Paper: ASAT-16-097-CE

16th International Conference on
AEROSPACE SCIENCES & AVIATION TECHNOLOGY,
ASAT - 16 – May 26 - 28, 2015, E-Mail: asat@mtc.edu.eg
Military Technical College, Kobry Elkobbah, Cairo, Egypt
Tel : +(202) 24025292 – 24036138, Fax: +(202) 22621908

Object Tracking Using Vision on Raspberry Pi

W. Askar[*], O. Elmowafy[†], A. Youssif [‡] and G. Elnashar [§]

Abstract:
Object tracking is an important task in several computer vision applications**.** Optical flow is one of the most widely used techniques in the image processing and video analysis fields. This paper implements an object tracking algorithm based on optical flow method to be computed by Raspberry Pi microcomputer. A Lucas–Kanade method has been used to calculate the velocity vector of the moving object between two consecutive frames. Two experiments are performed to evaluate the robustness of the proposed algorithm by the new computing device. The results were encouraging to use the proposed framework on wide variety of real time application.

Keywords: Video Tracking, Optical Flow, Lucas–Kanade, Raspberry Pi

1. Introduction

Video tracking is an important task within the field of computer vision. It is a fundamental step in many civilian and military applications. Many of tracking algorithms has been well studied in the last decades and implemented in different frameworks. The optical flow is the distribution of apparent velocities associated with apparent motion in the time varying images, [2]. There are many techniques that handle the computation of the optical flow. They can be classified into global and local techniques. Global techniques such as the Horn/Schunck approach yield dense flow fields while local methods such as the Lucas–Kanade technique are more robust under noise, [16]. Lucas–Kanade method carried out the computation of the optical flow in the neighborhood of the pixel algebraically, [3]. So, Lucas-Kanade method is preferred in single object tracking applications.

Tracking an object, in general, is a very challenging task and many problems have to be solved. The loss of information due to the projection of the 3D world on a 2D image is the main source of those problems. Also the cluttered-background, noise in images, partial or full occlusions, illumination changes and the real-time processing requirements are well-known difficulties [1-3]. In applications such as remote surveillance, troops tracking and terrorist chasing, which may performed by Unmanned Arial Systems (UASs), different kinds of constrains will be taken into consideration. The minimum weight and size, low power consumption, suitable cost and the real time processing capabilities are the most important requirements. The Raspberry pi microcomputer is one of promising embedded systems that can achieve most of the requirements with acceptable results.

[*] Egyptian Armed Forces, wesamaaa@gmail.com
[†] Dr., Computer Engineering Dept., New Cairo Academy, drosama@ebcegypt.com
[‡] Prof., Computer Engineering Dept., Helwan University, aliaay@yahoo.com.
[§] Egyptian Armed Forces, enjygamal@gmail.com.

This paper aims to estimate the object position as precisely and computational efficiently as possible on the Raspberry Pi microcomputer. It uses the Lucas-Kanade algorithm with some extensions and contributions to meet the real time requirements in order to implement a vision based object tracking system. It divided into six sections in addition to the references. The first section is the introduction. The second section is about the related works to this paper. The third section provides descriptive information about the framework environment. In section four, the proposed Raspberry Pi Tracking algorithm Framework is explained. Then the experimental results are demonstrated in section five. Finally, the paper ended by the conclusion in section six.

2. Literature Survey

The computation of 2-d image velocities, or optical flow, is a common topic in video tracking area. Optical flow is a dense field of displacement vectors. It defines the translation of each pixel in a region. Popular techniques for computing dense optical flow include methods by Horn and Schunck, and Lucas and Kanade. They computed flow vector using the brightness constraint, which assume brightness constancy of corresponding pixels in consecutive frames, [3]. For optical flow estimation, Horn and Schunck introduced both the brightness constancy and the spatial smoothness constraints. Their implementation is not robust to outliers caused by reflection, occlusion, motion boundaries etc., [6].

Lucas-Kanade algorithm is one of the most common optical flow algorithms that estimate the optical flow by simple way. Comparative studies designated that the Lucas-Kanade algorithms deliver accurate results while being significantly more efficient than other optical flow methods, [1,5]. It computes the optical flow vectors for any pixel by including its neighborhood into the calculations. Lucas-Kanade algorithm is a local optical flow technique. It assumes that the flow field is smooth locally, [5].

Shi and Tomasi proposed a tracking algorithm which iteratively computes the translation of a region (patch) centered on an interest point. They constructed the similar optical flow equation as that proposed by Lucas and Kanade. The KLT tracker evaluates the quality of the tracked patch by computing the affine transformation between the corresponding patches in consecutive frames to choose between continue tracking the feature or eliminate it, [3].

Bouguet et.al implemented a pyramidal Lucas Kanade feature Tracker, [14]. He represented the image by its Gaussian pyramid to compensate the large movement of the tracked object. A sub-pixel computation technique was introduced by using the bilinear interpolation for the non-integer pixel coordinates inside the loops of pyramids and iterations. Although this way gives accurate results, it consumes a lot of computation time. This makes it not suitable for real time application, especially when implemented on limited speed hardware. Also, he considered the same weight for all pixels neighboring the tracked point inside the moving window. On contrary, Sepehr et.al consider a 2-D Gaussian distribution as a coefficients for the pixel neighborhood to increase the accuracy, [13].

Rinu et.al. implement optical flow motion detection algorithm on Raspberry Pi. They used the Lucas-Kanade method to detect the motion for the whole image frame, [7]. Their algorithm compares between two successive image frames to find out a displaced object. They used Python and OpenCV for implementing the algorithm. The algorithm works well only at moderate object speeds.

Jianbo et.al proposes criteria to select features that can be tracked well, [15]. The tracking algorithm utilizes their method to select the strongest point representing the object inside the initially selected rectangle.

3. Locus-Kanade Optical Flow Calculation:

The Lucas-Kanade method is one of the most important algorithms of two-frame differential methods for motion estimation [4]. To calculate the velocity vector (optical flow) of an arbitrary point $P = (x, y)$ between two consecutive frames as shown as Fig.1; the Locus-Kanade method assumes that the intensity value of point P at frame I at time t will be the same at time $t + dt$.

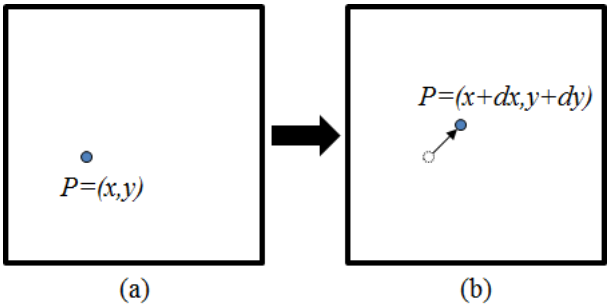


Fig. 1. Velocity Vector (Optical Flow) of Point $P = (x, y)$: (a) Image Frame $I(x, y)$ at Time t, (b) Image Frame $I(x + dx, y + dy)$ at Time $t + dt$

This assumption can be formulated as the following:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{1}$$

Applying the first order Taylor series approximation on equation (1) to obtain the optical flow constancy equation:

$$I_x . u \; + \; I_y . v = - \, I_t \tag{2}$$

Where I_x and I_y are the x-component and y-component of the gradient vector respectively (spatial derivatives), u and v are the components of the optical flow vector and I_t is the temporal derivative at point P. To emphasize the terms of spatial and temporal derivatives, eq. (1) may be re-written as the following:

$$\Delta I . [u \; v]^T + \; I_t = 0 \tag{3}$$

Although the derivative terms of equation (3) can be determined for point P, it still one equation with two unknowns u, v. Suppose a window with size $w \times w$ to be the neighborhood of the point P and suppose all pixels in it have the same u, v. Then equation (3) will expand into:

$$\begin{bmatrix} I_x\,(p_1) & I_y\,(p_2) \\ I_x\,(p_1) & I_y\,(p_2) \\ \vdots & \\ I_x\,p_{w\times w} & I_y\,(p_{w\times w}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t\,(p_1) \\ I_t\,(p_2) \\ \vdots \\ I_t\,(p_{w\times w}) \end{bmatrix} \tag{4}$$

Now equation (4) has more equations with two unknowns u, v. So it can be handled as the following:

$$\boldsymbol{A}.\boldsymbol{U} = \boldsymbol{B} \tag{5}$$

Where $\boldsymbol{U} = [u, v]^T$, $\boldsymbol{A}$ and $\boldsymbol{B}$ are the $w \times 2$ most left and the $w \times 1$ most right matrices.

$$(\boldsymbol{A}^{-\boldsymbol{T}}\boldsymbol{A})\ \boldsymbol{U} = \boldsymbol{A}^{-\boldsymbol{T}}\boldsymbol{B} \tag{6}$$

Finally, by solving equation (6) the displacement values u, v can be calculated from the following formula:

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} \tag{7}$$

The term $\boldsymbol{A}^{-\boldsymbol{T}}\boldsymbol{A}$ have to be invertible with recommended high eigenvalues, [14, 15].

4. The Tracking Algorithm Environment

The proposed tracking algorithm is programmed on Raspberry Pi microcomputer using Python and OpenCV to take the advantages of each component of them. It implements an object tracking module to be used in various unmanned systems especially UAVs. Also, other python library such as Numpy is utilized for its importance.

A. Raspberry Pi hardware

The Raspberry Pi is a low cost, small weight and credit-card sized computer, [8]. It is essentially a system-on-a-chip (SoC) with connection ports. It has a 32-bit ARM processor which delivers a range of processing speed from 700 - 1000 MHz and Videocore 4 GPU. The Raspberry Pi model B has a 512 Mb SDRAM comes with two USB ports and can be connected to an Ethernet network, [8]. It supports modern technologies such as OpenGL ES2 and hardware accelerated audio/video processing. These capabilities make the Raspberry Pi an exceptional platform for computer vision applications. The processor in the Raspberry Pi runs at 700MHz by default. But it can be overclocked by increasing its frequency with no problems happened on all speed modes [8].

B. Raspbian Operating System

Although Raspberry Pi can utilize different types of Linux distributions for its default operating system (OS), Raspbian is the recommended one. It is a free operating system based on Debian, optimized for the Raspberry Pi hardware. It comes with over 35,000 packages; precompiled software bundled in a good format for easy installation on Raspberry Pi hardware, [9].

C. Python and OpenCV

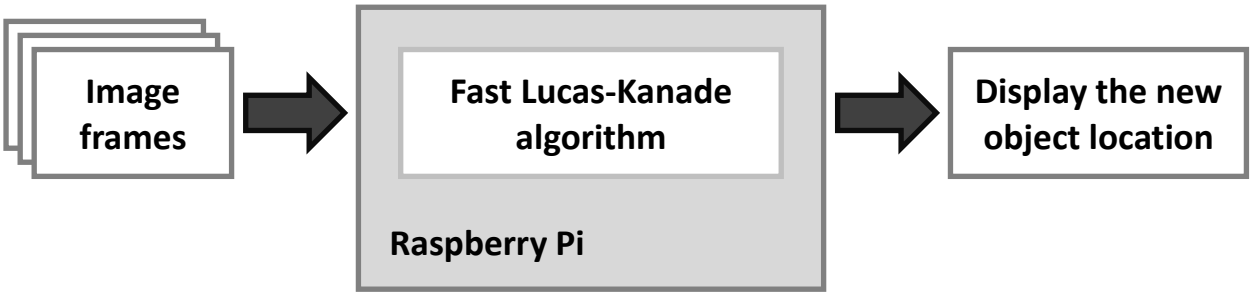
It can be programmed with Python or any other language that will compile for ARM v6, [10]. Python is a widely used general-purpose, high-level powerful programming language. Its design philosophy emphasizes code readability, and its syntax allows expressing concepts in fewer lines of code than would be possible in languages such as C++ or Java. Python supports multiple programming paradigms including object-oriented, imperative and functional programming or procedural styles. It offers all facilities that are wanted for programming

from the basic operations to advanced functions. It integrates with a lot of third-party tools to make everything possible in Python, [11].

In image processing and computer vision field the OpenCV, PIL and Numpy are famous libraries integrated with Python. OpenCV is an open-source library that includes grate number of computer vision algorithms. In general, Python is slower compared to other languages like C/C++. But it can be easily extended with C/C++ by writing computationally intensive codes in C/C++ and create a Python wrapper (package) for it. Then use these wrappers as Python modules. This gives two advantages: first, the code is as fast as original C/C++ code (since it is the actual C/C++ code working in background) and second, it is very easy to code in Python. This is the way that OpenCV-Python works in i.e. a Python wrapper around original C/C++ implementation, [12]. Numpy is the fundamental package for scientific computing in Python. It is very important to use Numpy for the reason of minimum computational time requirements. It is used to speed up the linear algebraic calculation, [17].

5. The Proposed Tracking Framework

This paper use Lucas-Kanade method to implement point tracking algorithm within a new framework based on Raspberry Pi microcomputer. Fig. 2 shows the block diagram of the proposed system.



As mention before, the Raspberry Pi microcomputer can overclock to perform processing at 1000 MHz speed. However, this speed still limited in object tracking task performed by several optical flow implementations. This limitation was taken into account in each step of the algorithm implementation. However, focusing on that the main problem is point tracking not motion detection [7], it can overcome this limitation relatively.

Back again with equation (7), all derivative terms I_x , I_y and I_t have to be calculated as fast as possible. Also their algebraic operations including matrix multiplications, raises to power, inverts and summations must be handled carefully. The sub-pixel accuracy calculation plays an important role in almost all steps. So the good manipulation of it gives perfect results.

The simple but efficient approaches were used to calculate the spatial and temporal derivatives. Let $Iwin$ be a sub-region of image I which centered on the tracked point$P$. It was extracted from I directly with the sub-pixel accuracy. Then the spatial derivatives I_x , I_y were calculated simply as the following:

$$I_x(x,y) = (Iwin(x+1,y) - Iwin(x-1,y))/2 \quad (8)$$

$$I_y(x,y) = (Iwin(x,y+1) - Iwin(x,y-1))/2 \quad (9)$$

The temporal derivatives I_t were handled simply by obtaining the difference between the current frame sub-region $Jwin$ and the previous one $Iwin$ as the following:

$$I_t(x,y) = Jwin(x,y) - Iwin(x,y) \tag{10}$$

Instead of using the exhausted loops fashion proposed by Bouguet [14], the proposed algorithm uses the algebraic libraries delivered by Numpy package to perform array and matrix operations.

To increase the accuracy a 2-D Gaussian distribution was considered as coefficients (w_g) for all Point P neighborhood just like Sepehr et.al, [13]. So, equation (7) would be modified to the following:

$$\begin{bmatrix} \sum w_g I_x I_x & \sum w_g I_x I_y \\ \sum w_g I_x I_y & \sum w_g I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum w_g I_x I_t \\ \sum w_g I_y I_t \end{bmatrix} \tag{11}$$

All above values had been calculated for each pyramid level and an iteration process was continued until either the displacement vector error is less than a certain threshold or the number of iterations exceeds certain value. The threshold and allowed number of iteration values all was predetermined experimentally.

6. Experimental Work

To examine the proposed system two main experiments are introduced. The first was by using the pre-implemented Lucas-Kanade tracking function provided by the OpenCV library. The second was by the proposed algorithm. All experiments were run initially on the Intel core 2 duo CPU to choose the fastest algorithm. All experiments were done on the purposely selected video, [18] and the five image sequences from PETS datasets, [19]. The frame size of all videos is 640×480 pixels and the results were approximately the same. Fig. 3 demonstrates an object tracked during some image frames.

Fig. 3. Object Tracking

Fig. 4 shows a comparison between the execution time in case of the traditional Lucas-Kanade implementation [14] and the proposed one. It is clear that the proposed algorithm calculated the displacement vector at approximately half time the OpenCV had. This was encouraging to implement it on the Raspberry Pi board.

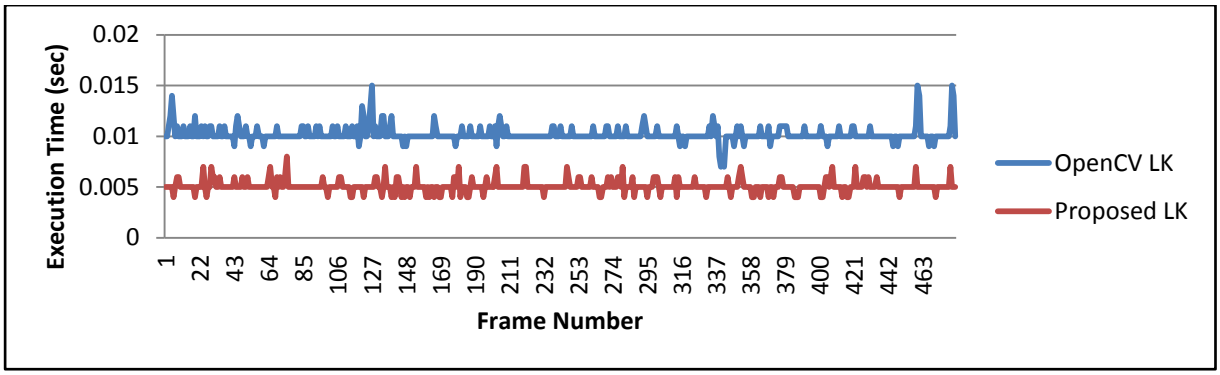


Fig. 4. Execution time comparison between Lucas-Kanade implemented by OpenCV and the proposed method

On the Raspberry Pi microcomputer an execution time comparison was performed between the Lucas-Kanade traditional implementation [14] and the proposed method. There was a significant inhancment on the calculation time as shown in Fig. 5.

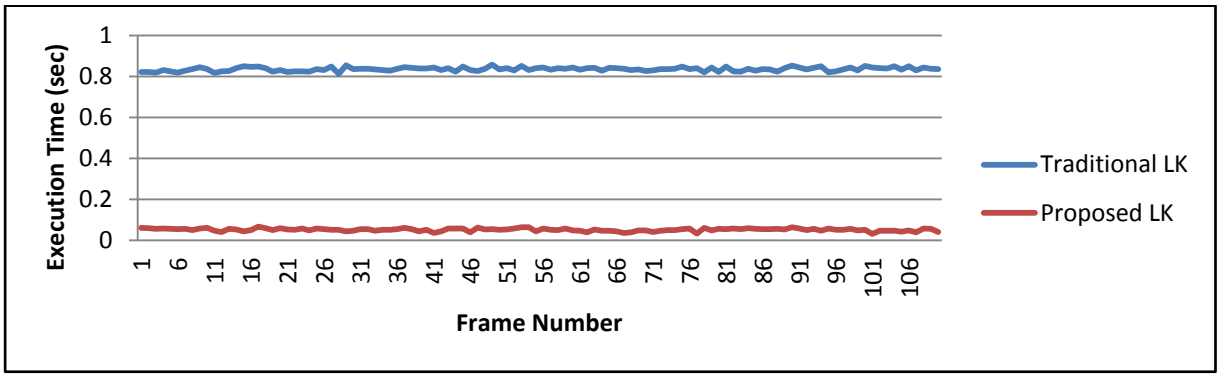


Fig. 5. Execution time comparison between Lucas-Kanade traditional implementation and the proposed method

The error of the displacement vector was calculated at different values of the search window(W), which is equivalent to sub-region mention in section 4, and the number of allowed iterations. After some experiments the values $W = 10$ and $K = 5$ gave suitable results for speed and accuracy.

The efficiency of the overclocking is emphasized. This is performed by switching between the different computational speeds and measure the execution time of the optical flow in the tracked point P in each case. Table 1 shows the average speeds of the tracking which can reach 20 frames per second. Fig. 6 demonstrates the resulted time processing to calculate the optical flow for P.

Table 1 Average calculation time for arm processor overclocking speeds

		Average calculation second/frame
Processor Speeds	700 MHz	0.087523
	800 MHz	0.084929
	900 MHz	0.076089
	950 MHz	0.072786
	1000 MHz	0.050638

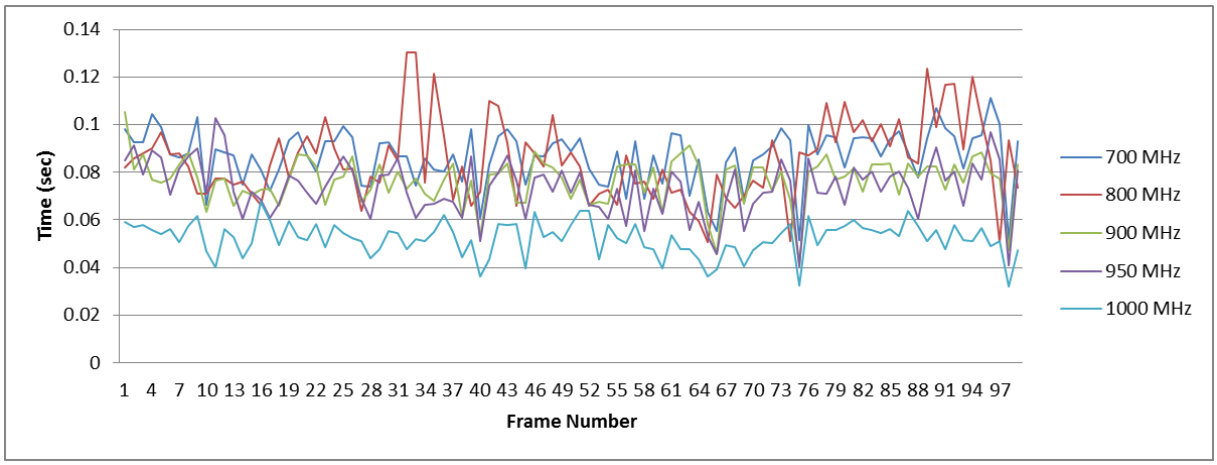


Fig. 6. Execution time comparison between different overclocked speeds

7. Conclusion

A Lucas-Kanade algorithm is implemented with optimizing techniques to be hosted and processed on the raspberry pi microcomputer. Different experiments were performed to compare between the OpenCV Lucas–Kanade tracking function and the traditional Bouguet one with the proposed algorithm. Results were encouraging to use the new framework with the proposed algorithm in many point tracking applications. It has been promising to use for unmanned aerial systems (UAS) that require special hardware configurations. This is because of its minimum weight and size and low power consumptions. In the future, the propose framework will be hosted on a UAV for vision based object hovering and chasing.

8. References

[1] Hanxuan Y., Ling S., Feng Z., Liang W. and Zhan S., "Recent advances and trends in visual tracking: A review," ", *Elsevier,"* August 24, 2011.

[2] Emilio, M. and Andrea C, *Video Tracking Theory and Practice*, 1st. ed., Vol. 1, Wiley, New York, 2011, pp. 1-71.

[3] Alper Y., Omar J. and Mubarak S., "Object Tracking: A Survey," ", *ACM Computing Surveys, Vol. 38, No. 4, Article 13,"* December, 2006.

[4] Ekta, P. and Dolley, S., "Comparison of Optical Flow Algorithms for Speed Determination of Moving Objects," ", *International Journal of Computer Applications,*" 2013, pp. 0975 – 8887.

[5] Tobias, B., Volker, E., and Thomas, S., "Robust Local Optical Flow for Feature Tracking," ", *IEEE Transactions on Circuits and Systems for Video Technology,"* September, 2012.

[6] Deqing S., Stefan R., Lewis J. P., and Michael J. B., "Learning Optical Flow," ", *Springer-Verlag Berlin Heidelberg (ECCV),"* 2008, pp. 83–97,.

[7] Rinu M. B. and Rooha R. A., "Optical Flow Motion Detection on Raspberry Pi," ", *Fourth International Conference on Advances in Computing and Communications,"* IEEE, August 27-29, 2014, pp. 151-152.

[8] Raspberry Pi Foundation, "Raspberry Pi Documentation", United Kingdom, 20/12/2014, http://www.raspberrypi.org/documentation/.

[9] Raspberry Pi Foundation, "RASPBIAN", United Kingdom, 18/11/2014, http://www.raspberrypi.org/documentation/raspbian/README.md.

[10] WhatIs.com, "Raspberry Pi ($35 computer)", April 2012, Tech Target, 22/12/2014, http://whatis.techtarget.com/definition/Raspberry-Pi-35-computer.

[11] Masoud N., "Python: An appropriate language for real world programming," ", *World Applied Programming,"* June 2011. pp. 110-117.
[12] Alexander M. and Abid K., *OpenCV-Python Tutorials Documentation*, 1st. ed., February 18, 2014, p. 6.
[13] Sepehr A. and Homayoun M., "Optical Flow Based Moving Object Detection and Tracking for Traffic Surveillance," ", *International Journal of Electrical, Robotics, Electronics and Communications Engineering,"* Vol. 7, No. 9, 2013.
[14] Bouguet J., "Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm," 1994.
[15] Jianbo S. and Carlo T., "Good Feature to Track," ", *IEEE Conference on Computer Vision and Pattern Recognition,"* June, 1994.
[16] Andres B. Joachim W. and Christoph S., "Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods", *International Journal of Computer Vision 61(3),"* 2005, pp. 211–231.
[17] NumPy community, *"NumPy User Guide,"* 1.8.0. ed., November, 2011.
[18] Admin Aerobot, "Drone Racing a GTR", February 2012, 5/1/2015 https://www.youtube.com/watch?v=TXTucbmk2o8.
[19] VIVID Tracking Evaluation Web Site, "PETS2005", April 2014, 22/12/2014, http://vision.cse.psu.edu/data/vividEval/datasets/datasets.html.