

State Machine Implementation for Human Object Tracking using Combination of MobileNet, KCF Tracker, and HOG Features

Fabiola Maria Teresa Retno Kinasih¹, Christ Freben Dommaris Saragih², Carmadi Machbub³,
Pranoto Hiday Rusmin⁴, Lenni Yulianti⁵, and Dian Andriana⁶

^{1,2,3,4,5} Bandung Institute of Technology, Indonesia

⁶Indonesian Institute of Sciences, Indonesia

¹fabiola.maria@students.itb.ac.id ²christ.saragih@students.itb.ac.id ⁶dian.andriana@lipi.go.id

³carmadi@lisk.ee.itb.ac.id ⁴lenni@lisk.ee.itb.ac.id ⁵pranoto@lisk.ee.itb.ac.id

Abstract: Since the Viola and Jones' method on real-time face detection was proposed in 2001, numerous works for object detection, person recognition, and object tracking have been published by papers and journals. Each method has its strong points and drawbacks. That means that in a system which only employs a standalone method, we could only get either speed or accuracy. In this paper, we proposed a state-machine method to combine face recognition, face detection, and tracker to harness the tracker promptness while maintaining the ability to distinguish the person of interest with the other person and backgrounds, to overcome the limitations of the standalone method. Subsequently, the information gathered from this image processing side will be delivered to the hardware tracker. The image processing side becomes a visual sensor that provides feedback or measurement value i.e. center point coordinate value from the detected face.

The 2 DOF hardware tracker camera platform being used implements Model Predictive Control to calculate required control action thus the platform is able to track the target object, keeping it at the center of the frame. MPC method is chosen because it produces an optimal control signal while considering the input signal saturation aspect. The MPC control signals deliver a good control pan and tilt system response with rise time < 1 second and overshoot <15%. It is also noticed that the FSM implemented in this paper is able to meet the goal with a considerable performance for indoor settings.

Keywords: computer vision, person recognition, state machine, tracker, visual servoing, MPC, pan-tilt camera.

1. Introduction

Object tracking has recently been an appealing topic regarding computer vision. With numerous publications in IEEE and almost half of them are quite recent (2014 and later) it is safe to say that object tracking is much discussed by academic society nowadays. Ever since the Viola and Jones' method on real-time face detection was proposed in 2001, numerous works for object detection, person recognition, and object tracking have been published by papers and journals. While each method has its own advantages, they possess their own limitations.

Such limitations could arise from the method's complexity which affects speed, accuracy limitation due to different information interpreted from pictures which actually contain the same objects but from a different point of view, or simply the limitation of method's ability. For example, although there is a fast method for Face Alignment as explained in [1], Face Recognition method has a considerably lower speed than other computer vision methods because there are three other steps other than Face Alignment in Face Recognition, thus takes longer computation time. Accuracy limitation in tracking a person using face detection often happens when the object turns over so that the face landmarks no longer appear in the frame. On the other hand, the tracking method does not have a way to determine which object to be tracked, it needs information from the previous frame about the target whereabouts.

Received: September 20th, 2019. Accepted: December 26th, 2019

DOI: 10.15676/ijeel.2019.11.4.5

Those limitations could affect a system in such ways:

- an accurate object tracking system with very slow performance on budget hardware
- a fast (high fps) object tracking system that misses the object target every once in a while
- a considerably fast face-detection based object tracking system that misses the object target every time the person turns around

Before continuing on how to overcome all those limitations, we need to consider some of the state-of-the-art methods in computer vision method. Especially in pure tracking method, there are dozens of trackers method as reviewed in some papers such as “Online Object Tracking: A Benchmark” and “The Visual Object Tracking VOT2017 challenge results” there are several state-of-the-art methods namely Siam FC, CSR-DCF, and MD-Net. Unfortunately, although those methods provide better accuracy than the 2014 state-of-the-art method KCF, their performance speeds are far lower than KCF. On the other hand, although YOLO v3 is appreciated by many as the finest method, and also the most popular, its computation is quite expensive. While YOLO-v3 could run with 30fps speed on a Pascal TitanX GPU, its speed is considerably slower (only up to 6 fps) on GTX 1050 Ti GPU. This leads us to find a faster method that could run on cheaper hardware. After some extensive search, we decide that Mobilenet-SSD architecture, with thinner convolutional layer (only 2 convolutional layers with 1024 thickness, compared to 3 of those used in YOLO) and no fully-connected layer as can be found in YOLO, is a better option to be run on cheaper device, with up to 12 fps speed on GTX 1050 Ti GPU. Those deliberately chosen methods, along with a face recognition method, will be three important parts in the computer vision method being used in this research.

This paper is a part of Object Tracking research in Control and Computer System Research Group, Institut Teknologi Bandung. Previous works in face detection and tracking[2], custom object tracker[3], and a combination of face and posture to track human movement[4]. The goal of this research is to track a specific person in interest. Regarding this goal, in this paper, we proposed a state-machine method to combine face recognition, face detection, and tracker to harness the tracker promptness while maintaining the ability to distinguish the person of interest with the other person and backgrounds, to overcome the limitations of standalone method previously mentioned in paragraph 2 and 3 in this section. While state machine has been implemented in various system and scheme such as for Blind Multiuser Channel Estimation in communication system [5], for Sintering Burn-Through Point in control system[6] and Artificial Emotion in computer vision [7], there is no previous work that applies state machine for object tracking purposes. This paper intends to propose the idea of applying the state machine concept in computer vision to improve object tracking system capabilities.

This paper would discuss all the computer vision methods being used in separated sections, the state-machine method, and also the hardware set-up and control method being used in the hardware tracker. The detailed process about how the open-sourced face recognition library works will be discussed in section 2: Face Recognition. The face detector that is being used based on Mobilenet-SSD architecture would be briefly discussed in section 3: Object Detection. A bit detailed mathematical reason why KCF is chosen will be discussed in section 4: KCF Tracker. The state machine proposed will be presented in section 5: Algorithm Switching using State Machine Concept. The hardware tracking part will be discussed in section 6: Camera Hardware Model and its set up in section 7: System Implementation. The results from the image processing side and from the whole tracker (both hardware and software) would be discussed in section 8: Results and Discussion.

2. Image Processing – Face Recognition

There are four main steps in doing face recognition: finding faces, posing and projecting face landmarks, encoding faces, and then find the person name/ID from the encodings database [8].

A. Finding Faces

Features that are being used to find face contours in this project are Histogram of Oriented Gradients[9]. There are three steps to find the HOG of a colored image:

- Convert the image to grayscale.
- Iterating for every pixel, compare the pixel (grayscale) value with its neighbor.
- Draw an arrow with its head pointing to the darkest neighbor

The result from doing those three steps to one of the dataset images is displayed in Figure 1.

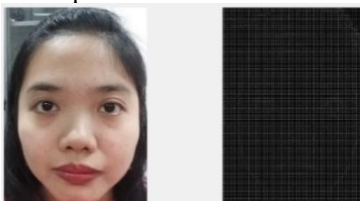


Figure 1. Original colored image (left), HOG Features (right)

Since the HOG patterns from faces are more or less similar, with knowing the common face HOG pattern [8,10] we could get the face area to continue to the next step

B. Posing and Projecting Face Landmarks

After the face area is found, the next step is to find the face landmarks. The face landmarks are important because if the computer compares raw pixel between two images of the same person with a different angle of view, it would be completely different. However, if the face landmarks are known, several transformations could be done to center those two images so that the comparison might be done successfully.

Face Landmark Estimations that is implemented in the face recognition library is based on Kazemi and Sullivan's works on Face Alignment using Regression Tree [1]. The algorithm will find the 68 landmark points on the face on eyebrows, eyes, nose, lips, and chin as shown below.



Figure 2. Sixty-eight Landmark points with lines drawn to connect points in one area

After the 68 landmarks point obtained, an affine transformation is applied to centering the image. The affine transformation includes translating, scaling, rotating, and shearing. All parallel lines will stay parallel after this transformation, thus the unique face landmarks of each face are preserved.

C. Encoding Faces

To distinguish the face between persons, the face recognition library applies Open Face face embedding[11] based on the triplet mining method [12]. A deep neural network will take the 68 landmark points to calculate 128 measurements output. In a single step of the triple mining method, such a neural net is trained with 2 faces of the same person and 1 different face. Subsequently, the neural net tweaked slightly to produce closer measurements for the 2 faces of the person than the measurements for the different person[8].

D. Finding Name/ID of the Person Face

While implementing the face recognition library, the people database should contain at least one photo of each person in the list saved with the person's name or ID. Then the program will find the 68 face landmarks, before measuring the 128 measurements stored as this person unique face encoding in the database.

Upon the program running, for each frame, the face area is searched using HOG Features, and for each face area, the landmarks will be obtained followed by the 128 measurements generated. The new 128 measurements from each face found in a frame would be compared with the known face encoding stored in the database. If the distance of the measurements is far (above the threshold, in this project set to 0.5) from all the known face encoding, the face would be marked as unknown. In other cases, the closest distance between the new 128 measurements and known face encoding will be chosen and then the face would be marked as the corresponding person in the database.

From the explanation of step C: Encoding Face and step D: Finding Name/ID of the Person Face it could be seen that one of the benefits of this face recognition method is its ability to add the database of known person without re-train the whole system (as in the case of EigenFaces and some other methods). The deep neural network training that is done for step C using extensive face dataset aimed for finding a neural net that is able to generate measurements of a face, so that any two measurements of different picture/pose of the same person will have closer values than measurements of two different people. This measuring system then could be applied to any new face picture, and the measurement results are stored in the database. While the system running, we could compare the measurement(s) taken from the face(s) found in the current frame with the measurements of faces in the database. The measuring method (steps A, B, and C) being used while recognizing is the same as the one being used in creating the database.

3. Image Processing – Face Detection

The object detection implemented is a derivation work from Tensorflow Object Detection API[13]. This face detector classifier is based on MobileNet SSD architecture.

A. MobileNet

MobileNet architecture is chosen because it has lower complexity compared to other convolutional neural networks. The main difference between Mobilenet and standard CNN is in the way it does the convolution, called depthwise separable convolution. In depthwise separable convolution, a standard convolution that takes $D_F \times D_F \times C_i$ sized input feature map F (D_F is the size of features spatial width and height while C_i denotes the input depth, as a number of channels) and produces $D_F \times D_F \times C_o$ (C_o denotes output depth) sized output feature map G using a $D_K \times D_K \times C_i \times C_o$ sized kernel K would have a formulation as below [14]

$$G_{k,l,c_o} = \sum_{i,j,m} K_{i,j,c_i,c_o} \cdot F_{(k+i-1),(l+j-1),c_i} \quad (1)$$

Thus the complexity of the standard convolution is:

$$D_K \cdot D_K \cdot C_i \cdot C_o \cdot D_F \cdot D_F \quad (2)$$

While the Depthwise Separable Convolution has two separated calculation steps:

- Depthwise Convolution: For every channel, doing convolution only in 1 spatial channel, as if doing 2D convolution. After doing this step, we will have as much as C_i results of spatial convolution.
- Pointwise Convolution: For each point in spatial convolution, doing pointwise (1x1) convolution with all the input channels. The channel dimension of the output will be changed to C_o as designed.

For the Depthwise Convolution, the complexity would be similar with the standard convolution one, but because it is done as much as the number of input channel C_i and the 2D kernel size would only be $D_K \cdot D_K$ sized, we could remove the C_o component from equation (2). Thus the Depthwise Convolution complexity becomes:

$$C_i \cdot D_K \cdot D_K \cdot D_F \cdot D_F \quad (3)$$

Likewise, because Pointwise Convolution’s kernel would be 1×1 sized, we could remove the kernel size components from the standard convolution’s complexity as explained in equation (2). Thus the Pointwise Convolution complexity becomes:

$$C_i \cdot C_o \cdot D_F \cdot D_F \tag{4}$$

Combined together, the total complexity of those two steps (Depthwise Convolution and Pointwise Convolution), thus the complexity of Depthwise Separable Convolution would become:

$$C_i \cdot D_K \cdot D_K \cdot D_F \cdot D_F + C_i \cdot C_o \cdot D_F \cdot D_F \tag{5}$$

Comparing the complexity of Depthwise Separable Convolution and Standard Convolution, we would obtain a complexity reduction for:

$$\frac{C_i \cdot D_K \cdot D_K \cdot D_F \cdot D_F + C_i \cdot C_o \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot C_i \cdot C_o \cdot D_F \cdot D_F} = \frac{1}{C_o} + \frac{1}{D_K^2} \tag{6}$$

It claimed that the Mobilenet architecture employing Depthwise Separable Convolution would perform 8 to 9 times faster than other CNN employing Standard Convolution[14].

B. Single Shot Multibox Detector

Single Shot Multibox Detector is a region proposal detector based on a feed-forward convolutional network[15]. The output of the SSD network is bounding boxes for all object class instances paired with the confidence scores of each object detected in a frame.

The dataset that is being used as the input to train the SSD network needs to contain ground truth information of the known object presence. Choosing which set of default boxes that appertain to the ground truth is a part of the SSD matching strategy. (The mathematical details for training an SSD network is beyond the scope of this paper.) An example of default boxes feature maps for SSD is shown as below.

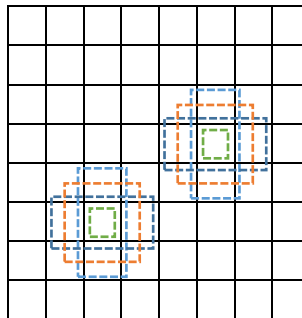


Figure 3. An example of default boxes in 8x8 feature maps for two objects in one frame

Considering the multi-scale feature maps that being used by SSD, the convolutional layers applied after base network layer, also known as Extra Features Layers would progressively decrease in spatial sizes, thus allow multiple scales predictions of detections [15]. For Mobilenet-SSD architecture, the base network layer used to extract features is Mobilenet. The architecture of the general SSD network is shown below.

The face detector is trained using WIDERFACE dataset [16]. Because of the large number of datasets and the broad range of variations in the dataset, this face detector performs incredibly robust within the various angle of view and is able to correctly detect faces from afar. This is the reason why the face detector is used as a ‘backup’ for face recognition, besides the speed. Detailed switching between face recognition and face detection will be discussed further in section 5. As discussed in the Introduction, the Mobilenet-SSD architecture is chosen because it has faster performance compared to the state-of-the-art method Yolo while being run on GTX 1050Ti GPU.

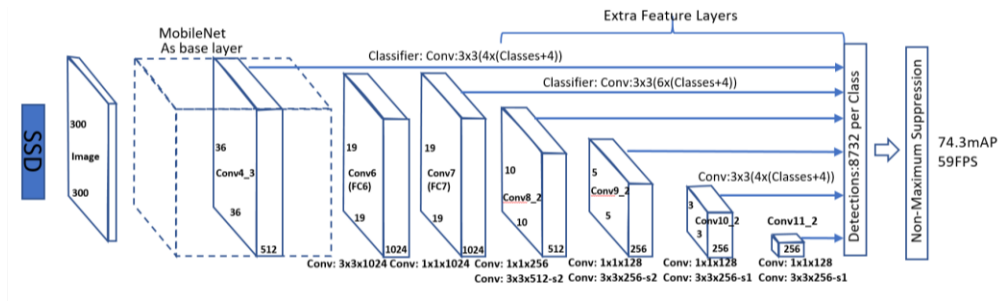


Figure 4. The general architecture of SSD Network. Notice that in this face detector, Mobilenet instead of VGG-16 is being used as the base layer [16]

4. Image Processing – Kernelized Correlation Tracker

The object tracker being used in this paper is the Kernelized Correlation Filter by Joao Henriquez [17]. Generally, correlation is commonly used in image processing or computer vision domain as a way to find similarities between the known object and the image patch that is being tested. Kernelized Correlation Filter is chosen to track the object of interest once it detected/recognized because of the efficient computation while considering all possibilities for image translation.

The preprocessing part in KCF has 4 steps:

- Hanning Window filtering form the initial bounding box
- Padding the region of interests
- Gaussian Response calculation
- Fourier Transformation to make spatial convolution more efficient

The detection part of the KCF algorithm has 5 steps:

- Get Principal Component Analysis descriptors
- Compress the features and the Kernel-Based Regularized Least Square model
- Calculate the Gaussian kernel
- Transform kernel into the frequency domain using FFT
- Filter response calculation in the frequency domain

If the maximum response obtained could exceed the threshold, the new bounding box will be produced. Else, the tracking failed and the tracker will raise a flag to notify the main program. Afterward, the new image patch obtained from the evaluated frame bounded by the new bounding box will be learned, so the tracker knowledge is updated in every iteration / every frame checked. The learning process itself has 5 steps similar to the detection part, except the last step is not to calculate filter response but updating the RLS model. The reason is that instead of finding the new bounding box, the goal of the learning process is to update the Regularized Least Square model.

Those full algorithms are implemented and available through the GitHub page and OpenCV [18] library. Several interesting building blocks that contribute to KCF Tracker's strong points will be discussed through these sub-sections.

A. Exploiting Circulant Matrices Properties

To make sure the tracker could spot the object being tracked even when there is a translational difference between the image patch under observation and training data, the KCF tracker is implicitly trained with all the possible translation [17]. To avoid explicit iteration of every translation possibility, cyclic shifts, both vertical and horizontal is being used. Then a matrix is built to contain the base sample and all the possible cyclic shifts of the sample called circulant matrix.

To have a closer look at what happened with that matrix and its properties, let discuss all the cyclic shifts for a 1-dimensional vector to build the matrix. The first row of the matrix will contain the base sample, and every subsequent row will contain the sample but shifted by one element from the prior row.

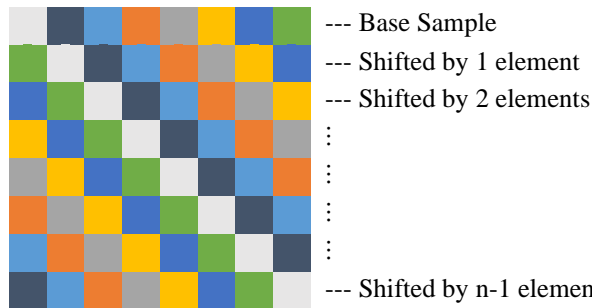


Figure 5. Circulant Matrix illustration for one-dimensional vector

A general mathematical representation of such matrix C , with the base sample c , represented as a one-dimensional vector $[c_1 \ c_2 \ c_3 \ \dots \ c_{n-1} \ c_n]$, could be formulated as follows:

$$C = \begin{bmatrix} c_0 & c_1 & c_2 & \dots & c_{n-2} & c_{n-1} \\ c_{n-1} & c_0 & c_1 & c_2 & \dots & c_{n-2} \\ c_{n-2} & c_{n-1} & c_0 & c_1 & \dots & c_{n-3} \\ c_{n-3} & c_{n-2} & \dots & \dots & c_{n-5} & c_{n-4} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ c_1 & c_2 & \dots & \dots & c_{n-1} & c_0 \end{bmatrix} \quad (7)$$

Circulant matrix has special properties that its eigenvectors, for all cases of circulant matrices, is the root of unity[19]. One of the most obvious eigenvectors that could be seen from the circulant matrix is a 1D vector with n elements, all 1. On the other hand, the eigenvalues will vary as the circulant matrices component varies. If a matrix containing all the eigenvectors of the circulant matrix, which is the root of unity, is built, it would end up as a Discrete Fourier Transform matrix with size $n \times n$. The definitive presence of eigenvectors makes it possible to diagonalize any circulant matrices, thus reducing the computational cost while maintaining the ability to spot the object of interest regardless of the translation.

B. Non-Linear Regression

The purpose of regression training in KCF is to obtain a solution w expressed as a linear combination of the samples[17] obtained from image patches. If the image patch being evaluated denoted by z , then the solution function could be expressed as:

$$f(\mathbf{z}) = \mathbf{w}^T \mathbf{z} \quad (8)$$

Since the non-linear regression is being used, the inputs would be mapped to non-linear feature space $\varphi(x_i)$ using kernel trick. The expression of the solution as a linear combination of the non-linear feature space is:

$$\mathbf{w} = \sum_i \alpha_i \varphi(x_i) \quad (9)$$

Now the goal of the training is to find the optimum alpha value. The tie-in between the alpha value and the kernel being used is expressed in the kernelized version of Ridge Regression solution [20]:

$$\alpha = (K + \lambda I)^{-1} y \quad (10)$$

Several kernels, including Radial Basis Function kernels family, would also be circulant when the data sample is circulant, and the proof is available on [17]. Thus the previous equation could be diagonalized in the frequency domain, considering only the first row of kernel matrix, yielding:

$$\hat{\alpha} = \frac{\hat{y}}{k_i^{xx} + \lambda} \quad (11)$$

C. Kernel Implementation

The kernelized correlation is then applied to image patches, and each sample patch translation is considered implicitly using circulant matrix structures. Correlation kernel between dataset sample and evaluated image patches expressed as:

$$K^Z = \kappa(P^{i-1}z, P^{j-1}x) = C(\mathbf{k}^{xz}) \tag{12}$$

This kernel matrix has circulant properties for kernel function κ Gaussian. Considering equation (8) and (9), the solution calculation for the evaluated image patch z is:

$$\mathbf{f}(z) = (K^z)^T \alpha \tag{13}$$

Exploiting the circulant properties of the kernel matrix K^Z , the computational complexity could be reduced by transforming the equation to the frequency domain via DFT so that the kernel could be diagonalized, thus only the diagonal elements of the kernel denoted by \mathbf{k}^{xz} are taken into account

$$\hat{\mathbf{f}}(z) = \widehat{\mathbf{k}^{xz}} \cdot \alpha \tag{14}$$

More detailed derivation and proofs of circulant properties of the kernel matrix, also the explanation about correlating with Gaussian kernel are available on [17]. The circulant matrices properties exploited in KCF give a significant contribution to KCF’s speed, which is the reason why the KCF method is chosen rather than the state-of-the-art method such as CSR-DCF, Siamese Network, or MDNet.

5. Image Processing Algorithm Switching Using State Machine Concept

On preliminary works mentioned in Introduction [4], there are two methods that are combined and switched when the condition is satisfied. To generalize the switching process, this paper aimed to design a state machine for method switching. The finite state machine that is being used in this project is a Moore machine because the state changes should only be affected by the result of each state. In each state, the program will run a different algorithm discussed in section 2-4. When the output result of one state yielded, a specific trigger would be raised, thus the program will know when the program should move to which states. All the state machine are implemented in Python 3.6 using PyTransitions module [21]

There are two state machines that have been evaluated, the first one has only two states, detecting state and tracking state, while the second one has three states, detecting, tracking, and recognizing. The first proposed state machine described below:

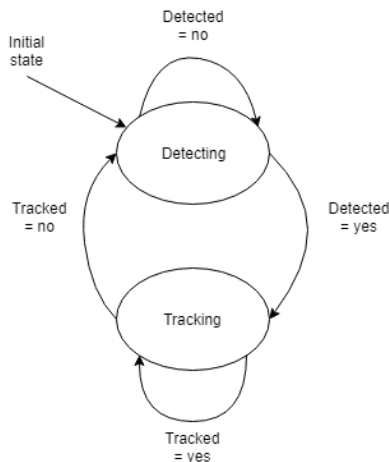


Figure 6. Two-State FSM proposed

In the two-state configuration, while in detecting state, the program will run the face recognition[8] program as discussed in Section 2. The face recognition will detect if there is human captured in the frame and recognize whether this human matched with the known person database. If the person’s name matched with the person being searched, the system will move to

tracking state. Tracking state using KCF tracker[17] works faster than detecting state, so the program will continue doing tracking as long as the tracker is able to track the person. If the tracker failed and the person is no longer being tracked, the system will move to detecting state. The system will evaluate the frame and find the person object in the evaluated frame. If a person found, then the person would be matched with the stored database, and if the person's name matched with the person being searched, the system will move to the tracking state again. But if the detection fails to find the person of interest, the system will stay in detecting mode and will re-detect for every new frame to find the person of interest.

Things are a bit different in three-state configuration. Now, the tracking and recognizing state are separated into two distinct states. While in the recognizing state, the program will run the face recognition. The detecting state no longer runs face recognition but instead running face detection[16] as discussed in Section 3. In the three-state configuration, the faster and wider-range face detection algorithm will work to back up while the recognizing state failed. In other words, when the searched person is not found or not recognized yet, the system will follow any person detected in the frame and will re-recognized the face if the confidence value is greater than a threshold value.

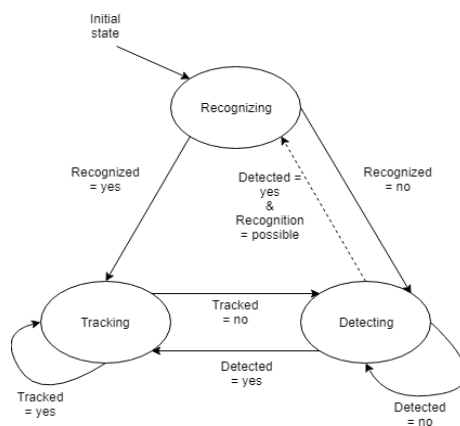


Figure 7. Three-State FSM proposed

6. Camera Hardware Model

A. Camera Projection Model

The camera projection model as explained in [22] define the camera coordinate system with x and y axes are the basis for the image plane. The z-axis in this coordinate system goes along the optical axis, which is perpendicular to the image plane. Focal length f is defined as the distance between the image plane and the origin point behind it. Such a camera coordinate system is explained in Figure 1 below.

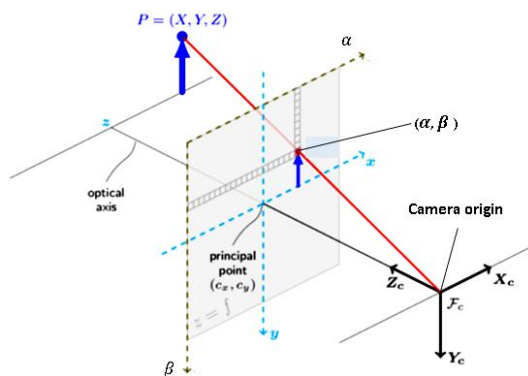


Figure 8. The coordinate frame in the lens/camera system

Furthermore, a real-world point ${}^cP = [x, y, z]^T$ expressed as a relative to the frame with the camera coordinate c , is projected onto the image plane as point $p = [\alpha, \beta]^T$ according to the following equation:

$$\pi(x, y, z) = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{f}{z} \begin{bmatrix} x \\ y \end{bmatrix} \quad (24)$$

B. Image Jacobian

From [23] the image Jacobian mathematical model and Jacobians interaction matrices are obtained, and the equation for the Jacobian interaction matrix can be expressed as:

$$\begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} \frac{f}{z} & 0 & -\frac{\alpha}{z} & -\frac{\alpha\beta}{f} & \frac{f^2+\alpha^2}{f} & -\beta \\ 0 & \frac{f}{z} & -\frac{\beta}{z} & \frac{-f^2-\beta^2}{f} & \frac{\alpha\beta}{f} & \alpha \end{bmatrix} \begin{bmatrix} T_x \\ T_y \\ T_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (25)$$

For this project, the DOF of the pan-tilt camera platform is only two instead of three, thus the state-space representation can be reduced by eliminating the z-axis movements (denoted by subscript z). And since the camera position is fixed, the translational component (denoted by T) can also be neglected, allowing the state-space representation to be furthermore reduced into second order. The remaining component is $\omega = [\omega_x \ \omega_y]^T$, where ω_y represents the pan angular velocity, ω_x represents the tilt angular velocity, and $s = [u \ v]^T$ is the actual pixel position in the camera frame.

$$\begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\frac{\alpha\beta}{f} & \frac{f^2+\alpha^2}{f} \\ -\frac{f^2-\beta^2}{f} & \frac{\alpha\beta}{f} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix} \quad (26)$$

Besides reducing the order, the equations could be simplified by considering the fact that pixel position value will always reach zero value thus parameter pixel $u \ v$ could be neglected. Therefore the Jacobian matrix for 2DOF pan-tilt camera platform could be expressed as follows

$$\begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} 0 & f \\ -f & 0 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix} = \begin{bmatrix} f & 0 \\ 0 & -f \end{bmatrix} \begin{bmatrix} \omega_y \\ \omega_x \end{bmatrix} \quad (27)$$

For pan-tilt camera time implemented in this paper, could be represented in a state-space form with state equations as follows

$$\begin{aligned} \dot{x}_{(t)} &= Ax_{(t)} + Bu_{(t)} \\ \dot{x}_{(t)} &= \begin{bmatrix} \dot{\alpha}_{(t)} \\ \dot{\beta}_{(t)} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha_{(t)} \\ \beta_{(t)} \end{bmatrix} + \begin{bmatrix} f & 0 \\ 0 & -f \end{bmatrix} \begin{bmatrix} \omega_{pan}(t) \\ \omega_{tilt}(t) \end{bmatrix} \end{aligned} \quad (28)$$

And the output equation as follows

$$\begin{aligned} y_{(t)} &= Cx_{(t)} \\ y_{(t)} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_{(t)} \\ \beta_{(t)} \end{bmatrix} \end{aligned} \quad (29)$$

The system is furthermore discretized using bilinear transformation, thus the state space representation in discrete-time becomes

$$x_{(k+1)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_{(k)} \\ \beta_{(k)} \end{bmatrix} + \begin{bmatrix} f \cdot T_s & 0 \\ 0 & -f \cdot T_s \end{bmatrix} \begin{bmatrix} \omega_{pan}(k) \\ \omega_{tilt}(k) \end{bmatrix} \quad (30)$$

The output, which is the pixel position of the face detected by the image processing algorithm, could be presented as follows

$$y_{(k)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_{(k)} \\ \beta_{(k)} \end{bmatrix} \quad (31)$$

7. System Implementation

A. System Configuration

As a whole, the system diagram block could be shown as follows. In this work, the camera being used is Logitech C920, pan-tilt actuators build from two motor servo Hitec HS-5645MG,

and a microcontroller Arduino Mega2560 to implement control algorithm. The image processing system which acts as a sensor is implemented in a personal computer powered by the Intel Core i5 processor, 16GB of RAM.

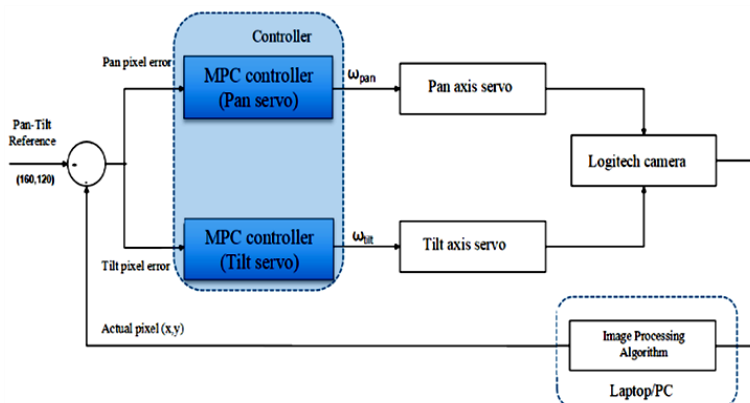


Figure 9a. Overall system configuration



Figure 9b. An actual pan-tilt camera system




Extracted frames from the camera will then be processed using the algorithm mentioned in section 2-5. The coordinate location from the target object will be sent to the microcontroller using serial communication. With the center point of camera frame ($u=160$ pixels, $v=120$ pixels) as the reference for pan-tilt hardware tracking, the coordinate location will be further processed using the MPC algorithm. The optimal control signal from MPC calculation will determine the actions given (moving servos in this case) in order to track the target object such that the target object will always be near/on the center of the camera frame.

B. Model Predictive Control

Model Predictive Control method is implemented for its ability to handle nonlinearity, in this case servo saturation, and its optimality. The MPC controller being used has the parameter weight for error $q = 0.1$, weight for control signal $r = 10000$, numbers of prediction horizon $N = N_u = N_y = 9$, the number of iteration for QP = 7, input saturation $u_{min} = -1$ radian/second and $u_{max} = 1$ radian/second. Those parameters were obtained via the tuning process in a system with camera focal length $f=680$ and control algorithm sampling time $T_s = 40$ ms. In an additional note, the ratio between q and r parameters will determine the system performance. A detailed explanation about MPC design for this pan-tilt camera platform is available on [24] while the design process is based on [25,26].


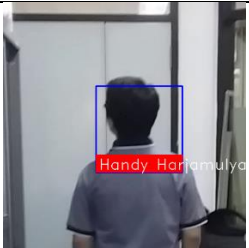
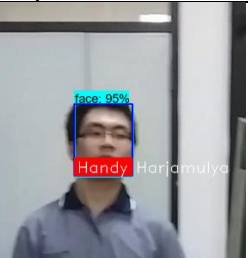
8. Result and Discussion

Table 1. Two State Implementation Result

State	Firstly the system will do recognition, then it will move to the tracking mode.	The system is able to track the person of interest when not all of the facial features present.	The system is able to track even when the person turns around.
Results			

Subsequently, the system with three state FSM is also tested to find another lab member named Handy Harjamulya. The results from the three-state FSM implementation is presented below:

Table 2. Three State Implementation Results

State	Firstly the system will do recognition, then it will move to the tracking mode.	The system is able to track when the person turns around.	When the tracker failed, the system will move to detection mode. If the confidence of detection is greater than 90%, the system will re-recognize the person.
Results			

To test the system, we use eleven videos contain some of the lab members and other people with the duration range from 7 seconds to 1 minute, and some live testing. To recognize a specific person, a database of known people contains 9 face/person from lab members is created. Firstly the system with two-state implementation is tested to find a lab member named Christ Saragih (the co-author). The results from the two-state FSM implementation is presented in Table 1.

Table 3. The system is able to find the correct person searched

The system tries to find lab member 'Christ Saragih'	The system tries to find lab member 'Handy Harjamulya'
	

When the person of interest is not present in the frame, the system will try to re-detect. The message "Detect failure, will re-detected" will be appeared to notify the user about the current state. Another case is when there are several people in a frame. The system is able to correctly find the correct person

The three-state implementation has the benefit that it can track the object located about 5 meters from the camera. This is possible because the face detection algorithm (with MobileNet-SSD architecture) has the ability to function properly at farther distance compared to face recognition mode. However, when the system is run for a small room, the two-state implementation has the advantages of running faster and smoother as there is less state-switching in operation.

Although the performance varies throughout hardware differences, usually the KCF tracker runs fastest, followed by the face detector, and face recognition takes the longest processing time. The speed of the face recognition mode is lower than 10 fps, while the speed of face recognition mode is between 10-20 fps, and the KCF tracker top performance in our system could reach up to 50 fps. Nevertheless, the system is fairly successful to achieve the goal to track the person of interest at a reasonable speed to track normal walking motion in indoor settings. With the variation of outdoor lighting scenes, the face recognition module has a high (>50%) false-negative rate, thus the system becomes less effective while working in outdoor settings.

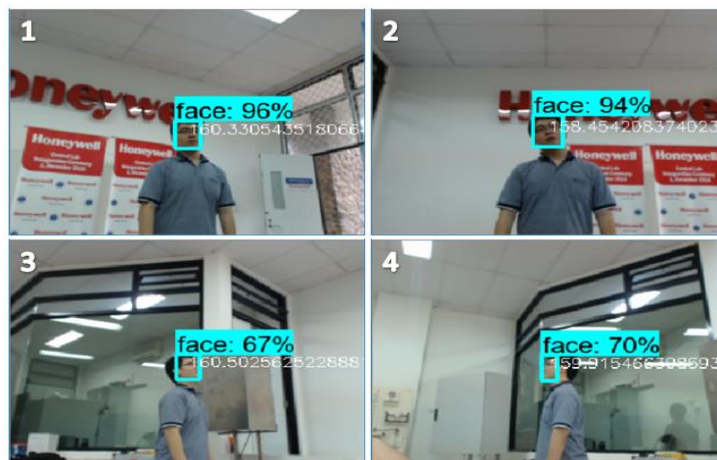


Figure 10. Initial position (1), movement position (2,3), and final position tracked moving object (4)

From the result above, we can state that the proposed system has the ability to identify people while running on a considerably faster speed than face recognition as a standalone method. For indoor settings, with affordable GPU (the entry-level GTX 1050 Ti) and affordable web camera, and for the walking scenario without abrupt movement, it is possible to achieve speed 30-50 fps while the system runs in the tracking mode. It is important to note that when the tracker failed, it would just take few frames to re-identify the person (few frames when the system runs the face-recognition mode with speed around 6 fps), before back to the tracking mode which is considerably faster. That means that we could achieve better accuracy (than standalone tracker) without significantly giving up the speed of the object tracking system.

After checking the performance of the image processing side, some experiments are conducted to observe the performance of a pan-tilt camera system hardware to a moving target. The target is a human face, a target will walk around in a room, and then the pan-tilt system will track the moving target. The goal of this test is for the pan-tilt system to track the moving target. Pan and tilt movements are shown in Figure 10.

While running the hardware tracker, we also log the Pan and Tilt movements to observe the response of MPC control that applied to both pan and tilt control loop separately. The parameter

of MPC control applied is discussed in section 7. Pan and tilt movements are shown in Figure 11.

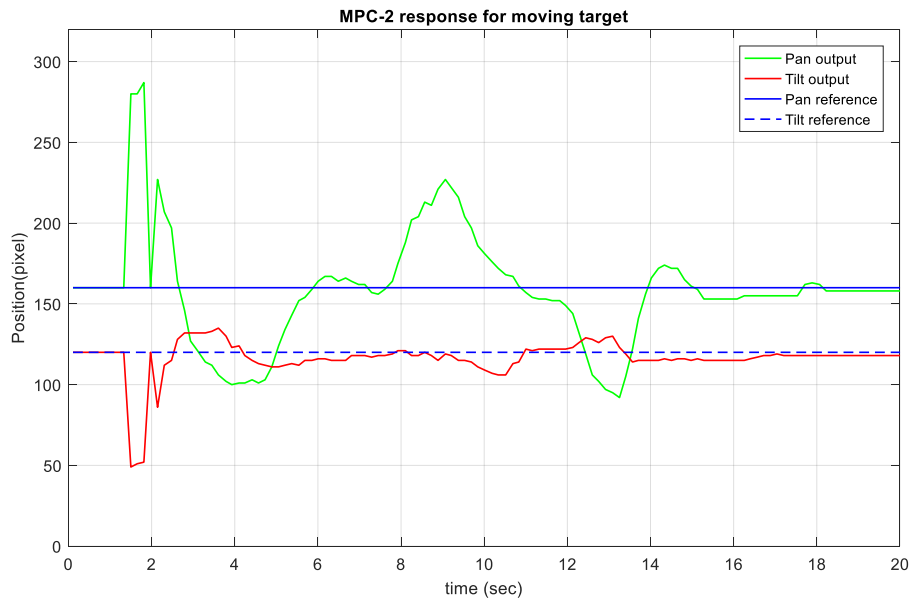


Figure 11. System response for moving target

9. Conclusion

From the discussion in the section above, it could be shown that the proposed system has the ability to identify a person while running on a considerably faster speed than face recognition as a standalone method. In a normal setting and normal camera, without abrupt movement, it is possible to achieve speed 30-50 fps while the system runs in the tracking mode. When the tracker failed, it would just take few frames to re-identify the person (few frames when the system runs the face-recognition mode with speed around 6 fps), before back to the tracking mode which is considerably faster. Complemented with an optimal hardware tracker, it could be concluded that the method proposed in this paper successfully achieved the goal to exploit the tracker promptness while maintaining the ability to distinguish the person of interest with the other person and backgrounds.

10. Acknowledgment

This research is a part of Research, Community Services, and Innovation program (Program Penelitian, Pengabdian kepada Masyarakat dan Inovasi/P3MI) funded by the Institute for Research and Community Services at the Institut Teknologi Bandung (LPPM ITB).

11. References

- [1] V. Kazemi, J. Sullivan, "One millisecond face alignment with an ensemble of regression trees", *IEEE Conference on Computer Vision and Pattern Recognition* 2014
- [2] S.R. Yosafat, C. Machbub, E.M.I. Hidayat, "Design and implementation of Pan-Tilt control for face tracking", *International Conference on System Engineering and Technology* 2017
- [3] S. Suryadarma, T. Adiono, C. Machbub, T.L.R. Mengko, "Camera object tracking system", *International Conference on Information and Communications Security* 1997
- [4] Andriana, A.S. Prihatmanto, E.M.I. Hidayat, C. Machbub, "Combination of face and posture features for tracking of moving human visual characteristics", *International Journal Electrical Engineering and Informatics* 2017

- [5] F.J.R. Ruiz, I. Valera, L. Svensson, F. Perez-Cruz, "Infinite Factorial Finite State Machine for Blind Multiuser Channel Estimation", *IEEE Transactions on Cognitive Communications and Networking*, Vol 4 no 2, pp. 177-191, 2018
- [6] Z. Chen, Y. Wang and F. Wang, "The research of control for sintering burn-through point based on finite-state machine," 2011 *International Conference on Electric Information and Control Engineering*, pp. 1830-1833., 2011
- [7] J. Li, Z. Wang and Y. Zhang, "An Implementation of Artificial Emotion Based on Fuzzy State Machine," 2011 *Third International Conference on Intelligent Human-Machine Systems and Cybernetics*, pp. 83-86. 2011
- [8] https://github.com/ageitgey/face_recognition (accessed March 3rd, 2018 2:50 PM)
- [9] N. Dala, B. Triggs, "Histograms of oriented gradients for human detection", *IEEE Conference on Computer Vision and Pattern Recognition 2005*
- [10] D.E. King, "Dlib-ml: a machine learning toolkit", *Journal of Machine Learning Research* 10, pp. 1755-1758, 2009
- [11] Amos, B. Ludwiczuk, M. Satyanarayanan, "OpenFace: A general-purpose face recognition library with mobile applications", *School of Computer Science Carnegie Mellon University*, 2016
- [12] Schroff, D. Kalenichenko, J. Philbin, "FaceNet: A unified embedding for face recognition and clustering", *IEEE Conference on Computer Vision and Pattern Recognition 2015*
- [13] Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K, "Speed/accuracy trade-offs for modern convolutional object detectors.", *IEEE Conference on Computer Vision and Pattern Recognition 2017*
- [14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications", arXiv:1704.04861, 2017
- [15] W. Liu, D. Anguelov, D. Erhan. C. Szegedy, S. Reed, C.Y. Fu, A. C. Berg, "SSD: Single Shot Multibox Detector", *European Conference on Computer Vision 2016*
- [16] <https://github.com/yeephycho/tensorflow-face-detection> (accessed June 7th, 2018 10:45 AM)
- [17] J. F. Henriques, R. Caseiro, P. Martins, J. Batista, "High-speed tracking with kernelized correlation filters", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014
- [18] Bradski, "The Open CV library", *Dr. Dobb's Journal of Software Tools*, 2000
- [19] M. Frigo, S. G. Johnson, *Circulant Matrices Lecture Note*, MIT, 2017
- [20] R. Rifkin, R. Rifkin, G. Yeo, and T. Poggio, "Regularized least-squares classification," *Nato Science Series Sub Series III Computer and Systems Sciences*, vol. 190, pp. 131–154, 2003.
- [21] <https://github.com/pytransitions/transitions> (accessed September 21st, 2018, 3:10 PM)
- [22] Corke, P, "Robotics, Vision and Control: Fundamental Algorithms in MATLAB." Springer, Berlin, Heidelberg, 2011.
- [23] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Trans. Robot. Automat.*, vol. 12, pp. 651–670, Oct. 1996.
- [24] F. D. Saragih, F. M. T. R. Kinasih, C. Machbub, P. H. Rusmin, A. S. Rohman, "Visual Servo Application Using Model Predictive Control (MPC) Method on Pan-Tilt Camera Platform", *International Conference on Instrumentation, Control, and Automation 2019*.
- [25] Richalet J, "Industrial applications of model based predictive control," *Automatica*, Vol. 29, 1993, pp. 1251–74.
- [26] M.Abu-Ayyad, R. Dubay, "Real-time comparison of a number of predictive controllers", *ISA*



Fabiola Maria Teresa Retno Kinasih was born in Surabaya, Indonesia in 1996. She received B.Sc. and M.Sc. in electrical engineering from Institut Teknologi Bandung (ITB), Indonesia, in 2016 and 2019 respectively. Since 2017, she is an Academic Assistant at School of Electrical Engineering and Informatics ITB, Indonesia. Her research interest vary from Computer Vision to Instrumentation and Control.



Christ Freben Dommaris Saragih was born in Padalarang, Indonesia in 1986. He received B.Eng. from Maranatha Christian University, Indonesia in 2009 and M.Sc. in electrical engineering from Institut Teknologi Bandung (ITB), Indonesia in 2019. Since 2009, he is an Automation Engineer and currently serving as an Electric Area Head at Riau Andalan Pulp and Paper (RAPP), Indonesia. His research interest is focused on Automation, Instrumentation, and Control.



Carmadi Machbub got Bachelor degree in Electrical Engineering from the Institut Teknologi Bandung (ITB) in 1980, Master degree (DEA) in Control Engineering and Industrial Informatics in 1988, and Doctorat degree in Engineering Sciences majoring in Control Engineering and Industrial Informatics from Ecole Centrale de Nantes in 1991. He is now Professor and Head of Control and Computer Systems Research Division, School of Electrical Engineering and Informatics, ITB. His current research interests are in control, machine perception and intelligent systems.



Pranoto Hidayat Rusmin was born in Magelang, Indonesia in 1972. He received B.Eng., M.Eng., and Doctor degrees in electrical engineering from Institut Teknologi Bandung (ITB), Indonesia, in 1996, 1999, 2009, respectively. Since 1998, he is a Lecturer at School of Electrical Engineering and Informatics ITB, Indonesia. His research interest is Internet Congestion Control.



Leni Yulianti was born in Bandung, West Java, Indonesia on July 1977. She received her Bachelor, Master and Doctoral Degree in Electrical Engineering from ITB. She is now a lecturer and researcher in School of Electrical Engineering and Informatics, ITB. Her research interests include statistical signal processing, visual tracking, visual-based control and state estimation.



Dian Andriana got her bachelor, master, and doctoral degree in School of Electrical Engineering and Informatics, the Institut Teknologi Bandung (ITB) in 1997, 2009, and 2019, respectively. She is a researcher at the Research Center of Informatics of the Indonesian Institute of Sciences. Her researches interests include decision support and intelligent systems. She has 6 papers published in Scopus indexed journals and conferences.