Makara Journal of Technology

Volume 25 | Issue 1 Article 4

4-1-2021

Mean-shift Object Tracking Algorithm with Systematic Sampling Technique

Yoanes Bandung
School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung 40116, Indonesia, bandung@stei.itb.ac.id

Aris Ardiansyah
School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung 40116, Indonesia

Recommended Citation
Bandung, Yoanes and Ardiansyah, Aris (2021) "Mean-shift Object Tracking Algorithm with Systematic Sampling Technique," *Makara Journal of Technology*: Vol. 25: Iss. 1, Article 4.
DOI: 10.7454/mst.v25i1.3789
Available at: https://scholarhub.ui.ac.id/mjt/vol25/iss1/4

Makara J. Technol. 25/1 (2021), 22–29
doi: 10.7454/mst.v25i1.3789

Mean-shift Object Tracking Algorithm with Systematic Sampling Technique

Yoanes Bandung* and Aris Ardiansyah

School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung 40116, Indonesia

**E-mail: bandung@stei.itb.ac.id*

Abstract

Mean shift is a fast object tracking algorithm that only considers pixels in an object area, hence its relatively small computational load. This algorithm is suitable for use in real-time conditions in terms of execution time. The use of histograms causes this algorithm to be relatively resistant to rotation and changes in object size. However, its resistance to lighting changes is not optimal. This study aims to improve the performance of the algorithm under lighting changes and reduce its processing time. The proposed technique involves the use of sampling techniques to reduce the number of iterations, optimization of candidate search object locations using simulated annealing, and addition of tolerance parameter to optimize object location search and area-based weighting instead of the Epanechnikov kernel. The results of the one-tail t-test with two independent sample groups reveal that the average performance of the proposed algorithm is significantly better than that of the traditional mean-shift algorithm in terms of resistance to lighting changes and processing time per video frame. In the test involving 999 frames of video images, the average processing time of the proposed algorithm is 83.66 ms, whereas that of the traditional mean-shift algorithm is 116.86 ms.

Abstrak

Algoritma Pelacakan Objek Berbasis Mean-Shift Dengan Sistematik Sampling. Mean-shift adalah algoritma pelacakan objek yang cepat karena hanya mempertimbangkan piksel pada area objek. Selain itu, algoritma ini memiliki beban komputasi yang lebih kecil sehingga sangat sesuai untuk digunakan dalam kondisi waktu nyata dalam hal waktu eksekusi. Penggunaan histogram citra menyebabkan algoritma ini relatif tahan terhadap rotasi dan perubahan ukuran objek. Namun, daya tahan algoritma terhadap perubahan pencahayaan belum optimal. Penelitian ini bertujuan untuk memperkenalkan penggunaan teknik *sampling pixel* untuk mengurangi beban data yang perlu diolah dalam pemrosesan dan meningkatkan kinerja algoritma dalam kondisi perubahan pencahayaan. Teknik yang diusulkan meliputi penggunaan teknik pengambilan sampel untuk mengurangi jumlah iterasi, optimalisasi lokasi objek pencarian kandidat menggunakan Algoritma *Simulated Annealing*, penambahan parameter toleransi untuk mengoptimalkan pencarian lokasi objek, dan pembobotan berbasis area sebagai pengganti *kernel* Epanechnikov. Dengan menggunakan statistik uji *one tail t-test* dengan dua kelompok sampel independen, hasil tes menunjukkan bahwa rata-rata kinerja algoritma yang diusulkan lebih cepat secara signifikan daripada algoritma mean-shift dalam hal waktu pemrosesan per *frame* video dan ketahanan terhadap perubahan pencahayaan. Hasil pengujian dengan 999 *frame* gambar video memberikan rata-rata waktu pemrosesan dari algoritma yang diusulkan adalah 83,66 ms sedangkan algoritma mean-shift adalah 116,86 ms.

Keywords: area-based weighting, mean-shift, object tracking, systematic pixel sampling,

1. Introduction

Object tracking is developed to recognize objects and estimate their position at each time change. The existence of object tracking is closely related to other fields, such as distance learning [1], [2], surveillance [3]–[8], robotic navigation [9], vehicle localization [10], [11], and smart cities [3]. One of the problems in object tracking algorithms include the need to ensure that they follow objects in real time and that their resistance to changes in environmental conditions, such as lighting, is increased. In general, object tracking algorithm speed is greatly influenced by the number of operations and iterations. The extent of operations and iterations depends on the image resolution and mathematical operation, as well as on the condition of the object undergoing rotation or size changes due to changes in camera distance.

Several related studies on the implementation of object tracking have developed algorithms for detecting sudden movements [12], tracking objects on low frame rate videos [8], and performing multistage tracking [6]. Other studies have focused on the optimization of existing algorithms or their combination [13]–[18]. In [19], a mean-shift algorithm was used to track moving objects with changes in scale and orientation. The results indicate that the accuracy of a mean-shift algorithm increases when the input increment area parameter is increased; however, the time needed for tracking also increases. Existing research has not shown any optimal increment area to obtain high accuracy while minimizing the time needed to carry out tracking. The research in [20] used a mean-shift algorithm as the basis for forming the mean-shift vector-based shape feature algorithm to improve classification accuracy in large-resolution spatial images. The research in [21] combined mean shift and fuzzy logic to detect waveform decomposition.

Most object tracking algorithms represent an object on the basis of its component tone or color. Objects with the same color composition are assumed to be the same. The use of color components to simplify implementation is considered sufficient to enable object tracking algorithms to represent objects while maintaining their computational load in the optimum state so that the processing time does not decrease. However, the color of an object is greatly affected by lighting conditions in the environment around the object. Changes in lighting may affect and change the component tone or color of an object. Several approaches are available to eliminate the effect of lighting on an object being tracked; they include using robust illumination normalization [22] and a method called quotient image [23, 24]. Kernel-based object tracking that is based on a mean-shift algorithm uses Epanechnikov kernel to reduce the effect [25] in which pixels that are located far from the midpoint of a given object become susceptible to occlusion and other objects influence the background, including luminance [26]. Using this kernel yields a difference in the weight of the image pixel based on the distance from the pixel to the midpoint of the object area; the farther a pixel is from the midpoint of the object, the smaller the weight of the pixel.

In real life, the object to be tracked moves in a dynamic environment and sometimes moves quickly. Therefore, the object tracking algorithm used should also be able to work quickly. The faster an algorithm is in determining the position of an object, the better its application will be. Hence, a study is needed to develop a tracking algorithm that can track objects rapidly. The current work is aimed at developing a mean-shift algorithm for estimating the location of objects and their movements in each given video frame (tracking objects on video) by using a static camera sensor.

2. Review of Mean-Shift Algorithm and Proposed Solution

The proposed algorithm was developed on the basis of the kernel-based object tracking algorithm introduced by Dorin Comaniciu in 2000. To develop the proposed algorithm, we conducted research on the workflow of this kernel-based algorithm and then identified potential workflows or capabilities that could be improved. Several considerations were derived from the study.

Number of Iterations. The existing algorithm uses only pixels in the area of the object being tracked, that is, the region of interest (ROI), and not all the pixels in the image [26]. The pixels in the ROI represent the object being tracked; however, the greater the number of pixels in the ROI is, the greater the number of pixels that must be processed. This increased processing requirement will affect the number of iterations of the algorithm, especially in the image histogram formation and positioning of the tracked object. The pixel sampling technique for use in ROIs was proposed to reduce the number of pixels to be processed by the algorithm without altering the level of representation of the histogram.

Calculation of Kernel Values. The kernel value is used for pixel weighting in the kernel-based tracking algorithm. The pixels located farther from the center of the ROI tend to be affected by background, other objects, or lighting conditions. The kernel value calculation is performed per pixel for each histogram formation and candidate location search. The following steps are needed to form a histogram by weighting kernel values: (1) Determine the center coordinates of the ROI as the kernel value is based on the Euclidean distance from the pixel to the center point of the object. (2) Normalize the coordinates of the original point in the range of -1 to 1 with the center of the image serving as the center point. (3) Calculate the kernel distance, which is the Euclidean distance from the center point of the ROI. (4) Calculate the kernel value (pixel weight) by using the Epanechnikov kernel function [26]. (5) Update the histogram by adding the kernel values to the image histogram.

In addition, the kernel values of pixels with insignificant weights located far from the center of the ROI require frequent calculations. This is due to the larger number of pixels in this position than in the position close to the center of the ROI, which has significant pixel weights (Figure 1). This study also aims to improve the solution by using color models that are relatively resistant to lighting changes and by adding search area tolerance in determining the position of object movement.

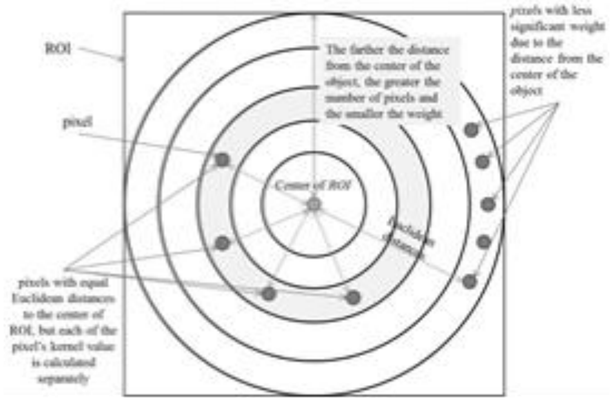

ROI
pixel
The farther the distance from the center of the object, the greater the number of pixels and the smaller the weight
pixels with less significant weight due to the distance from the center of the object
Center of ROI
Euclidean distance
pixels with equal Euclidean distances to the center of ROI, but each of the pixel's kernel value is calculated separately

Figure 1. Illustration of Epanechnikov Kernel. The Farther the Distance from the Center of the Object, the Greater the Number of Pixels and the Smaller the Kernel Value

3. Proposed Algorithm

The proposed algorithm works in two stages, namely, estimation of the direction of movement of an object based on changes in sample pixel color in the area of the object and estimation of the coordinates of the object's movement in that direction. The tracked object is represented in the form of a histogram of the color of the object using the hue color component of the pixel object. The hue color is used to eliminate the luminance effects of the environment around the object. The influence of the background behind the object being tracked is reduced by making a pixel weighting distinction on the basis of the pixel's position in the ROI. The ROI is divided into several subareas. The subarea at the center of the ROI is given the maximum weight while that outside the object area is given the minimum weight.

The sampling technique is used to accelerate the process of forming histograms and calculating the direction of movement of the object being tracked. The systematic sampling method is chosen in this work because of its ease in implementation. The pixels in the area of the tracked object are chosen systematically at certain intervals in the horizontal and vertical directions. In other words, instead of all pixels, the pixel selected as a sample is used to represent the object being tracked. In this way, the number of pixels processed decreases to $\frac{1}{interval^2}$ from the original number of pixels in the ROI. Given this decrease, the number of iterations needed to form the histogram and determine the direction of movement of the object also decreases in the same ratio.

The position of the object after movement is estimated by finding the point with the highest image histogram similarities between the image histogram at the object's location in the previous frame and the image histogram at the same location in the current frame by using the Bhattacharya coefficient value [27]. A straight line is formed between the starting point of the object's position and the estimated position of the object in the direction of its movement. The line is extended with a certain value called tolerance. The evaluation of the highest similarity values is carried out at each point passed in this line. The point with the highest Bhattacharya coefficient is assumed to be the position of the object after the movement. The simulated annealing algorithm is used to find the coordinates of the intended point in the search range.

Object Representation. The object to be tracked on the video frame is in a square region called the ROI. The object representation used in this algorithm uses the image histogram. The image histogram is chosen because the calculation method for image histogram formation is simple. In addition, the image histogram does not store the spatial position of the pixel object and only stores pixel color information. Such feature is beneficial when the object being tracked rotates, as illustrated in Figure 2. The rotation of the object being tracked will not greatly affect the shape of the histogram because the histogram only considers the number of pixels with a color in the ROI region. The shape of the histogram is relatively fixed even though the position of the pixel with that color changes.

The number of calculations on histogram formation is reduced by performing class divisions on histograms at certain intervals; these divisions are referred to as histogram classes. The value of a pixel hue in the ROI is assigned to the interval class. In this study, the number of classes is adjusted on the basis of the number of pixels in the ROI area by using Sturge's rule. The hue value is described as an angular degree (1° to 360°) so that the value of the hue per pixel (hue_{pixel}) is in the interval class (bin_{hue}), as in Equation (1).

$$bin_{hue} = hue_{pixel} \frac{N_{class\ interval}}{360} \tag{1}$$

Where $N_{class\ interval}$ is the number of histogram classes, that is determined using Sturge's rule [28], as seen in Equation (2), where W is the width of the ROI (in pixel) and H is the height of the ROI (in pixel).

$$N_{class\ interval} = 1 + 3{,}322 \log WH \tag{2}$$

Pixel Weighting. The ROI can shift far enough from the center of the object on the image histogram formation because the spatial position of the pixel is not stored. In addition, the area on the outside of the ROI tends to be affected by objects other than the object being tracked, including the background of the object being tracked. This drawback is addressed by using the difference in the weights of the pixel values according to the position of the pixel area on the ROI. Area-based weighting is proposed as a method for assigning weights to each pixel in the ROI area on the basis of the pixel area toward the center of the ROI.

Area-based weighting carries out pixel's weighting according to the subarea position in the ROI. Each subarea can have its own predetermined weight. To differentiate between areas that are at the center of ROI and areas on the outside of ROI, we divide the ROI area into nine subareas with three different weights according to the region (Figure 3). The use of area-based weighting techniques reduces the weight of the pixels that are on the outside of the ROI. In Figure 3, the subarea of the ROI in the outside diagonal position is given a weight of B_3 while the subarea in the center of the ROI is given a weight of B_1, where $B_3 \leq B_2 \leq B_1$. α is the width of B_1, β is the height of B_1, W is the width of the ROI, and H is the height of the ROI. The weighting value in this study based on the subarea position in Figure 3 is formulated in Equation (3).

The "pixel weighting based on area" technique is used because its calculation and implementation are easy. It is also assumed to be sufficient to accommodate deficiencies from object representations using histograms.

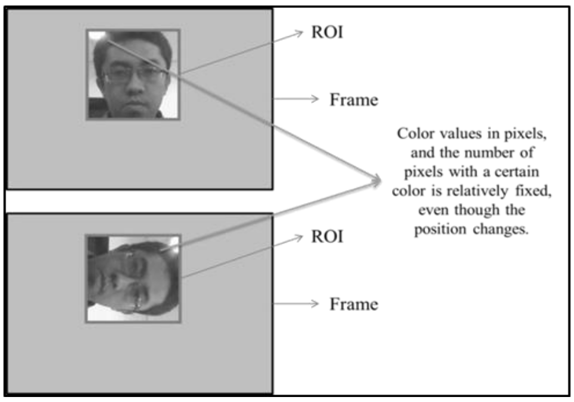

ROI
Frame
Color values in pixels, and the number of pixels with a certain color is relatively fixed, even though the position changes.
ROI
Frame

Figure 2. Effect of Rotation of Object on the Value of the Image Histogram

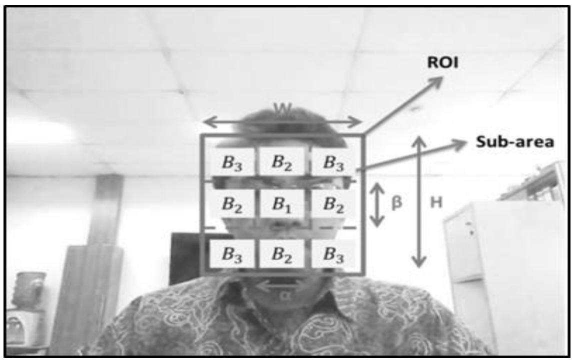


Figure 3. Difference in pixels' Weighting Values Based on Subarea

$$Weight = \begin{cases} B_1,\ if\ \left(x > \frac{W-\alpha}{2}\ and\ x < \frac{W+\alpha}{2}\right)\ and \\ \quad \left(y > \frac{H-\beta}{2}\ and\ y < \frac{H+\beta}{2}\right) \\ B_2,\ if\ \left(x > \frac{W-\alpha}{2}\ and\ x < \frac{W+\alpha}{2}\right)\ and \\ \quad \left(y < \frac{H-\beta}{2}\ and\ y > \frac{H+\beta}{2}\right) \\ B_2,\ if\ \left(x < \frac{W-\alpha}{2}\ and\ x > \frac{W+\alpha}{2}\right)\ and \\ \quad \left(y > \frac{H-\beta}{2}\ and\ y < \frac{H+\beta}{2}\right) \\ \quad B_3,\ else \end{cases} \tag{3}$$

The value of each histogram class must be normalized so that the total value of all histogram classes is 1. Normalization is performed by dividing the value of each histogram class by a constant, which is the total value of the entire histogram class. In normal histogram formation, the value of each histogram class is divided by the number of pixels in the ROI area. In the pixel weighting method, the value of each histogram class is divided by a constant, which is the total area of the entire subarea multiplied by the weight of each subarea. The values of these constants are defined in Equation (4).

$$C = \alpha\beta(B_1 - 2B_2 + B_3) + (\alpha H + \beta w)(B_2 - B_3) + WHB_3 \qquad \textbf{(4)}$$

with

$x = pixel\ coordinate\ in\ the\ x - axis$
$y = pixel\ coordinate\ in\ the\ y - axis$
$W = width\ of\ ROI\ (in\ pixel)$
$H = height\ of\ ROI\ (in\ pixel)$

In the formation of a weightless image histogram, the value of a particular histogram class is increased by 1 for each pixel with the color value in the histogram class interval. The value of a particular histogram class is increased by the weight of the pixel, as written in Equation (3).

Color Model. The use of RGB color models to represent pixel's colors is considered sensitive, especially when environment lighting changes around the object. Changes in luminance can result in significant changes in the RGB value of the pixel and alter the shape of the image histogram. Significant changes in the histogram value of the same object decrease the value of similarity using the Bhattacharya coefficient when the target and candidate histograms are compared.

The Hue Saturation Luminance or HSL color model is one of the frequently used color models. The HSL divides color components into hue, saturation, and luminance/lighting. By not considering the saturation and luminance values, the pixel's hue value is used to reduce the effect of lighting changes [29]. At the same color and normal lighting, the change in value of the RGB color components seems to be significant. In the HSL model, a noticeable change only occurs in the luminance value; the hue is relatively stable.

Pixel Sampling. Pixel sampling in the ROI is aimed at reducing the number of iterations in the formation of image histograms and calculating the direction of the movement of objects. In this study, pixel samples are taken systematically with the distance between i pixels in the horizontal and vertical directions. With this method, the number of pixels sampled on the ROI is (1⁄(i*i)) of the total number of pixels in the ROI. How the pixel sampling is systematically carried out on the ROI is shown in Figure 4.

As a result of the application of sampling to pixels, the value of the normalization coefficient in Equation (4) will decreases to $^1/_{i^2}$ from the initial value, resulting in Equation (5).

$$C_{sampling} = \frac{\left((B_1 - 2B_2 + B_3)\alpha\beta + (B_2 - B_3)(\alpha H + \beta W) + \right.}{i^2} \quad \textbf{(5)}$$

Overall, Algorithm 1 is the pseudocode that illustrates the process of forming image histograms from this object tracking algorithm.

```
Input   : image, ROI (size and current location), number of histogram classes,
Output  : Hue Array (Histogram),

1. Calculate the divider value by the formula:
   $p = \frac{360}{Number\ of\ Histogram\ Classes}$
2. Create an array to store the hue with a capacity of $p$,
3. Get the image in the ROI area and save it to the $BMP$ variable.
4. For each pixel exposed as a pixel sample at $BMP$, do:
      a. Calculate the pixel weighting value using Equation (3),
      b. Obtain a hue class index using Equation (1),
      c. Add the value of the hue array with an array index from step (b) equal to the weight value of step (a) divided by the value of Equation (5).
5. Return array hue as algorithm output.
```

Algorithm 1. Histogram Formation Algorithm

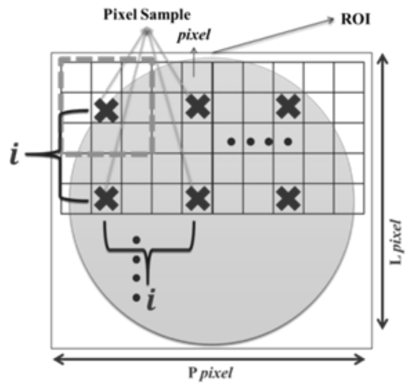


Figure 4. Systematic Pixel Sampling to Represent Objects on ROI

Determination of Object Movement Direction. The direction of object movement is determined by calculating the centroid coordinates using the image moment. The calculation is carried out by obtaining the interval class from the hue value of the affected pixel sample and then calculating the root of the interval class ratio between the target hue histogram and the candidate hue histogram as a multiplier of the sample pixel coordinates. Algorithm 2 determines the direction of movement of the ROI.

```
Input   : image, ROI size and current location, target and candidate histograms
Output  : coordinates of candidate ROI location
1. Perform image acquisition in the ROI area and save it to the $BMP$ variable.
2. For each pixel in coordinates $(x, y)$ at $BMP$ that is exposed as a pixel sample with $0 < x < W$ and $0 < y < H$, do:
      a. Obtain the hue class index using Equation (1) and save it to the $k$ variable,
      b. Calculate weight $\omega$ with,
         $\omega = \sqrt{\frac{H_{Target}[k]}{H_{Candidate}[k]}}$
      c. Change the value of:
$\_Moment1X = \_Moment1X + \omega x;$
$\_Moment1Y = \_Moment1Y + \omega y;$
$\_Moment00 = \_Moment00 + \omega$
3. Calculate the value of
   $D_x = \frac{Moment1X}{Moment00}$ and $D_y = \frac{Moment1Y}{Moment00}$
4. Change the value of current ROI coordinate to
   $ROI_{Kandidat}.X = ROI.X + D_x$ and $ROI_{Kandidat}.Y = ROI.Y + D_y$
5. Return the value as the algorithm output.
```

Algorithm 2. Determination of Object Movement Direction

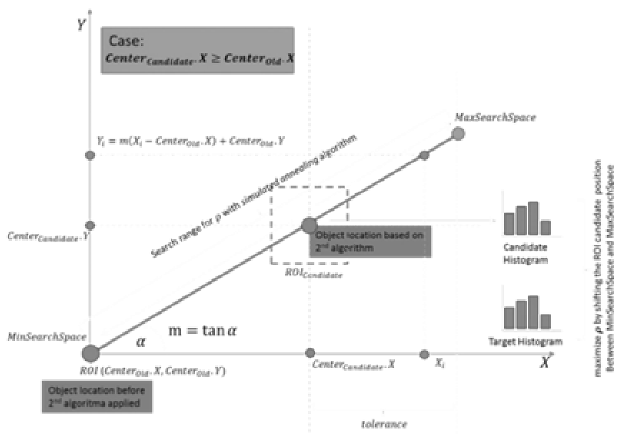

Y
Case:
$Center_{Candidate}.X \geq Center_{Old}.X$
$Y_i = m(X_i - Center_{Old}.X) + Center_{Old}.Y$
MaxSearchSpace
Search range for ρ with simulated annealing algorithm
$Center_{Candidate}.Y$
Object location based on 2nd algorithm
$ROI_{Candidate}$
Candidate Histogram
Target Histogram
MinSearchSpace
$m = \tan\alpha$
α
$ROI\ (Center_{Old}.X, Center_{Old}.Y)$
$Center_{Candidate}.X$
X_i
X
Object location before 2nd algoritma applied
tolerance
maximize ρ by shifting the ROI candidate position Between MinSearchSpace and MaxSearchSpace

Figure 5. Illustration of the Best Candidate Location Search

Determination of Optimum ROI Location. On the basis of the coordinates of the candidate ROI location in Algorithm 2, a straight line is obtained between the center of the previous ROI location and the candidate ROI (Figure 5). This line is extended as far as the tolerance value. The coordinates of the point with the highest similarity value between the target histogram and the candidate histogram are determined on the basis of the Bhattacharya coefficient value and using the simulated annealing algorithm.

The gradient value m is calculated on the basis of the coordinates of the ROI position and the coordinates of the candidate ROI position according to Algorithm 2. Equation (6) is used in the calculation.

$$m = \frac{Center_{Candidate}.Y - Center_{Old}.Y}{Center_{Candidate}.X - Center_{Old}.X} \qquad (6)$$

To obtain a set of value pairs (x,y), which will act as the candidate center coordinates of the ROI, we shift the value of *x* every one pixel between *MinSearchSpace* and *MaxSearchSpace*. The value of y can be obtained by Equation (7).

$$y_i = m(x_i - Center_{Old}.X) + Center_{Old}.Y \qquad (7)$$

This set of value pairs is then used as input for the simulated annealing algorithm by maximizing a function with the Bhattacharya coefficient value (ρ). The optimum location determination algorithm is presented in Algorithm 3.

```
Input   : image, ROI size and current location, target and candidate histograms, $Tolerance$
Output  : optimum coordinates of candidate ROI location
    1. Save the coordinates of the location of the ROI $(x, y)$ on the variable $Center_{old}$,.
    2. Calculate the coordinates of the candidate ROI location $(x, y)$ using
    3. Algorithm 2. Determination of Object Movement Direction
    4.  and save it to a variable $Center_{Candidate}$.
    5. Calculate the gradient of the straight line with Equation $(m)$ and that between the two points in step (1) and step (2) with Equation (6),
    6. If the value of $Center_{Candidate}.X \geq Center_{Old}.X$, then
$MaxSearchSpace = Center_{Candidate}.X + Tolerance$
$MinSearchSpace = Center_{Old}.X$
else,
$MaxSearchSpace = Center_{Old}.X$
$MinSearchSpace = Center_{Candidate}.X - Tolerance$
    7. For each $X$ as a point between $MaxSearchSpace$ and $MinSearchSpace$, calculate the value of $Y$ using Equation (7).
    8. Find the pairs of values $(x, y)$ from step (5) that give the largest Bhattacharya coefficient value using the simulated annealing algorithm.
    9. Return the pair of values $(x, y)$ as the algorithm output.
```

Algorithm 3. Algorithm for Determining the Best ROI Location

Design of Proposed Algorithm. The algorithm basically works in two stages, namely, determining the direction of movement of the object and then estimating the best location of the object (ROI) in that direction using simulated annealing. Algorithms 1 and 3 are used to change the steps of histogram formation and determine the direction of movement of objects in the mean-shift algorithm. Algorithm 1 is then used to form an image histogram, and Algorithm 3 is used to determine the direction of movement of the objects, as shown in Algorithm 4.

```
Input   : Video frame
Output  : ROI (location of ROI in each frame)
    1. Determine the pixel sampling method that will be used.
    2. Obtain the image of the current video frame ($Frame_{Old}$).
    3. Determine the area and location of the ROI (tracked object).
    4. Calculate the number of histogram interval classes using Equation (2).
    5. Create a target histogram with
    6. Algorithm 1. Histogram Formation Algorithm
    7. .
    8. Move to the next video frame ($Frame_{Current}$).
    9. Perform the following steps until the ROI location does not change or until a certain iteration is reached:
        a. Create a candidate histogram using
        b. Algorithm 1. Histogram Formation Algorithm
        c.  from the ROI location,
        d.  Determine the new location of the ROI
            with Algorithm 3,
        e.  Move the ROI to the location in step (b).
    10. $Frame_{Old} \leftarrow Frame_{Current}$ and return to
        step (5) until the last video frame.
```

Algorithm 4. Proposed algorithm for object tracking

4. Simulation and Evaluation

Simulation is aimed to evaluate the ability of the proposed algorithm to overcome certain conditions that can occur in the environment. A number of treatments to simulate changes in environmental conditions are carried out to ensure that the conditions do not significantly affect object tracking. The simulation and evaluation are carried out using a computer installed with Windows 7™, Intel® Core™ i7-3740QM processor @ 2.70 GHz CPU, 16 GB RAM, and a default camera with a resolution of 640 × 480 pixels for video capture. Lighting manipulation is conducted with Dell Webcam Central™ software by making changes to the brightness and contrast when the camera records objects. The parameters for simulation and evaluation are shown in Table 1.

The pixels in the video frame are treated as pixels with a depth of 24 bpp. Each algorithm is given 999 frames of images for processing. The addition of one frame serving as the first frame of the video is processed using the cascade classifier algorithm to obtain the ROI automatically. Visual Basic 2017™ is used to implement algorithms into a testing program. Several simulation results are shown in Figure 6.

Algorithm performance is evaluated by comparing the average execution time of the proposed algorithm against that of the mean shift. The execution time per frame is then tested using one-tail t-test statistics with two independent sample groups while assuming different population variances using Microsoft Excel 2010™.

Table 1. Parameters for Simulation and Evaluation

Parameter	Proposed Algorithm	Mean shift
Pixel Weighting	Area-based weighting, with the area parameters for the value of α rated as $^1/_3$ of the width of the ROI ($\alpha = {^W/_3}$) and with the value of β being $^1/_3$ of the height of the ROI ($\beta = {^H/_3}$). The values of B_1, B_2, and B_3 are 1, 0.75, and 0.5, respectively.	Epanechnikov Kernel
Color Model	HSL (only the hue is used)	RGB
Number of Histogram Classes	Adaptive using Sturge's rule according to the size of the ROI	20 Class
Pixel Sampling	Systematic with an interval of 3 pixels	-
Internal Loop	Optimized with simulated annealing	-

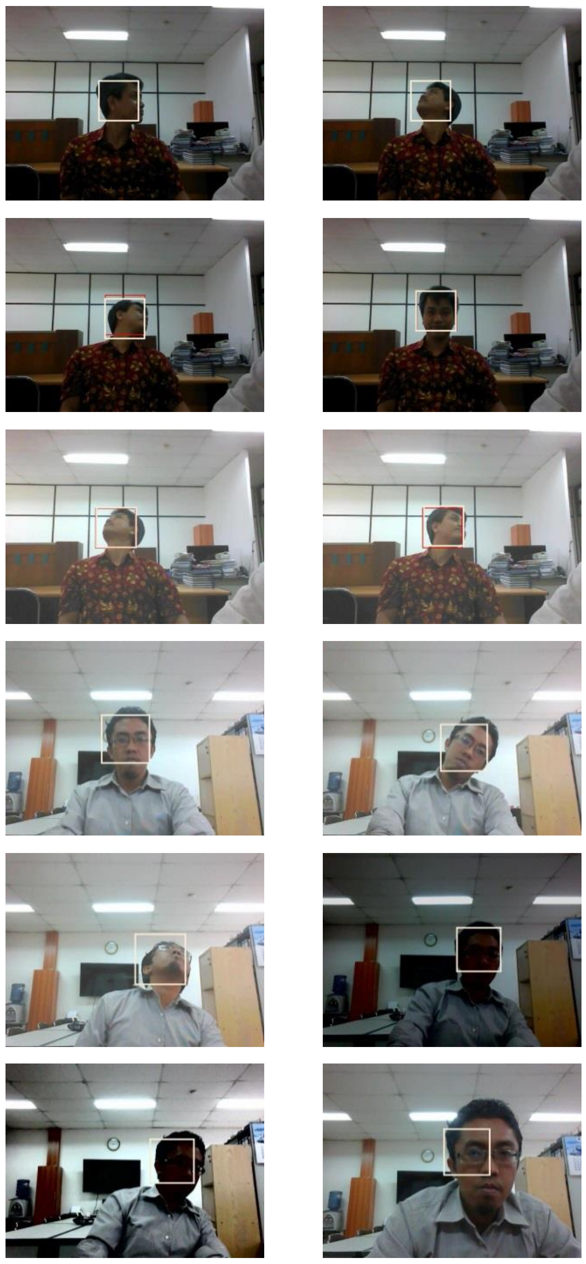
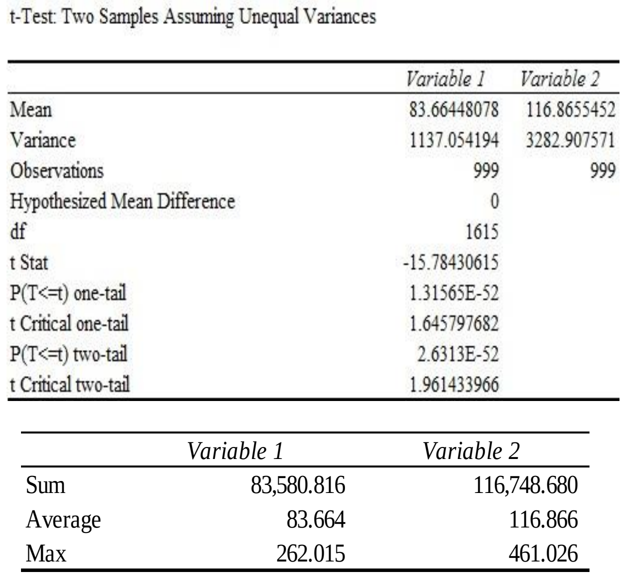


Figure 6. Algorithm Simulation with Rotational Conditions on Objects, Luminance, Contrast, and Size Changes

The initial testing hypothesis (H_0) states that no difference in average tracking speed exists between the proposed algorithm and the mean-shift algorithm. Figure 7 shows the results of the one-tail t-test conducted under the assumption of an unequal population variance. Here, two independent samples are used to compare the average processing times of the proposed algorithm and the mean-shift algorithm with a 95% confidence level. Variable 1 is the proposed algorithm, and Variable 2 is the mean-shift algorithm. Figure 7 shows that the value of $p_{\text{value-one tail}}$ (1.31565E-52) is smaller than the value of α (0.05). Hence, with a 95% confidence level, H_0 is



t-Test: Two Samples Assuming Unequal Variances

	Variable 1	*Variable 2*
Mean	83.66448078	116.8655452
Variance	1137.054194	3282.907571
Observations	999	999
Hypothesized Mean Difference	0	
df	1615	
t Stat	-15.78430615	
P(T<=t) one-tail	1.31565E-52	
t Critical one-tail	1.645797682	
P(T<=t) two-tail	2.6313E-52	
t Critical two-tail	1.961433966	

	Variable 1	*Variable 2*
Sum	83,580.816	116,748.680
Average	83.664	116.866
Max	262.015	461.026

Figure 7. Results of Statistical t-Test Test Between the Proposed Algorithm and the Mean-shift Algorithm

rejected; that is, a statistical difference in average processing time exists between the proposed algorithm and the mean-shift algorithm, with the former being faster than the latter. From the 999 picture frames given, the average tracking time of the proposed algorithm approaches 84-262 ms, whereas that of the mean-shift algorithm is 116-461 ms. Moreover, the proposed algorithm tends to have a relatively stable tracking time. The test results show that time variance value for the proposed algorithm is about 1 s and that for the mean-shift algorithm is 3 s.

5. Conclusion

This work proposes an algorithm with fast performance. The algorithm's resistance to changes in environmental conditions is determined in simulations. The proposed algorithm has an average processing time of around 83.66 ms per picture frame. The process time variance between frames is about 1 s. The p-value of the one-tail t-test is much smaller than the tolerance value of the test error (5%). This result proves that statistically, the proposed algorithm has a faster average processing time than the mean-shift algorithm. For future research, the proposed algorithm can be enhanced to solve three-dimensional tracking problems.

References

[1] K. Mutijarsa, Y. Bandung, L.B. Subekti, TELKOMNIKA 15/4 (2017) 1901.

[2] Y. Bandung, K. Mutijarsa, L.B. Subekti, Int. Symp. Electron. Smart Devices (2017) 6.

[3] G. Liu, S. Liu, K. Muhammad, S. Member, IEEE Access 6 (2018) 29283.

[4] Q. Liu, X. Lu, Z. He, C. Zhang, W. S. Chen, Knowl.-Based Syst. 134 (2017) 189.

[5] U.A. Agrawal, IEEE International Conference on Signal Processing, Computing and Control (ISPCC), 2017, p.187.

[6] G.S. Walia, S. Raza, A. Gupta, R. Asthana, K. Singh, Expert Syst. Appl. 78 (2017) 208.

[7] T. Mahalingam, M. Subramoniam, Appl. Comput. Informatics (2018) 1.

[8] G. Lee, R. Mallipeddi, M. Lee, Expert Syst. Appl. 80 (2017) 46.

[9] F. Hidayat, B.R. Trilaksono, H. Hindersyah, Int. J. Electr. Eng. Informatics 9/4 (2017) 632.

[10] C. Aynaud, C. Bernay-Angeletti, R. Aufrere, L. Lequievre, C. Debain, R. Chapuis, IEEE Robot. Autom. Mag. 24/3 (2017) 65.

[11] L. Yulianti, B.R. Trilaksono, A.S. Prihatmanto, W. Adiprawita, Int. J. Electr. Eng. Informatics 8/3 (2016) 676.

[12] H. Zhang, Y. Wang, L. Luo, X. Lu, M. Zhang, Neurocomputing 249 (2017) 253.

[13] X. Zhang, Y. Cui, D. Li, X. Liu, F. Zhang, Opt.-Int. J. Light Electron Opt. 123/20 (2012) 1891.

[14] J. Jeong, T. Sung, J. Bae, Expert Syst. Appl. 79 (2017) 194.

[15] I.A. Iswanto, B. Li, Procedia Comput. Sci. 116 (2107) 587.

[16] Y. Deng, F. Liu, J. Chen, G. Su, Optik (Stuttg). 125/16 (2014) 4572.

[17] J. Zhou, Z. Li, C. Fan, IET Image Process. 9/5 (2015) 389.

[18] A. Ardiansyah, 2018 Int. Conf. Appl. Eng. (2018) 1.

[19] A. Mahali, M. Izzuddin; Harjoko, J. Math. Nat. Sci. 24/2 (2014) 167.

[20] R. Qin, S. Member, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 8/5 (2015) 1974.

[21] Q. Li, S. Ural, J. Anderson, J. Shan, S. Member, IEEE Trans. Geosci. Remote Sens. 54/12 (2016) 7112.

[22] P. Kalaiselvi, S. Nithya, IOSR J. Comput. Eng. 14/3 (2013) 79.

[23] A. Shashua, T. Riklin-raviv, IEEE Trans. Pattern Anal. Mach. Intell. 23/2 (2001) 129.

[24] H. Wang, S.Z. Li, Y. Wang, (2004) 2.

[25] D. Comaniciu, V. Ramesh, Proc. IEEE Conf. Comput. Vis. Pattern Recognition CVPR 200, 2/7 (2000) 142.
[26] D. Comaniciu, V. Ramesh, IEEE Trans. Pattern Anal. Mach. Intell. 25/5 (2003) 564.
[27] P.A. Mar, J. Lorenzo-navarro, M. Castrill, 3.
[28] H.A. Sturges, J. Am. Stat. Assoc. 21/153 (1926) 65.
[29] Y. Zhang, X. Li, 2011 4th International Congress on Image and Signal Processing, 2011, p.974.