A Parallel Implementation of Multiple Non-overlapping Cameras for Robot Pose Estimation

Mohammad E. Ragab[1], and Ghada F. Elkabbany[2]
[1]Informatics Dept., Electronics Research Institute
Cairo, Egypt
[e-mail: mehab@hotmail.com]
[2]Informatics Dept., Electronics Research Institute
Cairo, Egypt
[e-mail: gelkabbany@eri.sci.eg]
*Corresponding author: Ghada Elkabbany

Received July 14, 2014; accepted September 16, 2014; published November 30, 2014

Abstract

Image processing and computer vision algorithms are gaining larger concern in a variety of application areas such as robotics and man-machine interaction. Vision allows the development of flexible, intelligent, and less intrusive approaches than most of the other sensor systems. In this work, we determine the location and orientation of a mobile robot which is crucial for performing its tasks. In order to be able to operate in real time there is a need to speed up different vision routines. Therefore, we present and evaluate a method for introducing parallelism into the multiple non-overlapping camera pose estimation algorithm proposed in [1]. In this algorithm the problem has been solved in real time using multiple non-overlapping cameras and the Extended Kalman Filter (EKF). Four cameras arranged in two back-to-back pairs are put on the platform of a moving robot. An important benefit of using multiple cameras for robot pose estimation is the capability of resolving vision uncertainties such as the bas-relief ambiguity. The proposed method is based on algorithmic skeletons for low, medium and high levels of parallelization. The analysis shows that the use of a multiprocessor system enhances the system performance by about 87%. In addition, the proposed design is scalable, which is necaccery in this application where the number of features changes repeatedly.

Keywords: Pose Estimation, Multiple-cameras, EKF, Robot Navigation, and Parallel Processing.

A preliminary version of this paper appeared in 17th IEEE International Conference on Image Processing (ICIP), PP: 3253 – 3256, Hong Kong, Sept. 26-29, 2010. This version enhances the sysytem performance by using parallel processing.

http://dx.doi.org/10.3837/tiis.2014.11.025

1. Introduction

Computer vision is gaining a larger importance in a variety of applications such as: robotics, man-machine interaction, and 3D surgical operations. It not only allows a great deal of flexibility but also the lowest degree of invasiveness compared to the most of other sensor systems. In order to be able to operate in real time there is a need to speed up different vision routines [2]. Computer vision is considered a good candidate for the application of parallel processing because of the large volumes of data and the complex algorithms commonly encountered. Parallel image processing systems can be classified into three categories: computer-based dedicated systems, computer-based general systems and digital signal processing (DSP)-based systems. According to this classification, the analysis and comparison of many kinds of realization technologies and structure characteristics are carried out in [3], and [4]. For an autonomous mobile robot, the first step to do any worthwhile task is to determine its location and direction. This is the pose estimation which is a crucial problem lasting-for-decades in computer vision as well as in various other fields. Obtaining the robot ego-motion is one of its many facets. The application range extends from mixed reality in movies to activity recognition [5], and guidance of bronchoscopic tracking [6].The EKF has been used to solve the pose estimation problem in different settings. The two main aspects of this use are: firstly, the number and arrangement of cameras, and secondly, the number and usage of filters. For example, a single camera and one EKF for both pose and structure are used in [7], [8], and in [9]. Additionally, a single camera, one EKF for pose, and many EKFs for structure are used in [10]. Moreover, four cameras arranged in two back-to-back stereo pairs, and one EKF for pose are used in [11] while the structure is obtained on-demand using triangulation. On the other hand, two cameras arranged as a one stereo pair, and one EKF for pose are used in [12] without solving for structure. An important benefit of using multiple cameras for robot pose estimation is resolving the bas-relief ambiguity [13]. In [14], [15], and [16], the multiple cameras are dealt with as a single generalized camera. Multiple cameras are used in [17] and in [18] mainly as fixed cameras to estimate the pose of an object with a known CAD model. To sum up, our motivation is to make use of the parallel processing to estimate the pose of a moving robot within an unknown indoor scene in real time. In particular, we speed up the multiple non-overlapping camera implementation in [1]. The analysis shows that the use of a multiprocessor system enhances the system performance by about 87%. In addition, the proposed design is scalable (that is to say, increasing the number of cameras and the number of processors enhances the system performance), which is an important factor in this application where the number of features changes repeatedly.This paper is organized as follows: in section 2, a background of pose estimation problem and its implementation in multiprocessor systems is given. Then, the proposed method is detailed in section 3. In section 4, a performance evaluation of the proposed method is given. Finally, the paper is concluded in section 5.

2. Background and Related Work

2.1 Robot Navigation and Path Planning

For any mobile platform (a robot or a vehicle), the ability to navigate in its environment is one of the most important capabilities at all. Robot navigation means its ability to determine its own position with respect to a frame of reference and then to plan a path towards some goal location. Mobile robot path planning in an unknown environment is a major problem in robot navigation. This process is crucial because it offers the robot the flexibility of reaching a point securely without any assumed prior knowledge about the scene. In the literature, mobile robot path planning problems are solved in various ways. For example, ultrasonic sensors are used in [19] and [20], and laser range finders and CMOS coupled devices (CCD) arrays are used in [21], [22], [23], [24], and [25]. These schemes allow a robot to use the sensors to map out its surrounding environment as a grid of empty (no obstacle), occupied (obstacle) or unknown locations (out of sensor's area). When a robot is introduced into a new environment, it takes the readings of all the available sensors. Through accumulating these readings, several points on the grid can be labeled as empty or occupied. Accordingly, the robot travels through the empty space. This process is repeated at each motion step of the mobile robot until it arrives at its destination.

2.2 Pose Estimation of a Multiple Camera System

Multiple cameras are increasingly prevalent on robots and vehicles. These cameras come in a variety of models such as: wide-angle, fish-eye, and catadioptric. Furthermore, although odometry is generally available on the vehicles, it may be inaccurate due to wheel slipping. However, vision applications may use wheel odometry as a strong prior for camera pose estimation, and in this case, an accurate camera calibration is required [26]. Many researchers try to solve the pose estimation problem of a calibrated multi-camera system [26], [27], [28], and [1]. Heng et. al., proposed an easy-to-use automated pipeline that handles both intrinsic and extrinsic calibration [26]. They run an automatic intrinsic calibration for each generic camera using a checkerboard. Afterward, they run an extrinsic calibration to obtain all camera-odometry transforms. The extrinsic calibration is unsupervised, using natural features, and only requiring the vehicle to be driven around for a short time. The intrinsic parameters are optimized in a final bundle adjustment step while performing the extrinsic calibration. In addition, the pipeline produces a globally-consistent sparse map of landmarks which can be used for visual localization. Lee et. al., in [27], [28] suggest a model for solving the pose estimation problem of a calibrated multi-camera system. They assume that the non-central rays passing through the 3D world points and the multi-camera system can be represented as Plücker lines. They show that the minimal solution for the depth of the points along the Plücker lines is an eight degree polynomial that gives up to eight real solutions. The coordinates of the 3D world points in the multi-camera frame are computed from the known depths. In addition, the pose of the multi-camera system can be obtained from absolute orientation. Moreover, the solution can be refined by including all the inlier correspondences in a non-linear refinement step. In order to construct a real-time system, it is essential to aim for high-performance by minimizing the computational latency. Unfortunately, tracking objects is quite complex and requires a vast amount of computation. However, various techniques have been proposed in an effort to reduce the computation. One such technique verifies and updates information about the positions of the selected small windows, or features, of an object. Selection of such windows is generally determined using a priori basis for interesting features. It is necessary to carefully select the points where motion information can be extracted. The basis for the selection of a point could be tracking of corners, windows with high spatial frequency content, or regions with particular brightness patterns [29]. Tomasi and Kanade in their work [30], and [31] derive a criterion for feature selection based on large contrasts of intensity. The Kanade-Lucas-Tomasi (KLT) feature selection and tracking algorithm, commonly accepted within the vision community, is chosen for this work compared to other feature detectors, it verifies an adequate quality without too much computational demand. Ragab et. al., in [1] solve the problem of pose estimation in real time using multiple non-overlapping cameras and the Extended Kalman Filter (EKF). Four cameras arranged in two back-to-back pairs are put on the platform of a moving robot. The axes passing through the camera centers of each pair are perpendicular. Because each camera has its individual EKF for the pose estimation, this model is suitable for parallel processing.

2.3 Parallel Feature Selection and Tracking

Feature selection and tracking are two of the essential problems in computer vision. The selection of features can be based on intuitive descriptions of feature quality. The feature tracker presented by Kanade, Lucas, and Tomasi in [30], and [31] has approached the selection of features in a way that is optimal by construction with respect to the accompanying tracking algorithm. That is to say, the KLT method has been successfully used for feature tracking as long as the baselines between successive frames are short. Feature selection involves evaluating every window in the image frame for a being textured by computing the eigenvalues of the 2×2 gradient matrix. Because the computation is completely independent for each window being examined, this is a natural candidate for acceleration using parallelism. Moreover, tracking in the KLT algorithm is accomplished by minimizing a dissimilarity measure between feature windows which is another source of potential parallelism [32]. Klein and Murray in [33] develop a system for Visual Simultaneous Localization and Mapping (VSLAM) called Parallel Tracking and Mapping (PTAM). In PTAM, online, real-time camera pose estimation (tracking) and structure estimation (mapping) are separated into two threads. There has been also much effort to implement the KLT on Graphic Processing Units (GPUs) to increase the speed especially with large frames and more features to track. Kim et. al., in [34] implement the KLT using an affine-photometric model on GPUs, albeit with a hight computational demand. Fassold et. al., in [35] reporte work done on carrying out the KLT using one GPU for each feature point. Their GPU implementations achieves real-time performance (> 25 frames per second) for High Definition (HD) video sequences and successfully track several thousands of points. Sinha et. al., in [36] combined the implementations of the KLT and Scale Invariant Feature Tracking (SIFT) feature extraction algorithms on GPUs which is suitable for video analysis in real-time systems. The GPU-based SIFT implementation extracts about 800 features from 640 × 480 video at ten frames per second which is approximately ten times faster than an optimized CPU implementation. In this work, we focus on using CPUs. The reason for this is the need for their general programing capabilities to deal with the problem at hand.

3. The Parallel Implementation of Multiple Non-overlapping Camera Pose Estimation

In this section we propose a new method for the parallel implementation of the pose estimation algorithm proposed in [1]. The algorithm therein presents the main problems of an autonomous mobile robot platform, which uses digital images for extracting information from the environment for visual navigation. In this algorithm, the pose estimation problem has been solved in real time using multiple non-overlapping cameras and the EKF. Four cameras arranged in two back-to-back pairs are put on the platform of a moving robot without an overlapping field of view. The two axes passing through the camera centers of each pair are perpendicular as shown in **Fig. 1**.

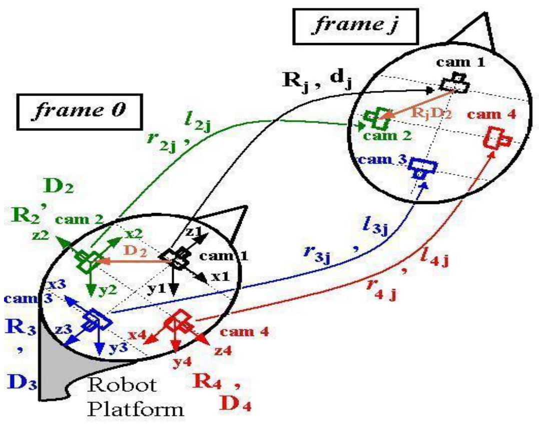


Fig. 1. The multiple non-overlapping camera model [1]

Each camera has its individual EKF for the pose estimation, therefore this model is suitable for the parallel processing on both levels of feature tracking and pose estimation. To solve this problem in a parallel environment, initially each camera (which represented by a task T_i) is assigned to one processor. During run-time, each processor can estimate the time needed for its tasks, and hence load sharing is possible. Since each task has a sequence of calculations to be performed, these calculations can be broken into sets of computations (subtasks) that can be executed by more than one processor. At the end of each motion step, processors may need to communicate the accumulated cameras' fillter outputs to make the motion decision. This decision must be done by only one processor. The amount of data to be transferred between processors, which increases the communication overhead, is changing based on the configuration area (image size $w*h$), the number of processors (M) and the length of motion sequence. The amount of data to be transferred will increase with the size of the configuration area. This process is repeated at each frame of the sequence until the robot reaches its destination. From the parallel processing point of view, this application is considered to be a divisible workload consisting of independent tasks of different sizes. The tasks' execution times are evaluated during run-time. **Fig. 2** represents the steps of the algorithm, where each vertex represents an individual task, while the edges are the dependencies among them (the communication between two adjacent tasks). The task T_0 represents the initialization process, tasks Ti's ($1 \leq i \leq 4$) represent the denoted camera pose estimation, and task T_5 represents the process of optimizing decision making based on the median filter and the invariant physical dimensions of the baselines between the cameras.

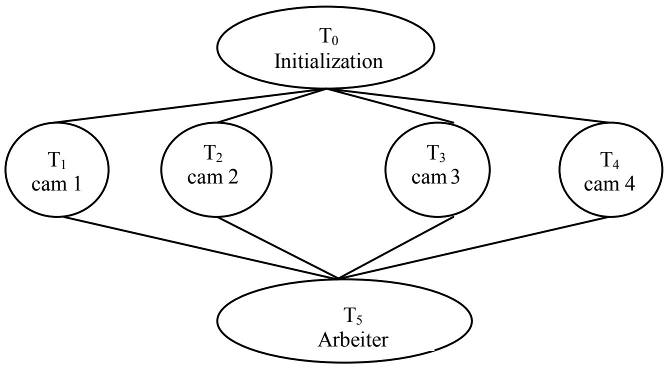


Fig. 2. System representation

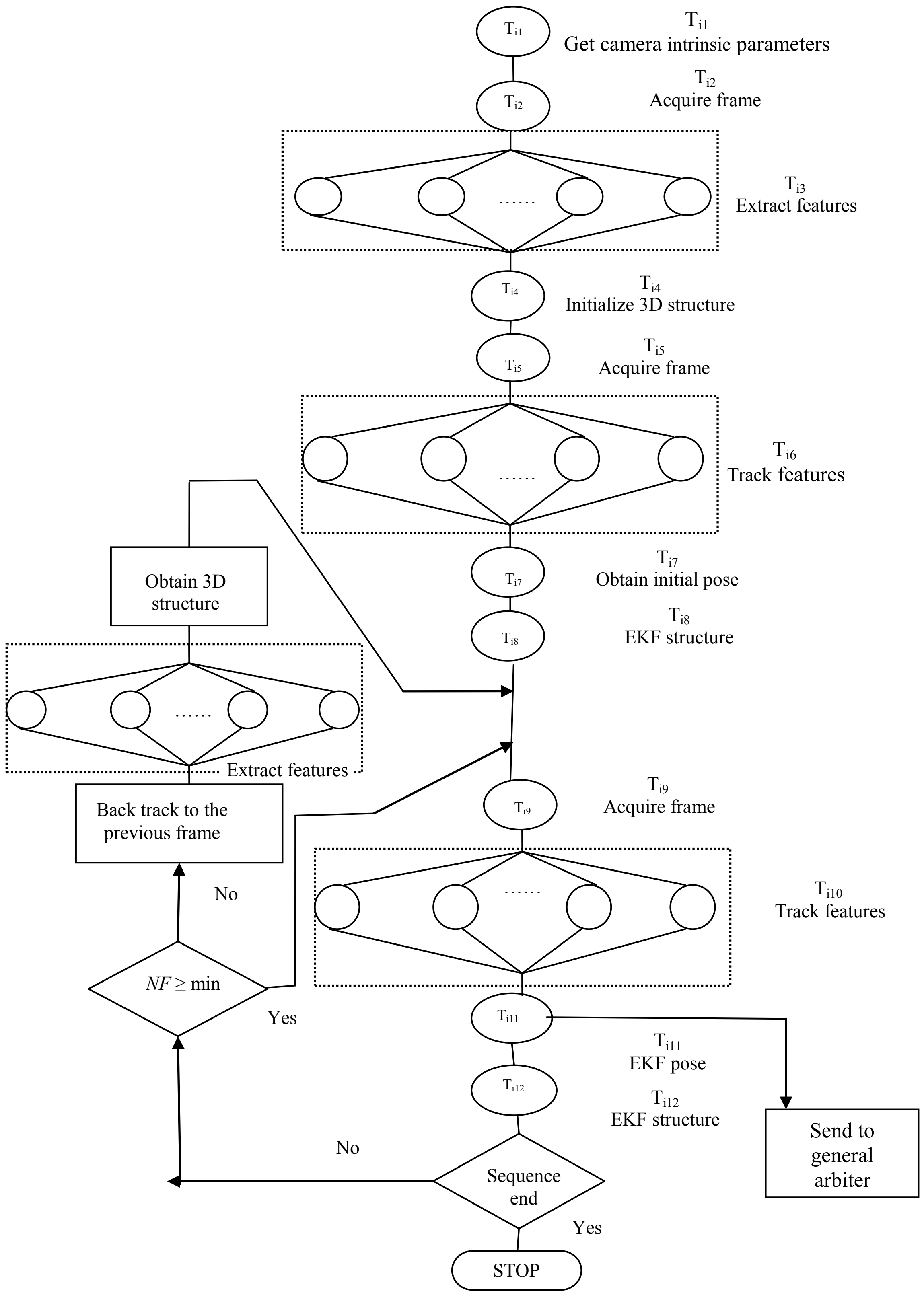


Fig. 3. System steps (these steps must be done for each camera, cam k) where ($1 \leq k \leq 4$)), *NF*: number of features, and *Min* is the minimum number of features

Moreover, **Fig. 3** represents the details of task T_i which describes the process of each camera (cam k). Based on the time needed to calculate these tasks they are assigned priorities and aranged in the in descending order (the camera which needs the largest execution time takes the highest priority). For "*M*" processors there are three cases: ($M \le 4$), ($4 < M \le 16$) and ($M > 16$) which will be explained below.

3.1 First Case ($M \le 4$)

When the number of processors "*M*" equals four (the number of cameras), each processor is sequentially assigned a camera from the list. In case of ($M < 4$), each processor is assigned $\left\lfloor \frac{4}{M} \right\rfloor$ task(s) from the list (where $\lfloor \ \rfloor$ denotes the floor function), and the remaining $\left(4 - \left\lfloor \frac{4}{M} \right\rfloor * M\right)$ tasks are assigned to the lightly loaded processor. As shown in **Fig. 2**, T_5 must be executed sequentially.

3.2 Second Case ($4 < M \le 16$)

In this case, we assume that the configuration area (image size $w*h$) can be divided into four quarters (Q_1, Q_2, Q_3, and Q_4), and each task T_i can be divided into four subtasks as shown in Figure 4. That is to say, each task T_i is assigned to $\left\lfloor \frac{M}{4} \right\rfloor$ processor(s) and the remaining $\left(M - \left\lfloor \frac{M}{4} \right\rfloor * 4\right)$ processors help the overloaded ones. These subtasks can be computed in parallel. To calculate the number of subtasks assigned to each processor, the following assumptions are made: First the number of processors assigned to the i^{th} task "T_i" is "N_i". Secondly, each subtask has its different execution time.

When ($M < 16$), and N_i is less than four, each processor is assigned $\left\lfloor \frac{4}{N_i} \right\rfloor$ subtask(s) and the remaining $\left(4 - \left\lfloor \frac{4}{N_i} \right\rfloor * N_i\right)$ subtasks are assigned to the lightly loaded processors. Finally, when ($M = 16$), each processor is assigned one subtask.

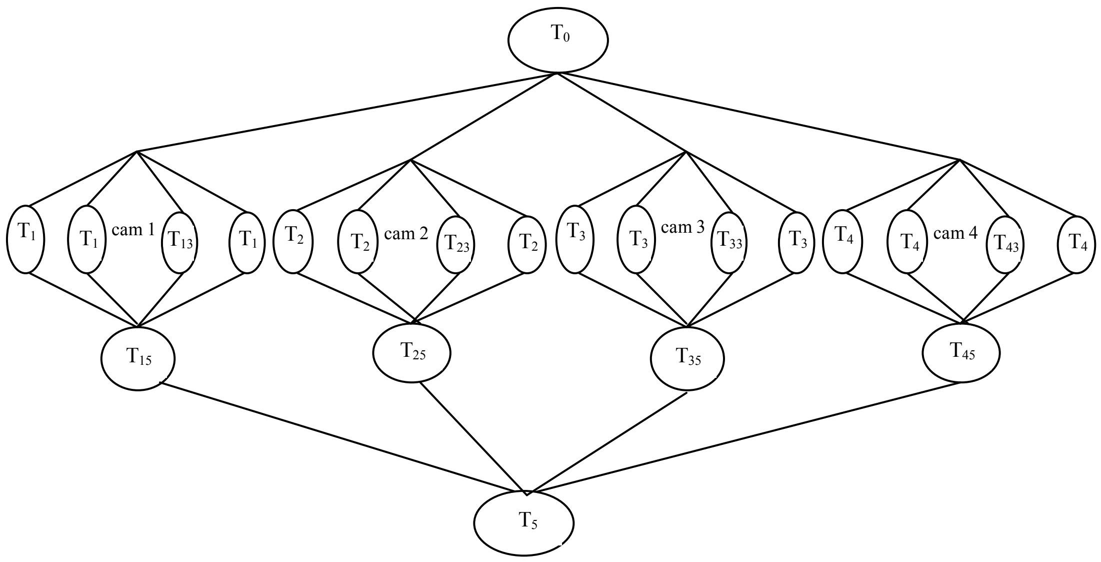


Fig. 4. Four quarter representation. T_{kl} $1\leq (k,l) \leq 4$, where k:denotes the camera, *l*: denotes the quarter, T_{k5} is the arbiter of cam k, and T_5 is the general arbiter

3.3 Third Case 3: *M* >16

Further enhancement is obtained when the number of processors is greater than 16. In this case, more than one processor can cooperate to execute task T_{kl} for each quarter. As shown in Figure 3 each task T_{kl} can be divided into small tasks called subtasks T_{i1}, T_{i2}, …., and T_{i12}. Each quarter can behave as a single camera as shown in Figure 5. This figures show that the most consuming time functions are: the feature extraction and the feature tracking, That is to say, more than one processor can cooperate to execute each quarter of each camera. In this case, the feature extraction and the feature tracking take the major amount of computation and can be executed in parallel after being divided into small subtasks. Finally, only one processor collects the data and computes the last task sequentially. This sequence must be repeated for all quarters belonging to each camera.

Execution time, which is refers to the total running time of the program, is the most obvious way of describing the performance of parallel programs. In fact, reducing this time is the aim of parallel processing. In the next sub-section, the execution time is calculated for different levels. Initially, for a frame within quqrter, then for a quarter, furthermore for a camera, and finally for the whole system.

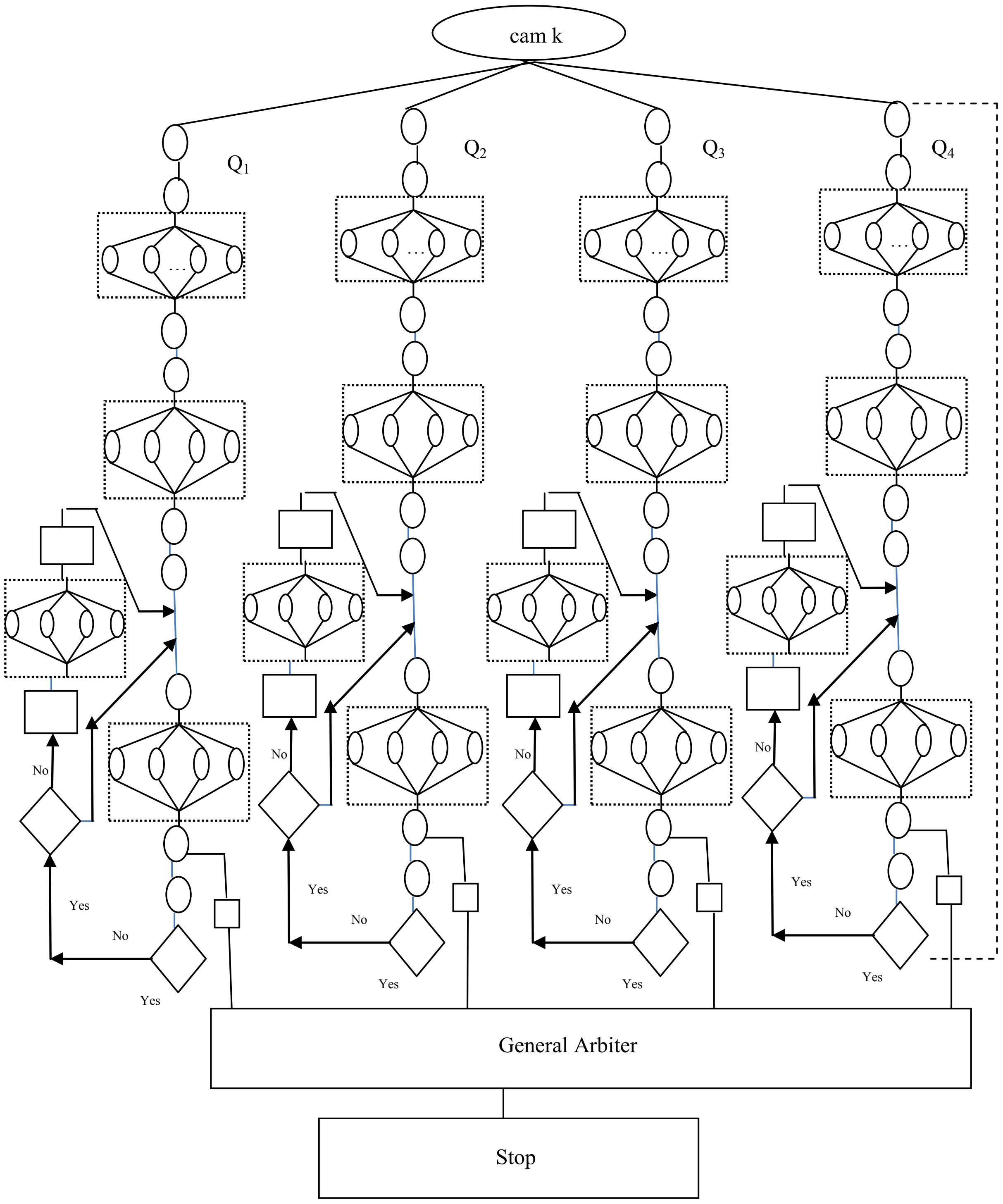


Fig. 5. Detailed four quarters representation. Each of the four branches has the same subtasks of Figure 3, albeit on the quarter level

3.4 Determining Parallel Execution Time

Assuming that T_{par} is the parallel execution time, "T_j" is the time needed to compute frame number "j" in parallel and $1 \le j \le$ length of sequence (LoS). Assume that for frame number "j" the total number of features is "NF_j", the minimum number of features for quarter number "l" is "$MinF_l$", and the number of processors need to execute each frame in quarter number "l" is "NP_l". Then, the execution time of quarter "l" of camera "cam k" is given by:

$$T_{kl} = T_{initial} + \sum_{j=1}^{LoS} T_j + T_{arbiter\ l} \tag{1}$$

There are two cases:

- Case 1: $\mathrm{mod}\left[\dfrac{NF_j}{MinF_l}\right] = 0$, $NP_l = \left[\dfrac{NF_j}{MinF_l}\right]$ (where mod is the mod function)
- Case 2: $\mathrm{mod}\left[\dfrac{MaxF_j}{MinF_j}\right] \neq 0$, $NP_l = \left\lceil\dfrac{NF_j}{MinF_l}\right\rceil$ (where $\lceil\ \rceil$ denotes the ceil function)

Using equations 2 and 3, we can calculate the time needed to compute frame "j" inside quarter "l", and T_{commj} is the communication time.

$$T_j = T_{pose} + T_{acquire} + T_{struct} + \left[\frac{T_{extract} + T_{track}}{NP_l} + T_{commj}\right] \tag{2}$$

Furthermore, using equations (1) and (2), the total time needed to execute camera "cam k" is given by:

$$T_{cam\ k} = \max\{T_{kl}\} + T_{arbiter\ k} + T_{comml} \tag{3}$$

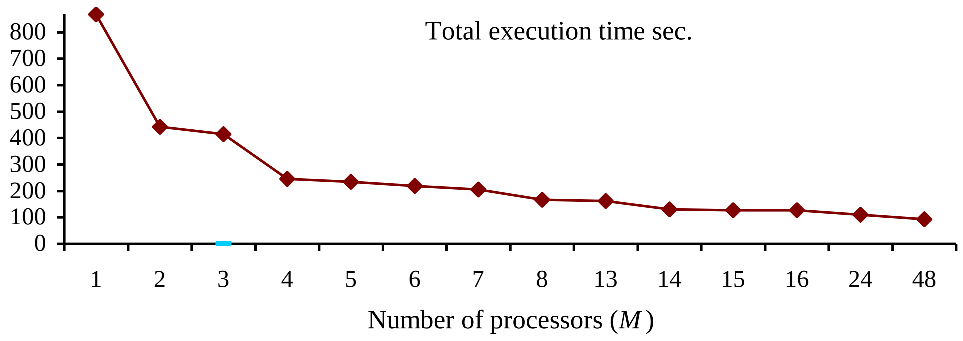
Finally the total execution time needed for the whole system is given by:

$$T_{par} = \max\{T_{cam\ k}\} + T_{commk} + T_{GA} \tag{4}$$

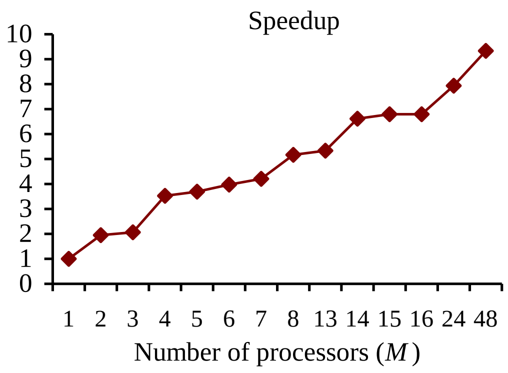
Where T_{commk} is the communication time, and T_{GA} is the time needed for general arbiter. In the next section, we will evaluate the proposed method and show the experimental results.

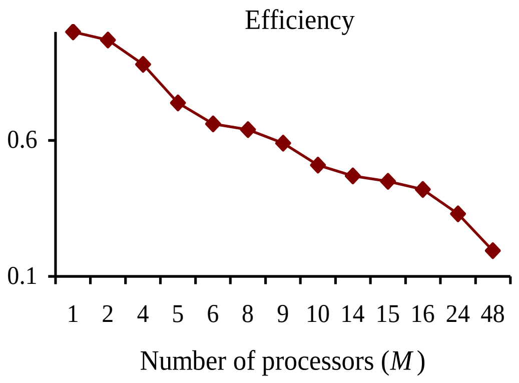
4. Experimental Results and Evaluation

In order to evaluate the system, a publicly available library written in "C" [32] is used. In particular, the library is used to assess the feature selection and tracking algorithm, KLT. In this work, we assume that each processor's speed is 2.5 Ghz, its memory "P_{mem}" equals 1Gbits, and the bandwidth "bg" is 4.2Gbps. Figure 6 shows the analysis of the proposed technique with different number of processors (from 2 to 48). Figure 6(a) illustrates the total parallel execution time. In addition, Figure 6(b) represents the speedup "S_p", which is the ratio between the sequential time and its corresponding parallel time. Moreover, Figure 6(c) illustrates the system efficiency "E_p" which equals S_p/M. Finally, Figure 6(d) represents the degree of system improvement with respect to the sequential version. This is the percentage of improvement in the system performance with respect to the sequential execution is defined by $(T_s\text{-}T_{par})/T_s$.

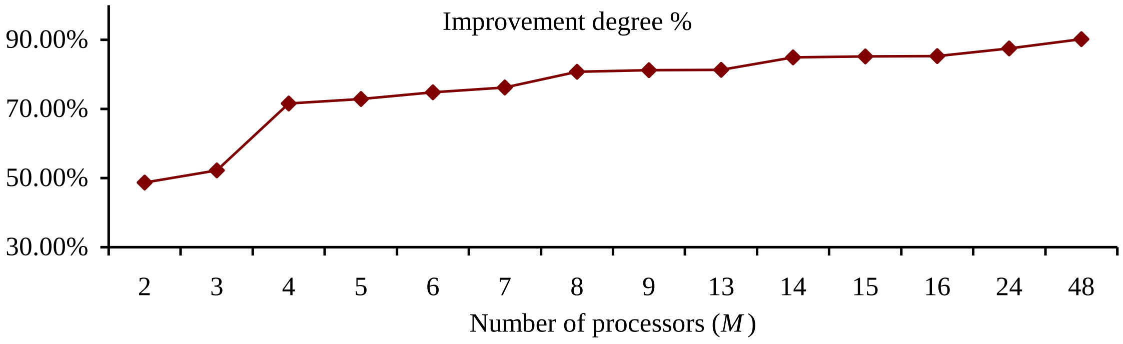


(a): The total execution time in seconds





(b): Speedup (c): Efficiency



(d) Improvement degree compared to sequential performance

Fig. 6. The system performance: Execution time, speedup, efficiency and the improvement degree

From the above figures, we note that:

- Figures 6(a) summarizes the total parallel execution time for different number of processors (from 2 to 48). It is shown that as the number of processors increases, the execution time decreases. The reduction is approximately 50% as the number of processors increases from one to two, while the total execution time decreases by about 85% when the number of processors equals 16. When increasing the number of processors from 16 to 24 the execution time decreases to 13% of the sequential time.

- Upon increasing the number of processors, the speedup increases, and consequently the efficiency decreases as illustrated in Figure 6(b), (c). On the other hand, Figure 6(d) shows the degree of improvement, compared to the sequential performance for *M* =2,4,6,8,10,14,16,24 and 48 the degree of improvement is 48.7%, 71.6%, 74.8%, 80.7%, 80.9%, 84.86, 85%,, 87.5% and 90.5% respectively.
- Increasing the number of processors reduces the total execution time but correspondingly increases the communication overhead and reduces the system efficiency. Using 24 processors leads to a reduction of about 87.5% of the processing time with efficiency equals to 33%. When increasing the number of processors more than 24, they need to exchange more data between them which could lead to the increase of the communication overhead and decrease the system efficiency. Therefore, using 24 processors are sufficient to compromise between system performance and utilization.
- The analysis shows that the use of a multiprocessors system enhances the system performance. In addition, the proposed design is scalable (increasing the number of cameras and the number of processors improves the system performance), this is an important factor in this application where the number of features seen by each camera changes repeatedly.

5. Conclusion

In the present paper, we have solved the robot pose estimation problem using parallel implementation on three levels. The first level of parallelization is the coarse grained level where the parallelization is done in the camera level. On the other hand, in the second level (medium level) of parallelization more than one processor cooperate to compute the work of each camera. Moreover, in the third level (fine grained level) the parallelization is done in the loop/instruction level, where feature extraction and feature tracking tasks are divided into small subtasks and executed in parallel. The analysis shows that the use of a multiprocessor system enhances the system performance. It is obvious that parallel implementation of the pose estimation approach proposed in [1] reduces its computation time by "87%" compared to the non-parallel implementations. In addition, the proposed design is scalable, which is neccesary for this application where the number of features changes repeatedly. The aforementioned results are a strong stimulus for our future hardware implementation of the system.

References

[1] M. E. Ragab and K. H. Wong, "Multiple Non-overlapping Camera Pose Estimation," in *Proc. of the 17th IEEE Int. Conf. on Image Processing (ICIP),* PP: 3253 – 3256, Hong Kong, Sept. 26-29, 2010. Article (CrossRef Link).

[2] T. Bräunl, "Tutorial in Data Parallel Image Processing," *Australian Journal of Intelligent Information Processing Systems (AJIIPS)*, vol. 6, no. 3, pp. 164–174, 2001. Article (CrossRef Link).

[3] Ma. Dongdong, L. Jinzong, Z. Bing, and Z. Fuzhen, "Research on the Architectures of Parallel Image Processing Systems," in *Proc. of the 2nd Int. Symposium on Intelligent*

Information Technology Application (IITA '08), PP: 146 – 150 Shanghai, 20-22 Dec., 2008. Article (CrossRef Link).

[4] J. Greco, "Parallel Image processing and Computer Vision Architecture", Bachelor of Science Summa Cum Laude thesis, Univ. of Florida, USA, 2005. Article (CrossRef Link).

[5] O.C. Jenkins, G. Gonzalez, and M. Loper, "Dynamical motion vocabularies for kinematic tracking and activity recognition," in *Proc. of CVPRW'06. IEEE*, vol. I, pp., 147–152, 2006. Article (CrossRef Link).

[6] L. Rai, S. A. Merritt, and W.E. Higgins, "Real-time image-based guidance method for lung-cancer assessment," *CVPR'06*. *IEEE*, vol. II, pp., 2437–2444, 2006. Article (CrossRef Link).

[7] T. J. Broida, S. Chanrashekhar, and R. Chellappa, "Recursive 3-D Motion Estimation from a Monocular Image Sequence," *IEEE Trans. Aerospace and Electronic Systems*, vol. 26, no. 4, pp.639-656, 1990. Article (CrossRef Link).

[8] A. Azarbayejani, and A.P. Pentland, "Recursive estimation of motion, structure, and focal length," *IEEE Trans. on PAMI*, vol. 17, no. 6, pp.562-575, June 1995. Article (CrossRef Link).

[9] A. Chiuso, P. Favaro, H. Jain, and S. Soatto, "Structure from Motion Causally Integrated Over Time," *PAMI*, vol. 24, no. 4, pp. 523-535, 2002. Article (CrossRef Link).

[10] Y. K. Yu, K.H. Wong, and M. Chang, "Recursive Three-Dimensional Model Reconstruction Based on Kalman Filtering," *IEEE Trans. SMC-B*, vol. 35, no. 3, pp. 587-592, 2005. Article (CrossRef Link).

[11] M.E. Ragab, K.H. Wong, J.Z. Chen, and M.M.Y. Chang, ,"EKF Based Pose Estimation using Two Back-to-Back Stereo Pairs," in *Proc. of ICIP'07*. IEEE, vol. VI, pp., pp. 137–140,2007. Article (CrossRef Link).

[12] Y.K. Yu, K.H.Wong, S.H.Or, and M.M.Chang, "Recursive recovery of position and orientation from stereo image sequences without three-dimentional structure," in *Proc. of CVPR'06*, vol. 1, New Yourk, Jun, 2006. Article (CrossRef Link).

[13] P. Baker, C. Fermuller, Y. Aloimonos, and R. Pless, "A spherical eye from multiple cameras (makes better models of the world)," in *Proc. of CVPR*, vol.1, pp. 576-583, 2001. Article (CrossRef Link).

[14] W. Chang, and C. Chen, "Pose Estimation for Multiple Camera Systems," in *Proc.of ICPR*, vol. 3, pp. 262 – 265, 2004. Article (CrossRef Link).

[15] M. Grossberg, and S. Nayar, "A general imaging model and a method for finding its parameters," in *Proc.of ICCV*, vol. 2, pp. 108-115, 2001. Article (CrossRef Link).

[16] R. Pless, "Using many cameras as one," in *Proc. of CVPR* , vol. 2, pp. 587-593, 2003. Article (CrossRef Link).

[17] D. C. Schurman, and D. W. Capson, "Direct Visual Servoing Using Network-synchronized Cameras and Kalman Filter," in *Proc. of ICRA*, vol. 4, pp. 4191-4197, 2002. Article (CrossRef Link).

[18] V. Lippiello, B. Siciliano and L. Villani, "Position and Orientation Estimation Based on Kalman Filtering of Stereo Images," in *Proc. of CCA*, pp. 702-707, 2001. Article (CrossRef Link).

[19] C. McMillen, K. Stubbs, P. Rybski, S. Stoeter, M. Gini, and N. Papanikolopoulos, "Resource Scheduling and Load Balancing in Distributed Robotic Control Systems," *Robotics and Autonomous Systems*, vol. 44, no. 3-4, pp. 251-259, 2003. Article (CrossRef Link).

[20] N. Ushimi, M. Yamamoto, J. Inoue, T. Sugimoto, M. Araoka, T. Matsuoka, T. Kiriki, Y. Yamaguchi, T. Hasegawa, and A. Mohri, "On-line Navigation of Mobile Robot Among Moving Obstacles Using Ultrasonic Sensors," *Lecture Notes in Computer Science, Robot Soccer World Cup V*, vol. 2377, pp. 477-483, 2002. Article (CrossRef Link).

[21] Pierre Lébraly, Eric Royer, Omar Ait-Aider, and Michel Dhome, "Calibration of Non-Overlapping Cameras- Application to Vision-Based Robotics," *BMVC*, pp: 1-12, 2010. Article (CrossRef Link).

[22] G. Macesau and F. Moldoveanu, "Computer Vision Based Mobile Robot Navigation in Unknown Environments," Bulletin of the Transilvania University of Brasov, Series I: Engineering Sciences, vol. 3, no. 52, 2010. Article (CrossRef Link).

[23] M. Güzel, “Autonomous Vehicle Navigation Using Vision and Mapless Strategies: A Survey,” *Hindawi Publishing Corporation Advances in Mechanical Engineering*, vol. 2013, pp.1-10, 2013. Article (CrossRef Link).

[24] Francisco Bonin-Font, Alberto Ortiz and Gabriel, “Visual Navigation for Mobile Robots: a Survey,” *Journal of Intelligent and Robotic Systems*, vol. 53, Issue 3, pp. 263-296, November 2008. Article (CrossRef Link).

[25] Guilherme N. DeSouza and Avinash C. Kak, “Vision for Mobile Robot Navigation: A Survey,” *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, Feb. 2002. Article (CrossRef Link).

[26] Lionel Heng, Bo Li, and Marc Pollefeys, “CamOdoCal: Automatic Intrinsic and Extrinsic Calibration of a Rig with Multiple Generic Cameras and Odometry,” in *Proc. of IEEE/RSJ Int. Conf.of Intelligent Robots and Systems*, July 18, 2013.

[27] Gim H. Lee, Bo Li, Marc Pollefeys, and Friedrich Fraundorfer, “Minimal Solutions for Pose Estimation of a Multi-Camera System,” in *Proc. of Int. Symposium on Robotics Research (ISRR)*, 2013.

[28] Bo Li, Lionel Heng, Kevin Koeser, and Marc Pollefeys, “A Multiple-Camera System Calibration Toolbox Using a Feature Descriptor-Based Calibration Pattern,” in *Proc. of EEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013. Article (CrossRef Link).

[29] S. Ghiasi, E. Bozorgzadeh, K. Nguyen, and M. Sarrafzadeh, “Profiling Accuracy-Latency Characteristics of Collaborative Object Tracking Applications,” *Journal of VLSI Signal Processing Systems*, vol. 42, Issue 1, pp. 43-55, January 2006. Article (CrossRef Link).

[30] B. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *Proc. of DARPA Image Understanding Workshop*, pp 121–130, April 1981.

[31] J. Shi and C. Tomasi, “Good features to track,” in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition CVPR’94*, Seattle, June 1994. Article (CrossRef Link).

[32] Michael O. McCracken**,** “*Evaluating Performance of Two Implementations of the Shi & Tomasi Feature Tracker*,” *Department of Computer Science and Engineering*, University of California, San Diego, 2008. Article (CrossRef Link).

[33] G. Klein and D.Murray, “Parallel tracking and mapping for small AR workspaces,” in *Proc. of IEEE and ACM Int. Symposium on Mixed and Augmented Reality*, pp. 1–10, Washington, DC, USA, 2007. Article (CrossRef Link).

[34] Jun-Sik Kim, M. Hwangbo, and T. Kanade, “Realtime Affine-photometric KLT Feature Tracker on GPU in CUDA Framework,” in *Proc. of 12th Int. IEEE Conf./ Workshops (ICCV Workshops), on Computer Vision*, pp. 886-893, Kyoto, Sept. 27 - Oct. 4 2009. Article (CrossRef Link).

[35] H. Fassold, J. Rosner, P. Schallauer, and W. Bailer, “Realtime KLT Feature Point Tracking for High Definition Video,” *VaclavSkala and Dietmar Hildebrand, editors, GraVisMa 2009-Computer Graphics, Vision and Mathematics for Scientific Computing*, 2010. Article (CrossRef Link).

[36] S. Sinha, J. Frahm, M. Pollefeys and Y. Genc, “Feature tracking and matching in video using programmable graphics hardware,” *Machine Vision and Applications*, vol. 22, Issue 1, pp 207-217, January 2011. Article (CrossRef Link).

Mohammad Ehab Ragab is an Assistant Professor at the Electronics Research In stitute, Cairo, Egypt. He received the B.Sc., M.Sc., degrees from Ain Shams Univers ity, and Cairo University, Cairo, Egypt, respectively. Then, he obtained his Ph.D. fro m the Chinese University of Hong Kong . His research interests include: Computer vi sion, Robotics and Image Processing.

Ghada Farouk ElKabbany is an Assistant Professor at the Electronics Research Institute, Cairo- Egypt. She received her B.Sc. degree, M.Sc. degree and Ph.D. degree in Electronics and Communications Engineering from the Faculty of Engineering, Cairo University, Egypt. Her research interests include High Performance Computing (HPC), Robotics, and Network Security.