Developing Parallel Application on Multi-core Mobile Phone

DhuhaBasheer Abdullah
Computer Science Dept.
Computer Sciences and Mathematics College
Mosul University
Mosul, Iraq

Mohammed M. Al-Hafidh
Computer Science Dept.
Computer Sciences and Mathematics College
Mosul University
Mosul, Iraq

***Abstract*—One cannot imagine daily life today without mobile devices such as mobile phones or PDAs. They tend to become your mobile computer offering all features one might need on the way. As a result devices are less expensive and include a huge amount of high end technological components. Thus they also become attractive for scientific research. Today multi-core mobile phones are taking all the attention. Relying on the principles of tasks and data parallelism, we propose in this paper a real-time mobile lane departure warning system (M-LDWS) based on a carefully designed parallel programming framework on a quad-core mobile phone, and show how to increase the utilization of processors to achieve improvement on the system's runtime.**

Keywords—Mobile phone; Parallel programming; Multi-core processor; Lane detection

I. INTRODUCTION

Mobility is a key term in the world of today. No matter where you are or what you are doing you are surrounded by a world of mobile devices. No longer are we confined to work at a desktop terminal, we can now work and communicate from virtually any location. This new mode of interaction has been made possible by the advances in the world of miniaturization[1]. We can now work and communicate by using a myriad of devices such as: Laptops, Ultra Mobile PC's, PDA's and mobile phones. Even though we may be surrounded by these devices, the question must be raised: are we using them to their fullest potential? One possible solution to this lies within the area of parallel processing, without realizing the true potential of embedded multi-core architecture, we are not making full use of this now technology[2].

Multi-core architectures for personal computers is an important field in our everyday computing , many frameworks and APIs focusing on parallel programming have been proposed for multi-core processors,that achieve speedup and maximum processor utilization.

However, very little researches have been proposed in the area of multi-core architecture for mobile phones primarilybecause this is a relatively new concept and not many end-products have embraced such architecture. The recent release of multi-core mobile phones optimized for both performance and power consumption, such as Samsung Galaxy SII and SIII[3], has revolutionized mobile computing and opened up the door to new research paradigms, especially for real-time processing.

Now we can consider complex algorithms for potential implementation that previously regarded as impractical for deployment on mobile phones platforms. For instance, performing certain computation on big data matrices on mobile phone was very difficult in the past, due in part to its memory constraints but primarily to the processing power of the mobile phone. It is desirable to be able to use complex algorithms whenever possible because they generally yield more accurate results. Fortunately, the recent release of multi-core mobile phones has empowered us to do exactly that, as true parallelism can now be achieved [4].

II. CONTRIBUTION

In this paper, we consider animplementation oflane departure warning systems, relying on the principles of task and data parallelism, we propose in this paper a parallel programming approach on quad-core mobile devices to detect road lanes and warn the driver if the car is departing the lane, that has the following properities:

1) Implementing parallel algorithm on multi-core mobile phone.

2) Show how to increase the utilization of processors to achieve improvement on the system's runtime.

III. RELATED WORK

On the parallel programming front, making a task parallelizable and run on multiple cores can be a grueling process. Challenging issues include thread synchronization, data race, and starvation.

Many attempts have been made by researchers and programmers alike to design a high-level framework that provides an abstraction layer for programmers to use. Such framework allows the programmers to fully focus on application development without unnecessary worry about parallel programming. The ParLab at Berkeley, UPCRC at Illinois, and the Pervasive Parallel Laboratory at Stanford propose a two-layer framework, consisting of the productivity layer where domain experts, assumed to have limited experience with parallel programming, can focus on application development, and the efficiency layer where computer scientists with strong background in parallel

programming can focus on improving the efficiency of the application [5].

Similar to the aforementioned framework, programming models such as algorithmic skeletons have also been proposed, aiming to benefit from multi-core architectures while decoupling the hassle of thread management from common programming. Skandium and Calcium [6] provide high-level parallel programming libraries based on the thread pool and ExecutorService frameworks in JAVA. Users only need to provide a threshold for threads and a set of initial parameters.

Daniel C. Doolan, and Laurence T. Yang in year 2006. They considered the problem of matrix multiplication to show and demonstrates that mobile devices are capable of parallel computation using Mobile Message Passing Interface (MMPI). MMPI allows parallel programming of mobile devices over a Bluetooth network [7].

PanyaChanawangsa, and Chang Wen Chen in year 2012. They demonstrate how proper utilization of a dual-core mobile processer can achieve tremendous speedup in mobile application [4].

Massimo Bertozzi, and Alberto Broggi in year 1998.They describes the Generic Obstacle and Lane Detection system (GOLD), a stereo vision-based hardware and software architecture to be used on moving vehicles to increment road safety [8].

Mars Lan, MahsanRofouei, Stefano Soatto and Majid Sarrafzadeh in year 2009. They built a SmartLDWS, that employs a novel lane detection algorithm that is both robust and scalable to overcome poor camera quality and limited processing power faced by most smartphones [1].

IV. PARALLEL PROCESSING

Parallel Processing refers to the concept of speeding-up the execution of a program by dividing it into multiple fragments that can execute simultaneously, each on its own processor. A program being executed across ***N*** processors might execute n times faster than it would using a single processor [9].

A. The benefit of using Parallel Processing

In the earliest computers, only one program ran at a time. A computation-intensive program that took one hour to run and a tape-copying program that took one hour to run would take a total of two hours to complete their task. An early form of parallel processing allowed the execution of both programs simultaneously. The computer would start an input/output operation, and while it was waiting for the operation to complete, it would execute the processor-intensive program. The total execution time for the two jobs would be a little over an hour [10][11].

V. DEPENDENCIES

Understanding data dependencies is fundamental in implementing parallel algorithms. No program can run more quickly than the longest chain of dependent calculations (known as the critical path), since calculations that depend upon prior calculations in the chain must be executed in order. However, most algorithms do not consist of just a long chain of dependent calculations. There are usually opportunities to execute independent calculations in parallel.

Let Pi and Pj be two program fragments. Bernstein's conditions describe when the two are independent and can be executed in parallel. For Pi, let Ii be all of the input variables and O_i the output variables, and likewise for P_j. P_i and P_j are independent if they satisfy:

- $Ij \cap Oi = \Phi$
- $Ii \cap Oj = \Phi$
- $Oi \cap Oj = \Phi$

Violation of the first condition introduces a flow dependency, corresponding to the first statement producing a result used by the second statement. The second condition represents an anti-dependency, when the second statement (P_j) would overwrite a variable needed by the first expression (P_i). The third and final condition represents an output dependency: When two statements write to the same location, the final result must come from the logically last executed statement[12][13].

VI. SPEEDUP

The speedup of code explains how much performance gain is achieved by running our program in parallel on multiple processors.A simple definition is that it is the length of time it takes a program to run on a single processor, divided by the time it takes to run on a multiple processors.Speedup generally ranges between 0 and **p**, where **p** is the number of processors [14]. Speedup is defined by the following formula:

$$\text{Total Speedup} = T_s / T_p \quad (1)$$

T_s : is the runtime without parallelism.

T_p : is the runtime with parallelism.

VII. MULTITHREADING AND PROCESSOR UTILIZATION

Since a mobile phone is considered a general-purpose device, application-level parallelism is the best we can achieve. Without explicitly using multiple threads, speedup from a multi-core architecture will not be obvious. In this paper, we propose a general guideline for breaking down a global task into multiple subtasks and later demonstrate how to apply this idea on a mobile lane detection system.

The first step towards parallelizing a task is to determine the optimal number of threads to use. Limiting thread contention is crucial for application speedup. Spawning too many threads than necessary not only disrupts other applications, but may also result in a longer execution time of the application due to the overhead associated with context-switching. A processor core can handle only one thread at a time. For efficiency purposes, a simple rule is to spawn as many threads as the number of cores available, thereby delegating one thread to each core and eliminating the need for time-slicing. A simulation was conducted by spawning different numbers of threads to execute certain tasks. A

dramatic improvement in execution time can be seen when we increase the number of worker threads from 1 to4. However, since there are only four available cores, increasing the number of threads do not enhance but aggravates the performance, resulting in a slightly longer execution time[15][16].

VIII. Proposed Mobile Lane Detection System

Transportation safety is an issue of ever increasing concern. In the U.S. alone, more than 30,000 casualties suffered from traffic crashes in 2009, according to the National Highway Traffic Safety Administration. A considerable number of car accidents occurred as a consequence of the drivers failing to keep the vehicles within the designated lane. In the wake of such tragedies, attempts to incorporate an Intelligent Vehicle Assistant System into automobiles have been made by adopting various computer vision approaches.

While LDWS has been installed on many trucks and othercommercial vehicles in both Europe and North America and has shown to significantly reduce preventable accidents, it still often remains as an option even for the luxury passenger vehicles. Cost is often cited as one of the main reasons impeding a wide-spread adoption of LDWS. On the other hand, the professional skill required to install such a system, to calibrate the camera, and to integrate it into vehicle's electronics should not be overlooked, either[1].

In order to bring LDWS to the mainstream market, we propose Mobile-LDWS,to make the system available for everyone. The system acquires images of the road and intelligentlylabels the lane marks. The heading of the vehicle as well as its position with respect to the road boundaries can thus be determined.If the system believes the vehicle is about to depart from the current lane, it triggers an alarm to alert the driver.Other applications of lane detection include autonomous driving for cruise control as well as robot navigation.

A. Development Platform

The system was developed on Android 4.1.2 (Jelly Bean). Released in September 2012. The phone's most outstanding feature is its processor – a superscalar quad-core 1.4 GHz Arm Cortex-A9 with 2 GB RAM [3]. Optimized for high performance and low power consumption, the Galaxy SIII is indeed an ideal platform for this system.

B. Data Acquisition

The proposed lane detection system uses the phone camera as the means of sensing. It captures raw image frames at the rate of 30 frames per second (fps) and feeds them to the preprocessing module. The phone is attached onto the car windshield by an off-the shelf GPS mount. It should be placed in such a way that the camera is able to see the road clearly while precaution should be taken during this step so that the device does not obstruct the driver's view (Figure 1).

Fig. 1. The M-LDWS on a Samsung GalaxySIII in action on an off-the-shelf GPS mount.

Android's camera API allows programmers to process the preview frames directly. This feature enables us to overcome the storage constraint, as we can process those frames and display the detected lane boundaries on the phone screen without having to record any of them. The frame data come in the form of byte arrays whose default format is YCbCr, to get the three base colors (read , green , blue), we convert the frame format from YCbCr to ARGB8888. Finally, each pixel values are stored in an image matrix. Now we can implement the basic matrix and linear algebra operations.

C. Lane Detection

The lane detection algorithm is based on the assumptions that the lanes are defined by a clearly painted white line.

To detect a line we first must find out the most important features of it, which are color and shape. We first used the color feature to detect a line. Since the color of the road lane is white, we perform an operation to extract white color and change it to green, based on the following:

$$g(x,y) = \begin{cases} 255, & \text{if } r(x,y) > val,\ g(x,y) > val,\ b(x,y) > val \\ 0, & \text{otherwise} \end{cases}$$

Where :
r(x,y) is the red value of the pixel in the position (x,y).
g(x,y) is the green value of the pixel in the position (x,y).
b(x,y) is the blue value of the pixel in the position (x,y).
val is a predefined value.

In the next step, the shape feature will be used to detect the lines, and perform a vertical edge detection filter to extract the lines (Figure 2).

D. Lane Departure Warning

The next step is to find out if the vehicle is in the lane or it is about to depart it. If the system believes the vehicle is about to depart from the current lane, it triggers an alarm to alert the driver. This can be done by scanning a specific region in the image for any lines that have been detected.

If the result of the detection process is true, that means that the vehicle is departing the current lane so the system alert the driver.

E. Proposed Parallel Framework

The next step is to determine if data parallelism is possible and appropriate. For many image processing tasks, data comes in the form of image matrices, or on the lower level 2-dimensional arrays. In many cases, they can be split up into smaller independent chunks and processed concurrently, reducing the execution time while producing the same output as when processed sequentially. For the image edge detection, we can split up the image matrix into smaller sub-matrices (matrix slicing) and slide an edge detection convolution mask over each sub-matrix concurrently. However, if the size of data to be processed is not significantly large, employing data parallelism will not yield much speedup as a result of thread overhead. In the case of this application, since the image are large, data parallelism is well worth a try. Generally speaking, given k processor cores, we should divide the input data of size n into n/k smaller chunks anddistribute them across k cores with each core running a single thread. Figure (3).

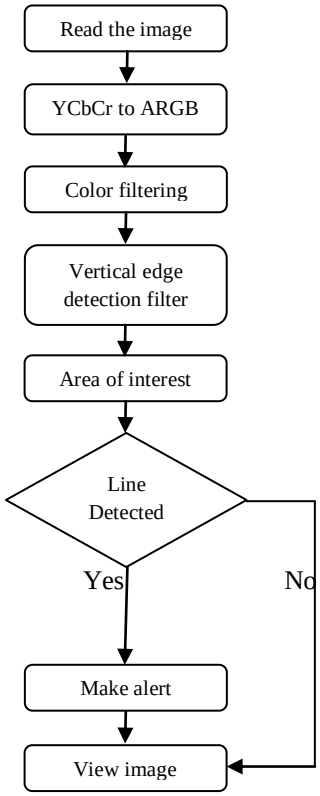

Fig. 2. The proposed serial lane detection system

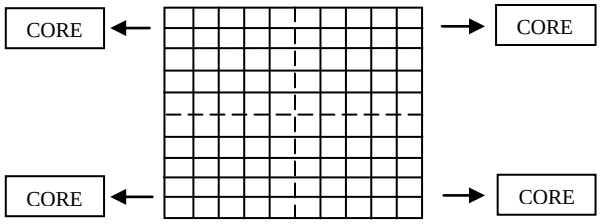

Fig. 3. Matrix slicing

The parallel computation requires several steps that are not required in the sequential version of the application (Figure4). One of the first main differences is that we need to determine the number of threads suitable for the phone hardware. Since Samsung Galaxy SIII has four cores, four threads are ideal for execution the task in hand. So that the image must be split in to four same size blocks called sub-image, then send each sub-image to one of the four cores we have. Once this operation has been completed each core can compute its own section of the image. The final stage is to gather all the results back in to one image. The result of the computation can then be available for the next steps.

To gain full control of task management, we also make use of theFutureTask class , allowing to track the progress of the submitted tasks and block until all of them have been completed. The four tasks are eventually submitted to an ExecutorService, which takes care of thread pool creation and assigns a submitted task to an available thread. More importantly, by using the ExecutorService, memory consistency is guaranteed, thus eliminating the trouble of thread synchronization.

IX. EXPERIMENTAL RESULTS

The system tested on image size (320×240), The results are shown in (Table (1)). Time results represents the average runtime of hundred frames.

TABLE I. EXPERIMENTAL **RESULTS**

Method	Run time
Serial	11043ms
Parallel	6506ms

From the time results in (Table (I)), we can calculate the speedup as below:

Total speedup = 11043/6506=1.69

This means that the reduction in the overall processing time is 41.08%.Figure (5) shows lane detection results.

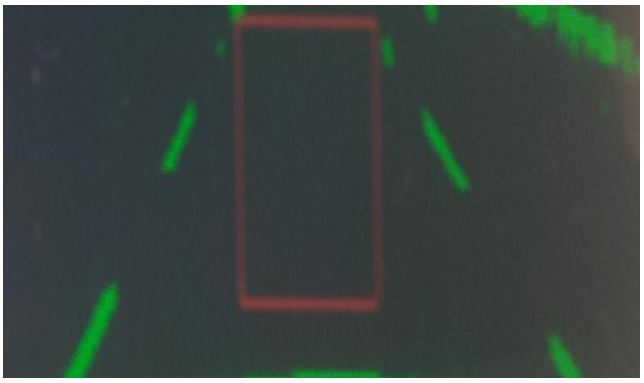

Fig. 5 Lane detection results

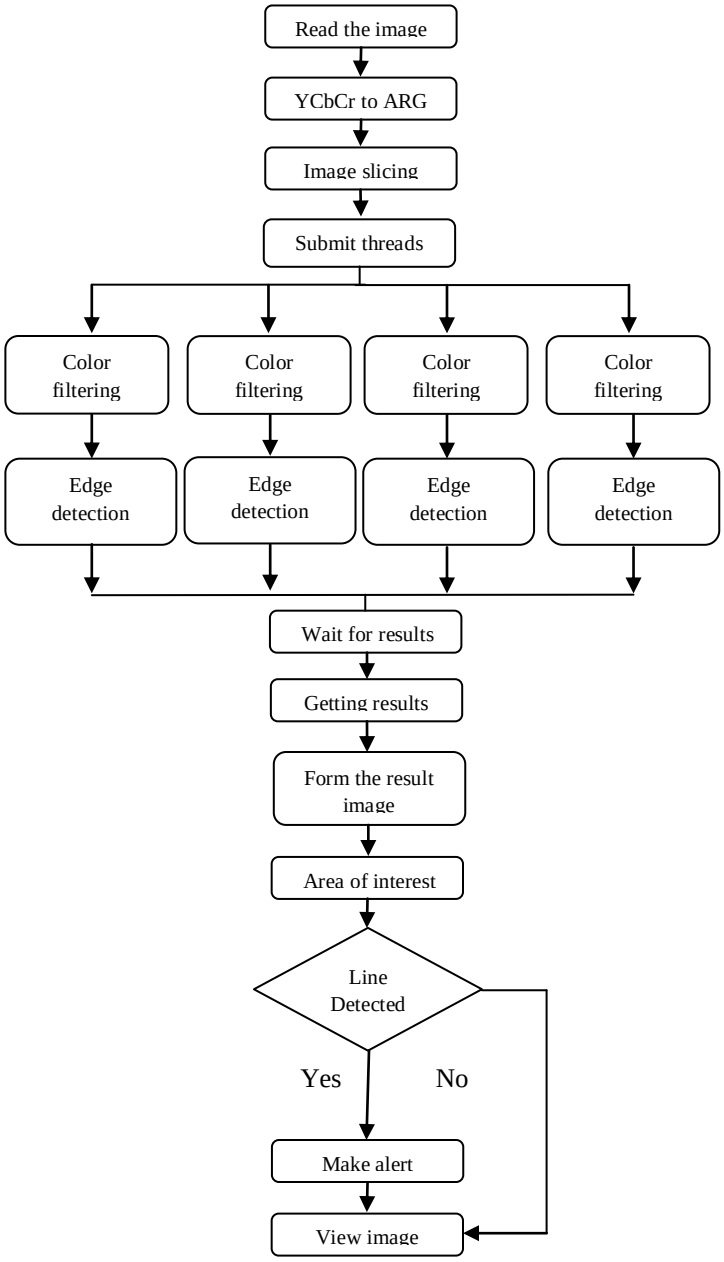

Fig. 4 Modified lane detection system with the proposed parallel framework

X. CONCLUSIONS

In this paper, we have demonstrated how to achieve speedup in a mobile lane detection system written entirely in JAVA by using the proposed parallel programming approach, based on the idea of task and data parallelism. Running on a quad-core Samsung Galaxy SIII. The proposed system shows significant reduction in the overall processing time and good speedup.

REFERENCES

[1] M. Lan, M. Rofouei, S. Soatto, M. Sarrafzadeh," SmartLDWS: A Robust and Scalable Lane Departure Warning System for the Smartphones", Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems, St. Louis, MO, USA, October 3-7, 2009.

[2] D. Doolan, S. Tabirca, L. Yang," MMPI a Message Passing Interface for the Mobile Environment", Proceedings of MoMM2008, Linz, Austria, 2008.

[3] Samsung I9305 Galaxy S III Full Specifications, http://www.gsmarena.com/samsung_i9305_galaxy_s_iii-5001.php

[4] P. Chanawangsa, C. Chen, "A New Smartphone Lane Detection System: Realizing TruePotential of Multi-core Mobile Devices", MoVid'12, 2012, pp.19-24.

[5] B. Catanzaro, et al., "Ubiquitous Parallel Computing form Berkeley, Illinois, and Stanford", IEEE Computer Society, 2010, pp. 41-55.

[6] T. Panagiotis, "Evaluating Skandium's Divide-and-Conquer Skeleton", Master Thesis, School of Information, University of Edinburgh, 2010.

[7] T. Yang, D. Doolan, "Mobile Parallel computing", Proceedings of The Fifth International Symposium on Parallel and Distributed Computing, IEEE International,2006.

[8] M. Bertozzi, A. Broggi, "GOLD: A Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection", IEEE Transaction on image processing, VOL.7,NO. 1, JANUARY 1998.

[9] J. Nancy, J. Richard, A. James, "PARALLEL PROCESSING: THE NEXT GENERATION OF COMPUTERS", National Energy Technology Laboratory,2011.

[10] M. Sasikumar, D. Shikhare, P. Prakash, Introduction To Parallel Processing, Prentice-Hall of India Private Limited, 2006.

[11] C. Evangelinos, C. Hill," Cloud Computing for parallel Scientific HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazons EC2.", ratio, vol. 2, no. 2.40, pp. 2–34, 2008.

[12] J. Hennessy, D. Patterson,"Computer Architecture: A Quantitative Approach", Morgan Kaufmann Publishers, SF, CA, 1996.

[13] H. Dietz, "Linux Parallel Processing HOWTO", v980105, 5 January 1998.

[14] D. Marshall, Parallel Programming with Microsoft Visual Studio, Microsoft Corporation by: O'Reilly Media, 2011.

[15] D. Abdullah and M. Al-Hafidh, " The True Powers of Multi-core Smartphone", IJCSI, , Vol. 10, Issue 4, No 2, July 2013.

[16] H. Guihot, Pro Android Apps Performance Optimization, Apress, 2012.