided.
7. *Calculation of the descriptor*. The descriptor describes the properties of the features, which surround the interest point. This region is detected with the Hessian-based detector. The next action is to define the characteristic that describes this region. This definition can be identified by drawing a square around the interest point and showing the orientation [22].

C. Other Feature Extraction Algorithms

In addition to SIFT and SURF algorithms, there are many other feature extraction algorithms in the literature. Some of these are PCA-SIFT [17], Colored SIFT (CSIFT) [23,] and Affine-SIFT (ASIFT) [24] algorithms. When these algorithms are examined, there is no algorithm that works successfully in every case. SIFT and its variants can be examined under five different conditions. These conditions are scaling, rotating, illumination, blur, and affine invariance [16]. SIFT achieves successful results in scale and rotation invariance. In other cases, the average results are obtained. The CSIFT algorithm is more successful than SIFT in blur invariance. In addition, it is as successful as SIFT in scale and rotation invariance, but the runtime is slower than of the SIFT. The ASIFT algorithm achieves successful results in affine invariance. In other cases, average results are obtained and the runtime is slower than of the other algorithms also. The PCA-SIFT algorithm generally gives above average results in scaling, rotation, illumination, and blur invariance. While in the others, it gives average results. Finally, the average results are obtained with the SURF algorithm for feature extraction, but it is the best of these algorithms in terms of runtime [16].

As can be seen, each algorithm has advantages and disadvantages. In every case, there is no algorithm that works with optimum accuracy and performance. Feature extraction algorithms can determine, whether the object exists in the next image/frame, and specify, whether an object exists in an image, unless the object view is completely changed. For this reason, they are frequently preferred in object tracking applications.

One of the biggest problems in object tracking applications is to detect that the object is the same object when the object disappears and appears again. Object extraction algorithms are one of the most common algorithms used for this purpose. In [16], Wu compared SIFT and its variants. When we look at SIFT and its variants in terms of runtime, it is seen that the fastest running algorithm is SURF. In addition, the SIFT algorithm is the second fastest algorithm in terms of operating speed. Since the runtime of the algorithm is very important in object tracking applications, in this study, we compared these two algorithms and the Graphics Processing Unit (GPU) version of SURF algorithm (GPU-SURF) with the proposed algorithm.

We used OpenCV Compute Unified Device Architecture (CUDA) implementation to calculate SURF features. The GPU-SURF algorithm produces similar results with the SURF algorithm. In cases, where the search space is too small, the GPU-SURF algorithm can produce incorrect results. Therefore, in this case, the keypoint extraction by the SURF algorithm is performed. This situation occurred only when calculating some object keypoints. As the video frame size is 640x480, the GPU-SURF algorithm is executed for keypoint extraction from video frames. In case the search space is small, the operating speeds of SURF and GPU-SURF algorithms are similar.

III. PROPOSED ALGORITHM

This section provides detailed information about the interest point based object tracking framework (IPBOT). The flow chart of the IPBOT is given in Fig. 2. In our object tracking application, we did not interfere the structure of the feature extraction algorithm and the matching process. Instead, the developed algorithm offers improvements on matching features. It is possible to divide the improvements into three stages. These are outlier detection, object modelling, and object tracking operations.

A. *Outlier Detection with DBSCAN Algorithm*

In IPBOT, the feature matching process is obtained by the feature extraction algorithm. The accuracy of the matches has a critical importance for the correct identification of the object position. Yet there is no such an algorithm that you have a 100 % success rate. To do this, it is very important to detect the wrong matching points correctly.

We are not involved in feature matching in this application. Instead, this study is focused on detecting incorrect matching. When the matchings are examined, it is observed that, generally, there are more correct than incorrect matchings. Besides, it is indicated that incorrect matchings are at points away from the object. Therefore, we can reduce the problem to find discrete points in a coordinate system.

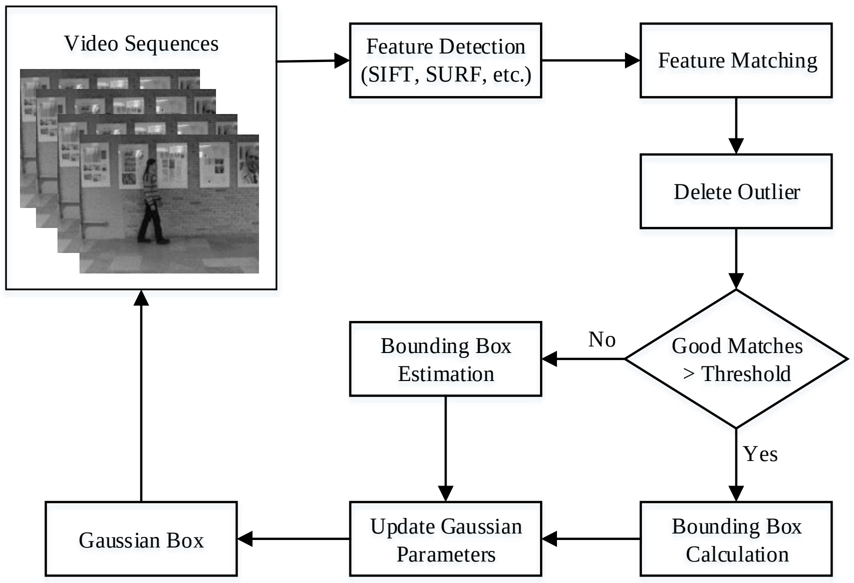

Fig. 2. The structure of the IPBOT algorithm.

DBScan algorithm [25], which is a statistical classification method, is used for the detection of discrete points. The DBScan algorithm is a density-based clustering algorithm. It can easily classify the clusters with different shape and different number of elements. It is also simple to use and implement. The structure of the DBScan algorithm is given below and shown in Fig. 3 [25]:

1. Starts from a desired point in the dataset (red point);
2. The distance between this point and other points is calculated. If the calculated distance is less than the epsilon value (eps), it is included in the cluster (blue points);
3. In this way, all detected points are considered to be central, and new cluster elements are detected iteratively. For this purpose, the Depth First Search approach is used;
4. Clustering process is executed for all points in the data set iteratively and finds all other groups (green points);
5. If the set is less than the minimum number of elements (MinPts), it is marked as noise (remaining black points).

In the DBScan algorithm, the most important parameters are the eps and MinPts parameters. Normally, eps is obtained by calculating the mean of the distance between points. However, in this study, we preferred to use a fixed value. The general algorithm of outlier detection with DBScan is given in Algorithm 1 and an example demonstration of the DBScan on interest points is given in Fig. 4. Algorithm 1 shows that the MinPts parameter is calculated according to the number of cluster elements. Moreover, as can be seen in Fig. 4, number of 5 matching matches (red features) are marked as incorrect and these matches are excluded from calculation. As a result, the accuracy of object detection significantly increased.

Algorithm 1. Outlier Detection with DBSCAN.

```
  Inputs: kp= keypoints, mt= matches
  Outputs: kp= kp results mt: mt results

1:  for all keypoints in dataset
2:      read next_point in keypoints
3:      if (next_point not visited)
4:          DFS (next_point)
5:      endIf
5:  endFor
6:
7:  max = find max element set
8:  if (1<max<6) minPts = 1
9:  else if (6≤max<12) minPts = 2
10: else if (12≤max<18) minPts = 3
11: else if (18≤max<24) minPts = 4
12: else minPts = 5
13: Delete sets that do not have enough elements
14: Save remaining keypoints and matches
```

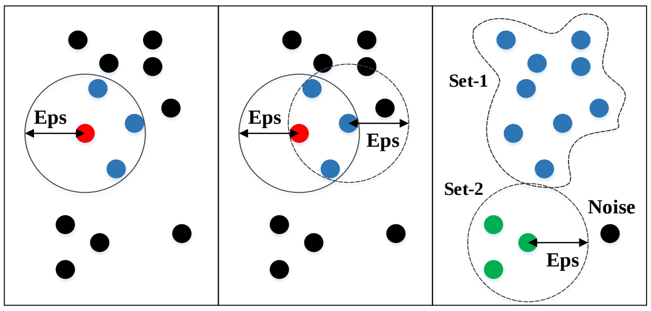

Fig. 3. The structure of the DBScan algorithm.

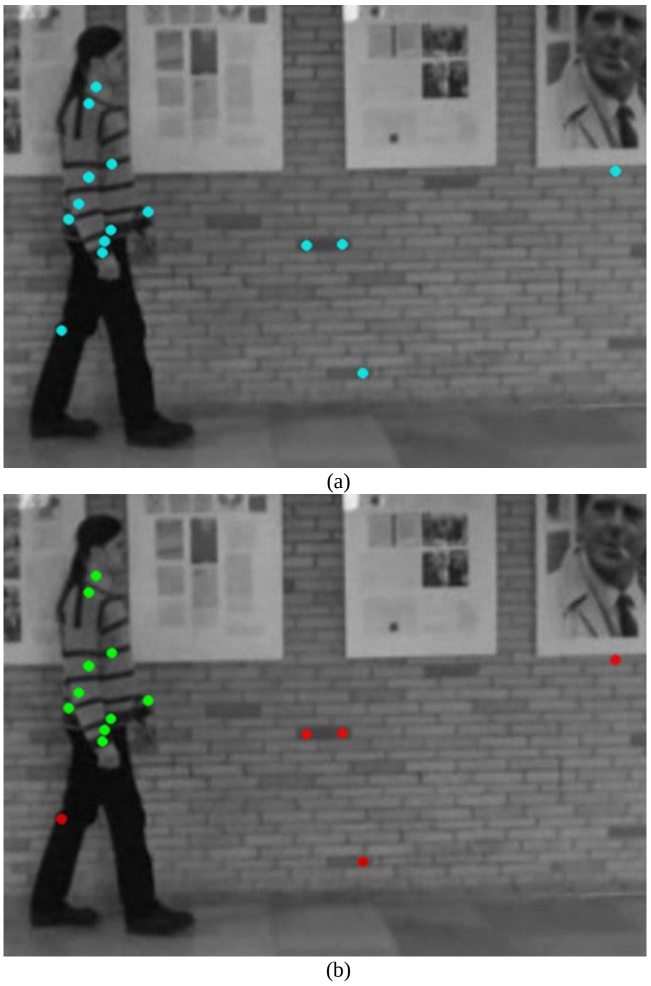

Fig. 4. Example demonstration of the DBScan algorithm on SIFT interest point: (a) Object keypoints extracted with SIFT shown in cyan color; (b) Final object keypoints after DBScan algorithm shown in green color. Incorrect object keypoint shown in red color.

B. *Object Modelling*

After clearing incorrect matching, the position of the object is estimated approximately. To estimate the object position, a homography calculation is made between the object and the video frame. A homography is a 3x3 matrix transformation that maps the points in an image to the corresponding points in the other image [26]. However, at least four matching points are required to calculate homography accurately. For homography calculation, a random Sample Consensus algorithm (RANSAC) is used [27]. The RANSAC algorithm is an algorithm developed by Fischler and Bolles in 1981. It ensures a robust fitting of models. In the experiments, successful results can be obtained, in case of outliers. However, we chose to apply at least 10 matching points in this application. Because, if there are incorrect matches and the number of matches is low, the object position cannot be found successfully.

In this study, the object is defined by box representation. As shown in Fig. 5, there are six values in our object model. As a result of calculating matching keypoints, x_{kp} and y_{kp} values are obtained, and by calculating homography, x, y, w, and h values are achieved.

The Gaussian model is used for these six values. Every value has its own Gaussian model. First, we initialize object models using the recent history of t object positions. Then, the probability function is calculated at a given value at time t as follows

$$P(X_t) = w_{i,t} \times n(X_t, \mu_i, \Sigma_i), \tag{2}$$

where X_t object modelling value in frame t; $w_{i,t}$ is the weight of the distribution in frame t; μ_i is the mean of the distribution; Σ_i : The standard deviation of the distribution. Here $n(X_t, \mu_i, \Sigma_i)$ is Gaussian probability density function and calculated as in (3)

$$n(X_t, \mu_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)}\sigma} e^{\frac{-X_t^2}{2\sigma^2}}, \tag{3}$$

where σ is the standard deviation. After initialization the all model, we should separate the model as bounding box estimation (BBE) or bounding box calculation (BBC). x, y, w, and h are Gaussian models used for the selection for the next algorithm. The algorithm selection process is also given in (4)

$$n = \sum_{k=1}^{g} \begin{cases} 1 \; if \;\; w_k > T, \\ 0 \; otherwise, \end{cases} \tag{4}$$

where g is the total number of Gaussian model and T is the threshold value. Here, it is checked, whether the Gaussian value is greater than the T threshold. If $n > 2$, the BBC is performed. The BBC operation is the process of generating values, which have not reached the threshold value by applying Gaussian model. Thus, four values are obtained for the object model. If $n \leq 2$, the BBE is performed. The x_{kp} and y_{kp} values are controlled with (4). If both exceeds the threshold value, the center of the object is determined by the values of x_{kp} and y_{kp}. If one of them does not exceed the threshold value, the Gaussian models are used for both of them to determine the center of the object.

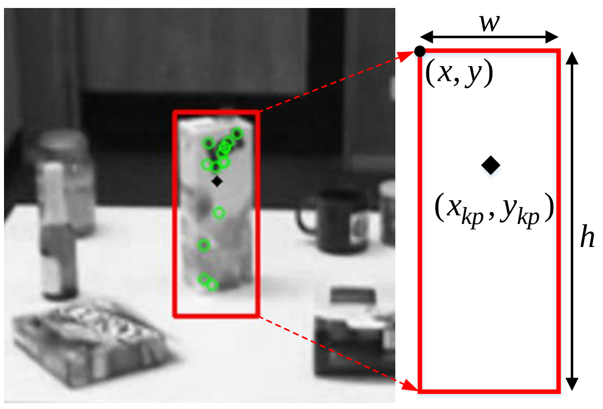

Fig. 5. Our 6-point object model.

Finally, the w and h values of the object are calculated according to the Gaussian distribution and the x and y values of the object are calculated using the center of the object as in (5) and (6):

$$x = xi_{kp} - w_i \frac{xo_{kp}}{w_o}, \tag{5}$$

$$y = yi_{kp} - h_i \frac{yo_{kp}}{h_o}. \tag{6}$$

(5) and (6) provide the equivalence (7)

$$\frac{w_{o_1}}{w_{o_2}} \equiv \frac{w_{i_1}}{w_{i_2}} \Rightarrow \frac{h_{o_1}}{h_{o_2}} \equiv \frac{h_{i_1}}{h_{i_2}}. \tag{7}$$

All parameters of (7) are shown in Fig. 6.

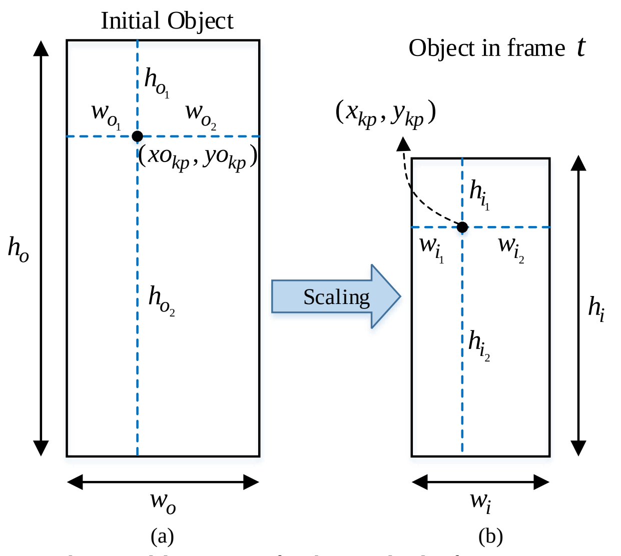

Fig. 6. Object model parameters for object and video frame.

After all these operations are completed, there are now two possibilities left, whether the object is found or not. If the object is found successfully, the next step is the object tracking. If not, it is taken to the next frame and the operations start from the beginning.

C. Object Tracking

In this section, Gaussian model is applied to the object tracking. Our interest point based object tracking may incorrectly predict the position of the object in some instant frames. This condition is usually temporary, and, in subsequent frames, this condition usually improves. Although this error is largely addressed by the calculations given in Section III-A, some frames may still be inaccurate. Nowadays, videos usually have a value of 30 fps. This means that the time between two frames is about 0.03 seconds. Based on this theory, Gaussian smoothing process is applied by using the object position in last 5 frames, because the last object positions are important in this process. However, in case of a mismatch in the last object position, this error is softened with Gaussian. Also, they can also adapt to abrupt the motion of the object. As a result of this study, the object tracking accuracy is increased between 2 % to 10 %.

IV. EXPERIMENTAL RESULTS

In this section, we present the experimental results. All experiments are performed on a machine equipped with an

Intel Core i7-3770 CPU at 8GB RAM and a GeForce GTX 660 Ti GPU. The GPU used in this study includes 7 streaming multiprocessors (SM) and each SM has 192 CUDA processors. The global memory size of the GPU is 2 GB and it is accessed via the GDDR5 interface. Double precision floating-point arithmetic is supported on the GPU architecture.

For object tracking experiments, we use Bonn Benchmark on Tracking (BoBoT) Dataset [28]. The BoBoT dataset includes 12 short video sequences provided an AVI format with 320x240 pixels at 25 fps. The number of frames on video in the BoBoT dataset ranges from 305 to 1308, and the ground truth data are given for each frame also. Due to the low video resolution, in some cases, the object can be very small. In this study, we convert the video resolution to 640x480. The first frames of the videos in the dataset and the targeted objects to be tracked are given in Fig. 7.

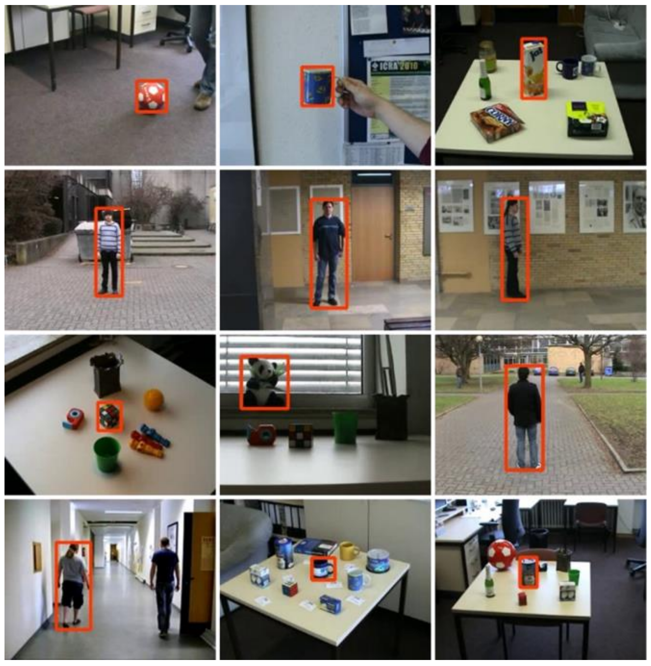
Fig. 7. First frames of BoBoT video dataset.

In the BoBoT dataset, different features are tested on each video sequences. For example, on the first video sequence, its aim is to track a Football Ball. In this video sequence, an abrupt motion and camera motion are tested. On the third video sequence, scaling and appearance changes are tested, as well as abrupt motion and moving camera. Table I shows the difficulties for each video sequence in the BoBoT dataset [29]. The features tested on each video sequence are indicated by the symbol "+" in Table I. Other untested features are indicated by the symbol "-". In order to evaluate the performance of the developed application, the first frame is taken from the video and the object is extracted with the ground truth. This extracted image is selected as a base image and the remaining frames are used for testing.

First 9 videos in the dataset were tested. Our tracker was integrated with SIFT, SURF, and GPU-SURF algorithms, and we compared the traditional methods with our tracker. As a result of the experiments, similar results were obtained on SURF and GPU-SURF algorithms. The only difference between two algorithms is that, the GPU-SURF algorithm works faster.

TABLE I. BOBOT DATASET VIDEO SEQUENCE AND THEIR DIFFICULTIES FOR OBJECT TRACKING [29] (D1: SCALE, D2: OCCLUSION, D3: CAMERA MOTION, D4: ABRUPT MOTION, D5: SIMILAR APPEARANCE, D6: SCENE CLUTTER, D7: MOTION DIRECTION, D8: APPEARANCE CHANGES, D9: DISAPPEARANCE, D10: ILLUMINATION).

Sequence No.	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
1	-	-	+	+	-	-	+	+	-	-
2	+	-	+	-	-	+	-	+	-	-
3	+	-	+	+	-	-	+	+	-	-
4	-	-	+	-	-	-	-	+	-	-
5	-	+	+	-	-	-	-	-	-	-
6	-	+	+	-	+	-	-	+	+	-
7	-	-	+	-	-	-	-	+	-	-
8	-	-	-	-	-	-	-	+	-	+
9	-	+	+	-	+	-	-	+	+	-
10	-	+	+	-	-	-	-	+	+	-
11	+	-	+	-	+	-	-	+	-	-
12	+	-	+	-	-	-	-	+	-	-

We evaluated our IPBOT tracker by comparing it with SIFT and SURF algorithms. Comparison results are shown in Fig. 8. In the figure, a video stream is given on each line. Also, the video no is shown on the left side of the figure. The results of the tracking algorithms on the figure are shown with a different color for each algorithm. The proposed IPBOT algorithm is indicated by green color, SIFT, SURF, and ground truth are indicated by red, blue, and turquoise colours, respectively. Our IPBOT tracker was tested with different challenges and has generally achieved satisfactory results. The results showed that the IPBOT algorithm provides more successful results than the other algorithms. In some cases, the standard SIFT and SURF algorithms cannot produce results for object tracking. For example, in the second video sequence, the SURF algorithm generally does not produce results. Similarly, both SIFT and SURF algorithms cannot generate adequate results for the 6th video sequence.

As we mentioned in the previous section, SIFT algorithm extracts more keypoint than the SURF and GPU-SURF algorithms. For this reason, the object tracking framework is more successful with the SIFT algorithm. In order to measure the success rate, the average of Sensitivity (Se) and Precision (Pr) values was used. Their equation is given in (8)

$$Se = \frac{TP}{TP + FN} \Rightarrow Pr = \frac{TP}{TP + FP}. \tag{8}$$

In these equations, the True Positive (TP) value indicates the correctly estimated object point number, while the False Positive (FP) and False Negative (FN) values indicate the number of incorrect object points and missed number of object points that cannot be detected, respectively.

The developed algorithm is suitable for SIFT and its variants. They fail when the object view changes dramatically. As seen in Table II, the success rate of SIFT algorithm is low, because the object view changed at sequence numbers 1, 4, and 7. The developed algorithm provided great improvements in these type of videos. As seen in the Table II, the success rate of object tracking was greatly increased. Only in the 5th and 8th video sequence, there was a slight increase in success rate. This is because the SIFT and its variants were already successful in these videos.

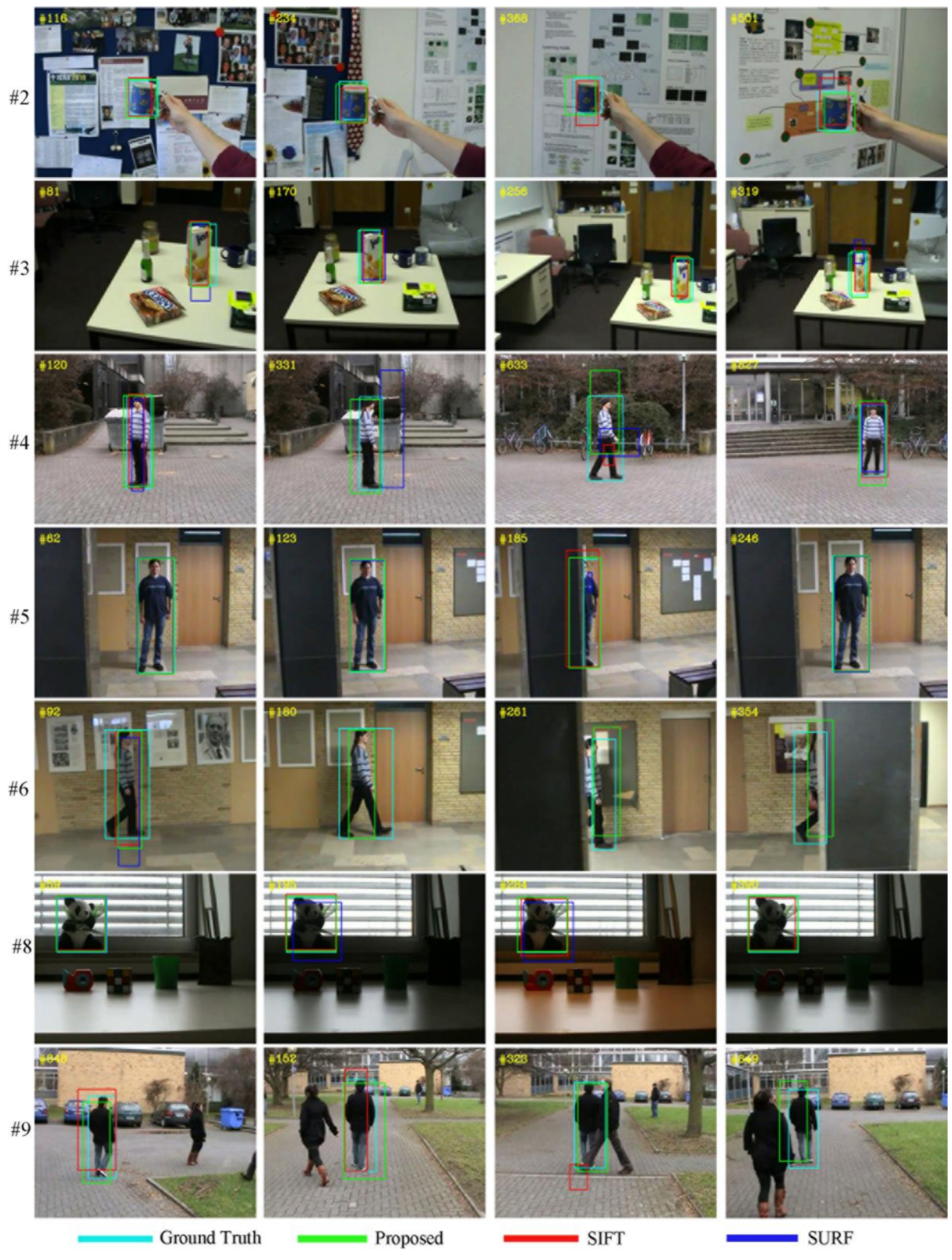

Fig. 8. Tracking results for video sequence.

As a result, the success rate increased from 56.88 % to 82.2 % for the SIFT, from 51.05 % to 77.25 % for SURF, and from 48.61 % to 78.11 % for GPU-SURF. The success rate of the SURF and GPU-SURF algorithms is similar, but GPU-SURF is much faster in terms of runtime than the other two algorithms. The runtime of all algorithms based on the frame per second (fps) are given in Table III. In addition, the percentage of developments is given in parentheses. As seen in Table III, the slowest algorithm is SIFT algorithm, while the fastest algorithm is the GPU-SURF algorithm. The IPBOT algorithm is integrated into SIFT, SURF and GPU-SURF algorithms. In addition, the GPU-SURF + IPBOT algorithm achieved an average of 39.15 fps. This suggests that it may be appropriate for a real-time application.

TABLE II. COMPARATIVE RESULTS OF SUCCESS RATE WITH SIFT, SURF, AND GPU-SURF ALGORITHMS. THE PERCENTAGE OF DEVELOPMENTS IS GIVEN IN PARENTHESES.

Sequence No.	**SIFT**	**SURF**	**GPU-SURF**	**SIFT + IPBOT**	**SURF + IPBOT**	**GPU-SURF + IPBOT**
1	33.5	24.0	24.7	73.2 (% 118.4)	60.6 (% 152.4)	63.1 (% 155.7)
2	46.2	39.9	33.9	85.5 (% 85.0)	86.3 (% 116.5)	82.9 (% 144.3)
3	76.3	71.2	68.0	91.6 (% 20.2)	87.7 (% 23.1)	85.8 (% 26.1)
4	45.9	38.3	37.8	81.1 (% 76.8)	65.9 (% 71.9)	69.5 (% 83.9)
5	93.1	91.1	90.6	93.8 (% 0.7)	93.7 (% 2.8)	92.9 (% 2.6)
6	42.4	42.8	42.9	67.2 (% 58.7)	66.7 (% 55.8)	71.6 (% 66.8)
7	39.4	32.5	32.1	81.7 (% 107.1)	77.3 (% 138.1)	77.7 (% 141.9)
8	87.7	79.3	71.0	97.8 (% 11.4)	93.7 (% 18.1)	93.5 (% 31.7)
9	47.3	40.3	36.5	67.8 (% 43.3)	63.5 (% 57.6)	66.0 (% 81.0)
Average	56.9	51.1	48.6	82.2 (% 44.5)	77.3 (% 51.3)	78.1 (% 60.7)

TABLE III. COMPARATIVE RESULTS OF RUNTIME (FPS) WITH SIFT, SURF, AND GPU-SURF ALGORITHMS. THE PERCENTAGE OF DEVELOPMENTS IS GIVEN IN PARENTHESES.

Sequence No.	**SIFT**	**SURF**	**GPU-SURF**	**SIFT + IPBOT**	**SURF + IPBOT**	**GPU-SURF + IPBOT**
1	9.1	18.8	48.9	9.0 (%-0.2)	18.4 (%-2.4)	48.1 (%-1.5)
2	6.3	12.1	37.8	6.3 (%-0.7)	11.8 (%-1.9)	37.3 (%-1.4)
3	7.7	14.2	42.7	7.7 (%-0.2)	14.2 (%-0.6)	41.3 (%-3.3)
4	4.6	9.9	27.0	4.6 (%-1.2)	10.0 (%0.9)	29.1 (%-7.8)
5	6.9	13.2	37.6	6.8 (%-1.1)	13.2 (%-0.0)	36.5 (%-2.9)
6	7.4	12.7	34.4	7.2 (%-1.7)	12.8 (%-0.8)	36.3 (%-5.5)
7	8.8	18.4	51.7	8.6 (%-2.4)	18.2 (%-1.5)	51.0 (%-1.4)
8	8.0	14.8	39.2	7.9 (%-1.2)	14.8 (%-0.1)	41.4 (%-5.6)
9	4.5	10.9	29.8	4.5 (%-0.3)	11.0 (%-0.8)	31.4 (%-5.2)
Average	7.0	13.9	38.8	7.0 (%-0.8)	13.8 (%-0.6)	39.2 (%-0.9)

V. CONCLUSIONS

Our main goal is to create an object tracking framework by getting rid of the incorrect matching keypoints generated by the feature extraction algorithms. The feature extraction algorithms can extract many properties of the object and use these properties successfully. With this presented framework, all these algorithms can be used for object tracking application.

For a feature extraction algorithm, after matches the properties, outlier detection, object modelling, and object tracking are performed. Outlier detection is performed using the DBScan algorithm. The DBScan algorithm minimizes keypoints mismatches that occur as a result of keypoints matching between the object and the video frame. Thus, even when the number of matching keypoints is small, the object position is successfully detected. Another improvement is performed at the object modelling stage. The object structure is modelled using 6 points, and the Gaussian model is used for each point. Normally, object detection can be done via keypoints, but the number and accuracy of the matching keypoints is important for successful object detection. If there are few keypoints matches, it must be error-free, but only, if there are many keypoints matching, the incorrect points may be accepted. Our proposed method estimates the object position accurately even if there are few and incorrect matching keypoints. This progress is possible with the 6-point Gaussian object model.

The IPBOT algorithm is adapted to the traditional SIFT, SURF, and GPU-SURF algorithms and the adapted IPBOT methods are compared with the traditional algorithms. According to the results, the runtime of the IPBOT algorithm is approximately the same as of the traditional algorithms. As seen in the experiments, there is no significant runtime difference between the traditional algorithms and the IPBOT algorithm. Furthermore, IPBOT provides to increase the object tracking performance by about 50 %. In future work, the authors will try to evaluate this developed algorithm using background model or supervised classification.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] N. Y. Khan, B. McCane, and G. Wyvill, “SIFT and SURF performance evaluation against various image deformations on benchmark dataset”, in *Proc. of International Conference on Digital Image Computing: Techniques and Applications*, 2011, pp. 501–506. DOI: 10.1109/DICTA.2011.90.

[2] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey”, *ACM Computing Surveys*, vol. 38, no. 4, pp. 1–45, 2006. DOI: 10.1145/1177352.1177355.

[3] S. A. Mahmoudi, M. Kierzynka, P. Manneback, and K. Kurowski, “Real-time motion tracking using optical flow on multiple GPUs”, *Bulletin of the Polish Academy of Sciences-Technical Sciences*, vol. 62, no. 1, pp. 139–150, 2014. DOI: 10.2478/bpasts-2014-0016.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (SURF)”, *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008. DOI: 10.1016/j.cviu.2007.09.014.

[5] D. G. Lowe, “Distinctive image features from scale-invariant keypoints”, *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. DOI: 10.1023/B:VISI.0000029664.99615.94.

[6] S. Jianbo and C. Tomasi, “Good features to track”, in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, 1994, pp. 593–600. DOI: 10.1109/CVPR.1994.323794.

[7] V. Caselles, R. Kimmel, and G. Sapiro, “Geodesic active contours”, *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–79, Feb.–Mar. 1997. DOI: 10.1023/A:1007979827043.

[8] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes - active contour models”, *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988. DOI: 10.1007/BF00133570.

[9] J. B. Shi and J. Malik, “Normalized cuts and image segmentation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, Aug. 2000. DOI: 10.1109/34.868688.

[10] 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.

[11] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking”, in *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Fort Collins, CO, USA, 1999, vol. 2, pp. 246–252. DOI: 10.1109/CVPR.1999.784637.

[12] D. Russell and G. Shaogang, “A highly efficient block-based dynamic background model”, in *Proc. of IEEE Conference on Advanced*

Video and Signal Based Surveillance, Como, Italy, 2005, pp. 417–422. DOI: 10.1109/AVSS.2005.1577305.

[13] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection", in *Proc. of 6th International Conference on Computer Vision*, Bombay, India, 1998, pp. 555–562. DOI: 10.1109/ICCV.1998.710772.

[14] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–38, 1998. DOI: 10.1109/34.655647.

[15] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification", in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, RI, USA, 2012. DOI: 10.1109/CVPR.2012.6248110.

[16] J. Wu, Z. M. Cui, V. S. Sheng, P. P. Zhao, D. L. Su, and S. R. Gong, "A comparative study of SIFT and its variants", *Measurement Science Review*, vol. 13, no. 3, pp. 122–131, 2013. DOI: 10.2478/msr-2013-0021.

[17] K. Yan and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors", in *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004, vol. 2, pp. 506–513. DOI: 10.1109/CVPR.2004.1315206.

[18] L. Juan and O. Gwon, "A comparison of SIFT, PCA-SIFT and SURF", *International Journal of Image Processing*, vol. 3, no. 4, pp. 143–152, 2009.

[19] H. Y. Zhou, Y. Yuan, and C. M. Shi, "Object tracking using SIFT features and mean shift", *Computer Vision and Image Understanding*, vol. 113, no. 3, pp. 345–352, 2009. DOI: 10.1016/j.cviu.2008.08.006.

[20] Q. Miao, G. Wang, C. Shi, X. Lin, and Z. Ruan, "A new framework for on-line object tracking based on SURF", *Pattern Recognition Letters*, vol. 32, no. 13, 2011, pp. 1564–1571. DOI: 10.1016/j.patrec.2011.05.017.

[21] M. Guerrero, "A comparative study of three image matching algorithms: Sift, Surf, and Fast", M. S. Thesis, Civil and Environmental Eng., Utah State Univ., 2011.

[22] S. M. Jurgensen, "The rotated speeded-up robust features algorithm (R-SURF)", M. S. Thesis, Electrical Eng., Naval Postgraduate School, 2014.

[23] A. E. Abdel-Hakim and A. A. Farag, "CSIFT: A SIFT descriptor with color invariant characteristics", in *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, vol. 2, pp. 1978–1983. DOI: 10.1109/CVPR.2006.95.

[24] G. Yu and J.-M. Morel, "ASIFT: An algorithm for fully affine invariant comparison", *Image Processing On Line*, pp. 11–38, 2011. DOI: 10.5201/ipol.2011.my-asift.

[25] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", in *Proc. of Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, pp. 226–231, 1996.

[26] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features", *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59–73, 2007. DOI: 10.1007/s11263-006-0002-3.

[27] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. DOI: 10.1145/358669.358692.

[28] H. Shao, T. Svoboda, T. Tuytelaars, and L. Van Gool, "HPAT indexing for fast object/scene recognition based on local appearance", in *Proc. of 2003 Image and Video Retrieval: Second International Conference*, vol. 2728, pp. 71-80, 2003. DOI: 10.1007/3-540-45113-7_8.

[29] S. Dubuisson and C. Gonzales, "A survey of datasets for visual tracking", *Machine Vision and Applications*, vol. 27, no. 1, pp. 23–52, Jan. 2016. DOI: 10.1007/s00138-015-0713-y.