5-1-2016

Parallel Static Object Detection

Tirambad Shiwakoti
University of Nevada, Las Vegas, shiwakot@unlv.nevada.edu

PARALLEL STATIC OBJECT DETECTION

By

Tirambad Shiwakoti

Bachelor of Computer Engineering

Tribhuvan University
Institute of Engineering, Pulchowk Campus, Nepal
2011

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2016

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

April 22, 2016

This thesis prepared by

Tirambad Shiwakoti

entitled

Parallel Static Object Detection

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Ajoy K. Datta, Ph.D.
Examination Committee Chair

John Minor, Ph.D..
Examination Committee Member

Yoohwan Kim, Ph.D.
Examination Committee Member

Venkatesan Muthukumar, Ph.D.
Graduate College Faculty Representative

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Interim Dean

Abstract

The need for parallelism is growing with the broadening of computing in the real world where computing is an integral part of any field. In the early days of computing, adding transistors to the CPU could solve computation complexity. This is not the case now, where we can no longer advance the hardware capabilities at the pace of the advancement of computing problems. One of the fields which is intensive in computation is image processing. If it were just for one frame of an image, we could cope with the computation overhead. When the need is to compute video frames, in some cases real-time video analysis, sequential execution of each frame could delay the result. In this thesis, we propose a parallel implementation of computing video frames. In particular, we focus on detecting new static objects that arrive in the already defined static background. This has practical implications as well. In a traffic crossing which is prone to accidents, this can be used to detect a vehicle or person in distress. The sequential implementation of this is fairly simple. However, as this is a computation-intensive problem, it would be more efficient to design a parallel solution.

Acknowledgements

I would like to express my sincere gratitude to my supervisor *Dr. Ajoy K Datta* for providing me with all the guidance, motivation, and supervision to complete my thesis and other academic objectives.

I am also grateful to my thesis committee members *Dr. Yoohwan Kim, Dr. John Minor and Dr.Venkatesan Muthukumar.* I would also like to offer my gratitude to my parents, my brothers and sisters for their love and support. I am extremely thankful for the support of my lovely wife Dinu.

Finally, I would like to thank all my friends, seniors, and juniors for providing me all the help they can to complete this thesis.

Tirambad Shiwakoti

University of Nevada, Las Vegas

May 2016

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

With the advancement in computing, today's computers are able to solve the much more complex problems. But with the advancement in hardware architecture, the problems are also becoming much more complex. The complexity of these problems can be defined from two perspectives: size and computation complexity. Some problems are complex to solve because of the sheer size of the data to be processed. Some problems are complex due to the higher computation complexity required for the solution.

These problems can be solved either through sequential or parallel programming. But as the complexity increases, the sequential approach becomes much more unrealistic because we cannot make a single processor so much faster to catch up with the complexity. So we have to use parallel programming to solve these complex problems. The problem using the parallel programming approach is to identify the part of the program which can be parallelized. It turns out that in a general program, the proportion of the code which can be parallelized is low compared to the sequential part of the program.

One of the fields which can use parallel programming to solve its problem is the field of image processing. If we focus on an application which uses only single image(frame) processing, there is no point of applying parallel programming to solve the problem, because the communication overhead will have greater effect when we try to process only a single image. Whereas if we are trying to process a stream of frames, like in analyzing video, the processing of different frames on different nodes of a multiprocessor system will become beneficial.The key in this approach is to use an algorithm which can be parallelized over different nodes. Most of the image processing algorithms which uses adaptive models are difficult to parallelize. For example, Adaptive background mixture models for real-time tracking [2] models the value of the particular pixel as a mixture of Gaussian. If we were to implement a parallel version of this algorithm then we would not be able to distribute incoming frames from the video to different nodes since each pixel requires the history from the previous frame for the proper modeling. So, we have to identify an algorithm which can be parallelized over different nodes in the system.

1.2 Objective

The purpose of parallel programming is to identify the part of the program that can be parallelized and to share the computation of the parallel execution over different nodes in the network. There are various ways of designing a parallel solution. Most of the parallel algorithms are designed from the sequential version with some modification to suit the parallel hardware. In this thesis, we are proposing a parallel algorithm to identify new static objects in an already defined background. There are various factors to consider before designing a parallel solution. We have to design our solution in such a way that there is less communication overhead. The ideal parallel solution would break the computation, distribute to different nodes, compute in different nodes and aggregate the result in a result node.

Nowadays, we have multiprocessor machines which have many processors working together using shared memory. In addition to it, there are also multicore machines where each processor has multiple cores. With the correct implementation, parallelism can be achieved in multicore and multiprocessor machines. For example, concurrent threads can be run on different cores and each thread can compute different computation so that parallelism can be achieved with the use of concurrency. As we know that each machine is limited in its number of processors and number of cores, the ideal parallel solution would be the utilization of many machines over a network. This solution has a drawback as well. Since we are using many machines we have to establish a communication protocol between the machines. This will give rise to a communication cost. So the ideal solution will find a trade-off between communication cost and computation gain.

In our research, we are achieving parallelism by using multiple nodes and using a communication protocol to communicate data between different nodes in the system. The core of our solution is to send the incoming frames to different nodes where each node computes on each frame and sends the result to a node which collects the result frames for the user to see the result. While communicating and computing the frames we have designed an algorithm which identifies the first frame as the static background. The static background is passed to all the nodes for reference. When a node receives a new frame, it compares the new frame with the reference frame to identify any new object in the frame. Two consecutive results are then compared in each node to identify any static object that has appeared in the frame. We will present a detailed explanation in the methodologies chapter.

1.3 Outline

In chapter 1, we discussed the area of the research. We gave a brief overview of various aspects of the research and the motivation for this research

In chapter 2, we will discuss the different types of parallelism.We will discuss similarities and differences between parallel programming, concurrency, and distributed computing. We will also discuss the various algorithms for motion detection in image processing. Our major effort would be to discuss the incorporation of image processing algorithms with parallel programming. We will discuss the details on drawbacks and limitations when using parallelism with different image processing algorithms.

In chapter 3, we will discuss various areas of research that are being conducted in the field of parallel programming and distributed computing. We will particularly focus on research which uses the traditional sequential algorithm and implements a parallel version of the algorithm. We will also discuss research in the field of image processing and motion detection.

In chapter 4, we will discuss our approach to the parallel programming and image processing solution. We will discuss in detail how our parallel algorithm works. We will also discuss the implementation of the algorithm. We will discuss the results obtained and analyze the result from the theoretical perspectives of parallel programming.

In chapter 5, we will discuss the results obtained from our algorithm and analyze them.

In chapter 6, we will provide the conclusion of our research and drawbacks and limitations of our research. We will also discuss different avenues which can be explored in the future in this area of research.

Chapter 2

Background

2.1 Parallelism vs Concurrency vs Distributed Systems

Modern computers are equipped with many processors running under the same system over a shared memory. Each processor in this multiprocessor system can also have multiple cores in them. With this advancement of technology, the concept of parallelism, concurrency, and distributed systems are somewhat merged. But there are some underlying differences. Parallel Programming is the execution of many processes simultaneously whereas concurrency is a system where many processes/threads execute independently. Parallel programming may or may not use shared memory, whereas distributed systems generally consist of independent sets of computers with their own local memory communicating with each other using message passing. These differences are discussed in detail in the following sections.

2.1.1 Parallelism

Parallelism is the simultaneous execution of multiple processes. As problems become more complex, it will take more time for a single processor to compute the problem. One of the ways of increasing computational speed is by using multiple processors operating on a single problem. While doing so, the complex problem is split into parts and each part is processed in a separate processor in parallel [3, p. 5]. For a multiprocessor system to work they need to communicate with each other. This is done either through shared memory or through message passing.

Shared Memory Multiprocessor System

The most natural extension of a single processor system to a multiprocessor system is to add a number of processors which work over a shared memory. In this system, multiple processors are connected to multiple memory modules such that each processor can access any memory module in a shared memory configuration [3, p. 6].

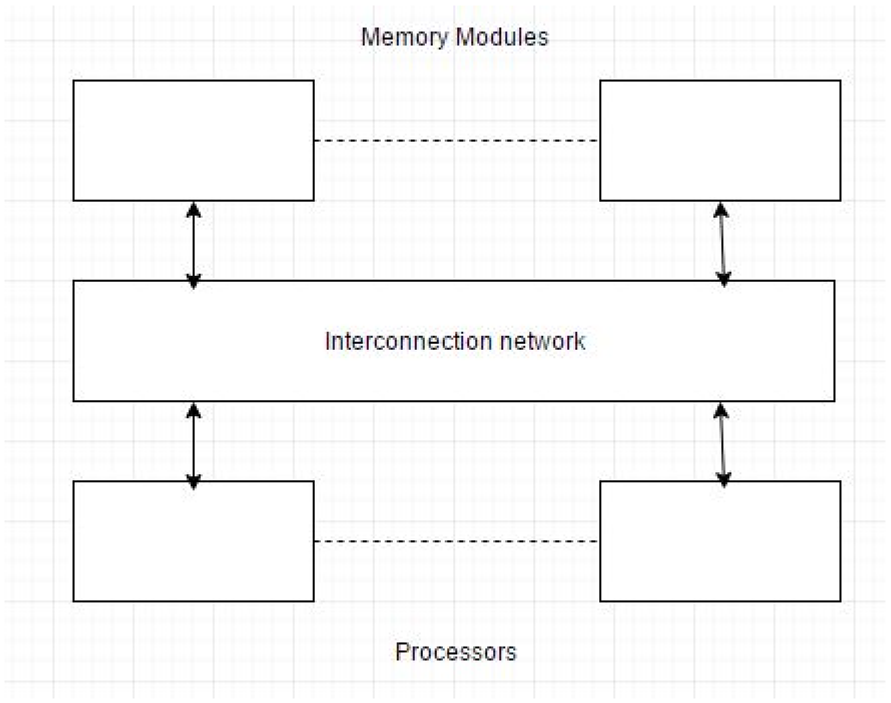

Figure 2.1: Shared Memory Multiprocessor Model

Message Passing Multicomputer

An alternative to a shared memory multiprocessor system is the message passing multiprocessor system where multiple computers, i.e, processors with their own local memory, are connected by an interconnection network. In this type of system, communication between processors is done through the interconnection network where message are passed between computers using message passing protocols [3, p. 8].

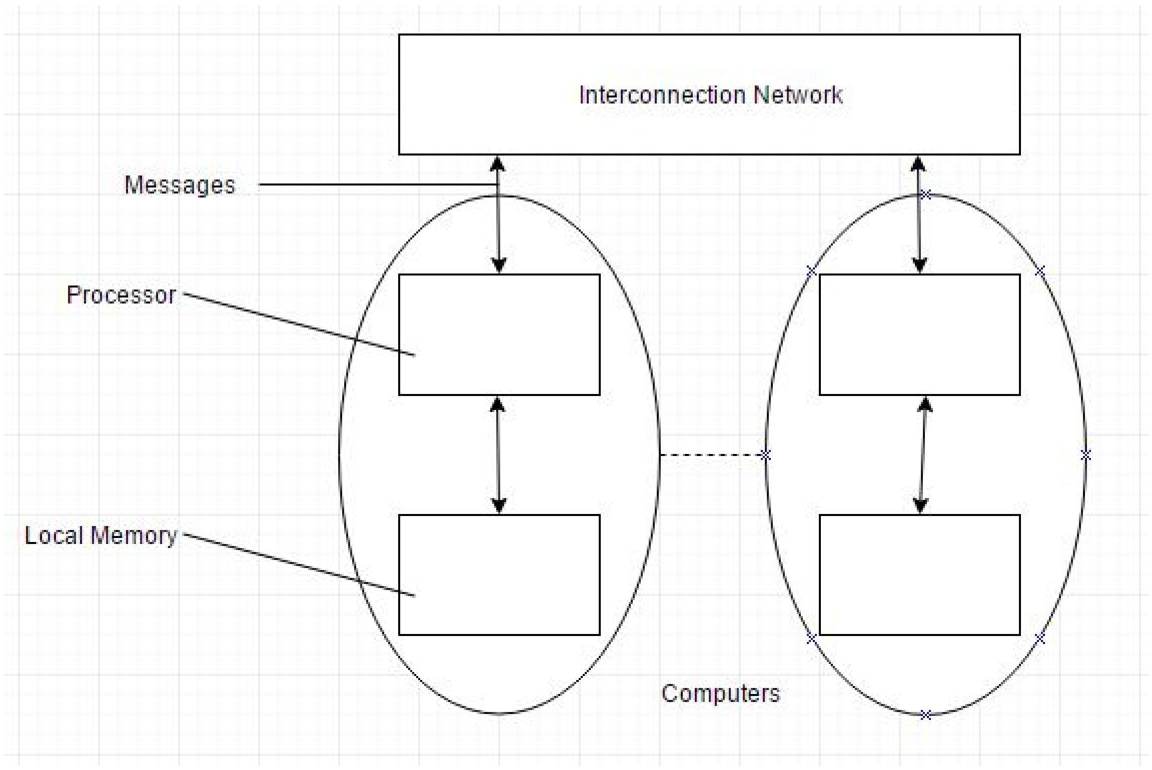

Figure 2.2: Message Passing Multiprocessor Model

2.1.2 Concurrency

Concurrency is the composition of independently executing processes. Concurrency gives a way to structure a program into independent pieces, and we have to find a way to coordinate these independent pieces using different communication methods. Concurrency does not mean that different processes/threads run simultaneously as in parallelism. The execution of these processes/threads is independent to one another. So, if we want to coordinate these executions we have to establish modes of communication between these processes/threads. Most of the programming languages have their own ways of synchronization for concurrency.

2.1.3 Distributed Systems

A distributed system is a system where many independent computer systems with their own sets of processors and memory work together to solve a complex/big task by splitting the task [4] . Technically, a distributed system is the same as a message passing multiple processor system where work is split into many tasks for each computer and data/messages are communicated between these nodes/computers using a messaging protocol. Open MPI(Message Passing Interface) is one of the popular Message Passing API developed for communication between computers in parallel and/or distributed computing.

2.2 Computer Vision and Image Processing

Computer vision in normal understanding is the study of images. The main goal of computer vision is to extract information from images. Images, in terms of data structures, can be represented by an array where elements of the array represent corresponding pixels from the image, and the value in the array represents the corresponding value of the pixel from the image. After we quantify the image as a set of pixels values, then we can perform different analysis on these values to obtain meaningful information from the image. In our research, we are focusing on identifying static objects in an already defined static background. This requires the study of the video. Videos are studied as a sequence of image frames in computer vision. To identify new objects in the frame, we need a reference from the earlier frames. There are various ways of analyzing video frames. We have to note that we will select or design only those algorithms which can be parallelised. In this section, we will discuss some of the areas of image processing which are particularly related to our solution.

2.2.1 Background Subtraction

Background Subtraction is widely used to detect moving objects in a frame from a static camera. The fundamental basis of background subtraction is to get the difference between the current frame and the reference frame. The reference frame is called the background frame [5].The background subtraction method has to continually adjust the background due to various factors such as the change in light intensity, an addition of new objects in the frame etc. There are various approaches for doing background subtraction which will be discussed in the literature review section.

2.2.2 Frame Difference

Frame difference is one of the simplest ways of calculating the difference between the frames. Using this approach, if we want to check whether there is a new object in the frame or not, we just check the difference between the new frame and the reference frame. The main drawback of this approach is that the reference frame is not adjusted with the change in various parameters as in the background subtraction method. But there is one fundamental advantage when we parallelise this solution. Since there is only one reference frame, we can compute the difference of incoming frames simultaneously in parallel nodes.

Chapter 3

Literature Review

There is a lot of research being carried out in the field of parallel programming/distributed computing. There is also much research done on the field of image processing, particularly in motion detection which is our focus. In this section, first we will discuss research on parallel programming, and on how different sequential algorithms are being parallelised. Then secondly, we will discuss the research done/being done in the field of image processing and motion detection.

3.1 Distributed Systems Applications

Most of the distributed systems applications are based on huge data sets. With the increase in data size, the usage of mining these big data sets and extracting the information is ever increasing. [6] There were various problems in traditional centralized data warehouse systems used for centralized data mining. For example, we have to regularly upload critical data in the center. Due to the centralized system, there is a long response time. Due to this, a distributed data mining system is much more effective. The below diagram displays the distributed data mining architecture. [6]

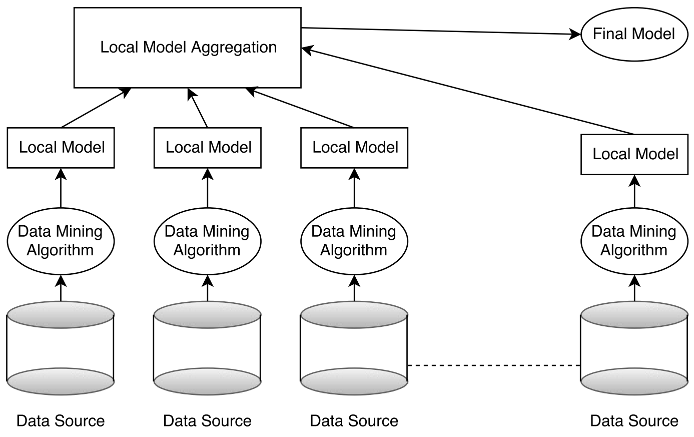

Figure 3.1: Distributed Data Mining Framework

We can understand from the figure that a true data mining application utilizes the distributed system in full. Here, local data sources are mined locally using a data mining algorithm to create a local model. Then these local models are combined using aggregation to create a final model. This approach is beneficial mainly in two contexts. The obvious one is that since a distributed system is used, it is much faster than the traditional data warehousing architecture. The other one is that since there is no center collection of data, privacy of the data remains intact.

3.2 Data Clustering and Parallel Algorithm

Mostly parallel algorithms are used in the cases where there are huge data sets involved. Sometimes even with small data sets, if the solution of the problem requires iteration, then the use of parallel algorithms will reduce the time complexity of the problem. One such algorithm which requires huge data sets to compute is k-means algorithm. [7] In the K means algorithm we consider a set of n points X_1, X_2, X_3, ... X_n. The objective of this algorithm is to find a centroid point m. The following method is used to solve this problem classically [8]

1. Start with k starting points.
2. Compute the Euclidean distance from all points to the k points. Then find the closest cluster centroid among the k centroids.
3. Recompute centroid taking the average of the points associated with a centroid in step 2.
4. Repeat 2,3 until a point of convergence is met.

Using the parallel programming approach, the set of n data is broken into p equal parts where p is the total number of processors. The set of k starting points is available to all processors. Now, each processor calculates the euclidean distance between all the points it has with all the k set of points. But there is the extra overhead of communication, when processors have to communicate the set of points associated with a particular centroid point. In this way, the sequential algorithm is parallelised into a parallel solution.

3.3 Using MPI for Parallel Programming

Message Passing Interface (MPI) is a library specification which is used extensively for data communication between independent processes [1]. MPI was designed exclusively for parallel hardware. As we discussed in 2.1.1, a message-passing multicomputer consists of independent processors with their own local memory communicating with each other using message passing protocol. MPI is a library defining the protocol for these systems and consists of various functionality to communicate between processes. Some of the functionality which is used frequently is discussed in the table below.

Functions	Usage
MPI_Init()	Initialize the MPI execution environment
MPI_Comm_size()	Returns the number of processors
MPI_Comm_size()	Returns the identifier for the process running (0 to Number of Processors - 1)
MPI_Send()	Send Data from one processor(SOURCE) to another processor(DESTINATION)
MPI_Recv()	Receive Data coming from the source
MPI_Status()	A data structure send in the MPI_Send() Packets which contains meta information
MPI_Finalize()	Terminate the MPI execution environment

Table 3.1: Conceptual Syntax of frequently used functionality in MPI. For detailed usage, see [1]

3.4 Motion Detection and Background Subtraction

As discussed in 2.2.1, Background Subtraction is a widely-used method for detecting moving objects inside a frame from a static camera [5]. The Background subtraction method can be divided into following points.

1. Declare a background. This is the initial static frame which serves as the initial background.
2. Apply a motion detection algorithm to find any new objects in the frame.
3. Update the background to adapt to the variance luminance conditions and geometry settings.

There are several methods proposed for performing the background subtraction method. Some of them are:

3.4.1 Frame Difference

Frame difference is the technique of finding new objects in the frame by checking the difference between the new frame and the reference frame. This is basically done by comparing the pixel values of the new frame with the old frame. If change is seen in the new pixel, then it implies that a new object has appeared in the frame. This is a primitive image process algorithm for motion detection. The main drawback of this algorithm is in the adjustment of the background frame or the reference frame. This drawback can be somewhat minimized by the use of threshold values. In this approach, the change in pixel value up to the threshold is not considered as the appearance of a new object in the frame. The new object is considered only when the pixel difference exceeds the threshold value.

3.4.2 Motion tracking using Gaussian Distribution

As discussed in 3.4, most of the background subtraction methods are adaptive in nature which modifies the background with the change in environment. Most of these algorithms are not robust in the case where there can be many moving objects in the frame, particularly if they move very slowly. *Chris Stauffer* and *W.E.L Grimson* have proposed a more robust algorithm in their paper *Adaptive background mixture models for real-time tracking* [2]. Most of the adaptive models value all the pixels as one particular distribution. Instead, it will be more robust if pixel-specific modeling is done. The gaussian of the pixels, which corresponds to the background color, is considered the background and those which do not correspond are considered to be foreground.

Chapter 4

Methodologies

The main objective of our research is to find new static objects in an already defined background using parallel programming. In this chapter, we will discuss the idea behind our proposed solution. We will also discuss our algorithm including pseudo code. We will also discuss the tools used for the implementation of our algorithm.

4.1 Parallel Static Object Detection : Algorithm Design

The problem statement of our research is: *find a solution to finding any new static objects that appear in a defined background using parallel programming.* While designing any parallel algorithm, we have to understand that solving any problem through parallelism must follow the fundamental rule of divide and conquer. So, we have to design our algorithm in such a way that there will be a heavily computation task that can be divided between multiple nodes. Our algorithm can be summarized in the following points:

1. The Master node collects video frames from a source (video file or a camera).
2. The Master node then sends a background frame to all worker nodes for reference.
3. Worker nodes store background frames and all its grayscale, blue, green and red channel frames.
4. The Master Node continuously sends new frames to the worker nodes .
5. Worker nodes receive new frames and break them into channels like the background frames. It computes the frame difference between two successive new frames and the background frame to identify any new static pixels in the frame.
6. Worker nodes send results to their respective collector.
7. The Collector collects their respective result frames and writes them to a video file.
8. Users can see five different results for five types of videos of the same frame.

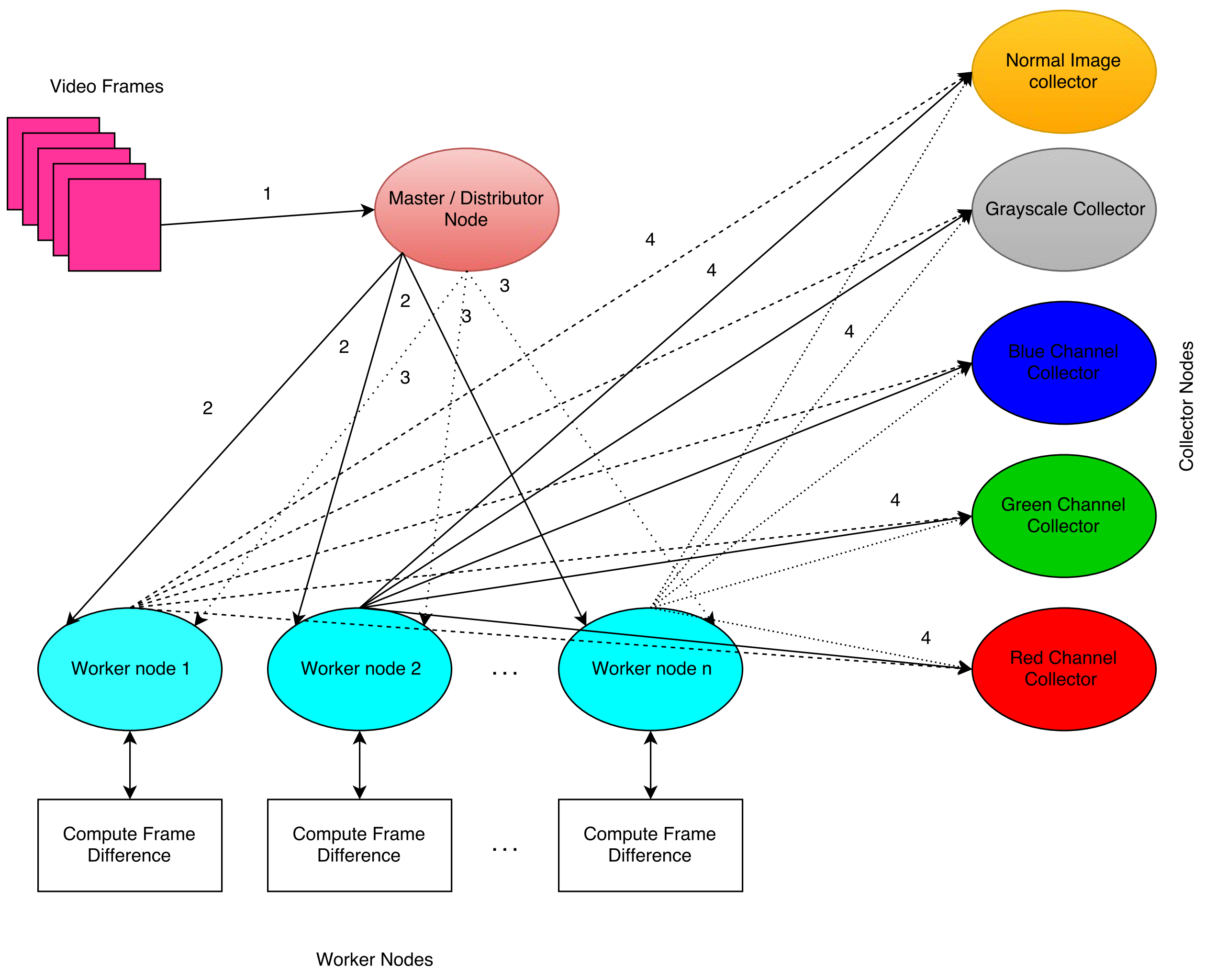

Figure 4.1: Design of Parallel Static Object Detection.

As shown in the diagram 4.1, we can break different functionality of our algorithm into 4 different actions.

- **Action 1:** Read frames from the video files or camera.
- **Action 2:** Send the initial background or the reference frame to the worker nodes.
- **Action 3:** Send the new frames to the worker nodes.
- **Action 4:** After computation of the frame difference, send the result frames to their respective channel collector. The Collector collects the result frames and writes into a result video.

4.2 Frame Difference in Parallel Static Object Detection

We have used the frame difference method for our algorithm. Any worker node at any particular instant will have three sets of frames. The background or the reference frame and the two new frames to compare with the background frame and with themselves. It is not always sure that the new frames are successive. The frame gap between the two new frames depends upon the number of worker nodes. This is because the master node sends the continuous frame in a loop starting from worker 1 to worker n and then repeats again from the worker node 1. If there are 5 worker nodes, then the frame gap between two successive new frames for a worker node is 5, *i.e* if the first new frame of a worker is indexed at 3 then it will receive the new frame indexed at $3 + 5 = 8$. There will be a frame gap of 5 frames between two successive frames for a particular node.

4.3 Parallel Static Object Detection : Pseudo Code

```
Data: Input Video File testVideo.avi
Result: Output Video Files out_Normal.avi, out_Gray.avi, out_Blue.avi, out_Green.avi, out_Red.avi
rank = current process rank
size = get size/total number of nodes
destination = 1
if (rank is 0 // Master) then
    while (The end of video file/ camera capture) do
        frame = read frame from video file
        if (frame is first frame ) then
            MPI_SEND(First frame as background to worker nodes)
        end
        else
            MPI_SEND(Send frame to worker node destination)
            destination ++;
        end
        if (destination is greater than or equal to (size - 5)) then
            Reset destination to 1
        end
    end
end
else if (rank is worker node) then
    backgroundFrame = MPI_Recv(Receive background frame from the master)
    break backgroundFrame into Gray, Blue, Red and Green Channel
    while (The end of video frame from master) do
        firstNewFrame = MPI_Recv(frame from master)
        break firstNewFrame into Gray, Blue, Red and Green Channel
        secondNewFrame = MPI_Recv(frame from master)
        break secondNewFrame into Gray, Blue, Red and Green Channel
        if ((background - firstNewFrame) is greater than threshold and (firstNewFrame -
        secondNewFrame) is less than threshold) then
            Static object detected
            Send highlighted frame to collector
        end
    end
end
```

```
else if (rank is size - 1) then
    normalFrame = MPI_Recv(Normal frame from workers)
    out_Normal.avi.append(normalFrame)
end
else if (rank is size - 2) then
    grayFrame = MPI_Recv(GrayScale frame from workers)
    out_Gray.avi.append(grayFrame)
end
else if (rank is size - 3) then
    blueFrame = MPI_Recv(Blue frame from workers)
    out_Blue.avi.append(blueFrame)
end
else if (rank is size - 4) then
    greenFrame = MPI_Recv(Green frame from workers)
    out_Green.avi.append(greenFrame)
end
else if (rank is size - 5) then
    redFrame = MPI_Recv(Red frame from workers)
    out_Red.avi.append(redFrame)
end
Output out_Normal.avi, out_Gray.avi, out_Blue.avi, out_Green.avi, out_Red.avi
```

Algorithm 1: Algorithm: Parallel Static Object Detection

4.4 Implementation Tools

Implementation of our research is done in structured programming. There were some external libraries used for image processing and Message Passing. We will discuss the usage of those tools in this section.

4.4.1 Programming Language : C

One of the major reasons for using C is because C has a close association with hardware and compiles faster than other object oriented languages.

4.4.2 Image Processing Library : OpenCV 3.1.0

OpenCV is an open source Computer Vision Library. OpenCV is the most popular library used in computer vision, particularly focusing on real time computer vision. The reason for using OpenCV is its huge functionality which provides various interfaces with image processing. By using OpenCV, we don't have to focus on array manipulation of images extensively. We have used OpenCV mostly for reading, writing and manipulation of images in this research. [9]

4.4.3 Message Passing Interface : Open MPI 1.6.5

Open MPI is also an Open Source Message Passing interface. OpenMPI provides a set of functionality to access different functions of a message passing system [1] . OpenMPI hides the connection layer from the programmer and provides interfaces to communicate between processors. One of the main drawbacks of Open MPI is the difficulty in debugging MPI code in our program.

Chapter 5

Results

5.1 Experimental Setup

System Parameters	Specifications
Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
Byte Order	Little Endian
CPU(s)	8
On-line CPU(s) list	0-7
Thread(s) per core	2
Core(s) per socket	4
Socket(s)	1
NUMA node(s)	1
Vendor ID	GenuineIntel
CPU family	6
Model	60
Stepping	3
CPU MHz	800.000
BogoMIPS	4788.74
Virtualization	VT-x
L1d cache	32K
L1i cache	32K
L2 cache	256K
L3 cache	6144K
NUMA node0 CPU(s)	0-7

Table 5.1: Experimental Setup Specifications

Following are the snapshots of the resulting frames grabbed from the result videos of Normal Video, Grayscale Video, Blue Channel Video, Green channel video and Red Channel Video.

5.2 Normal Image

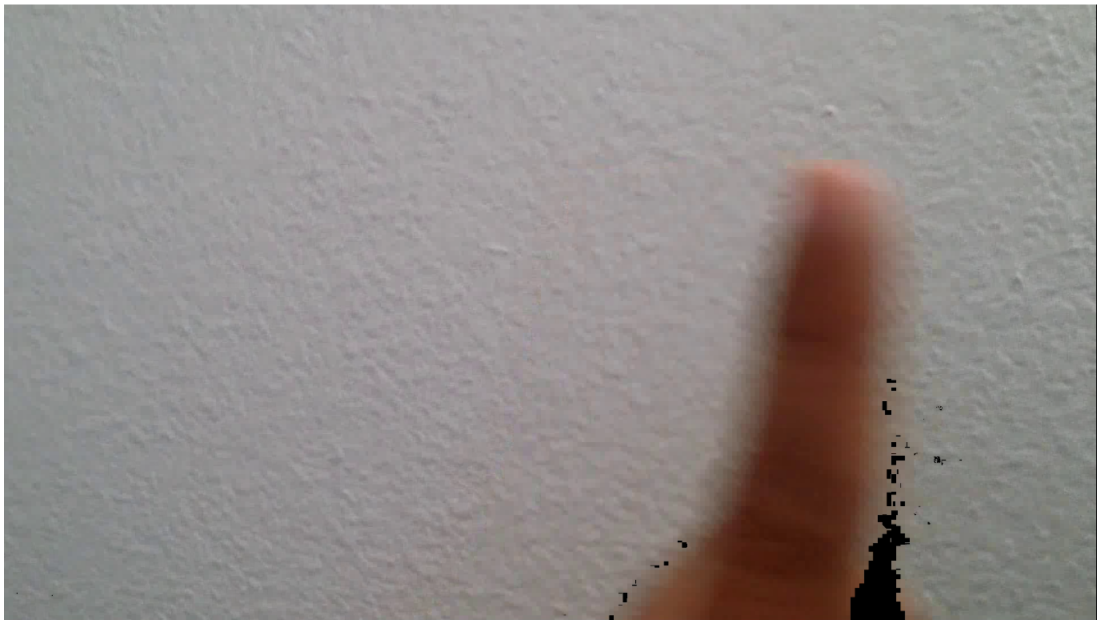
Figure 5.1: Normal with Motion

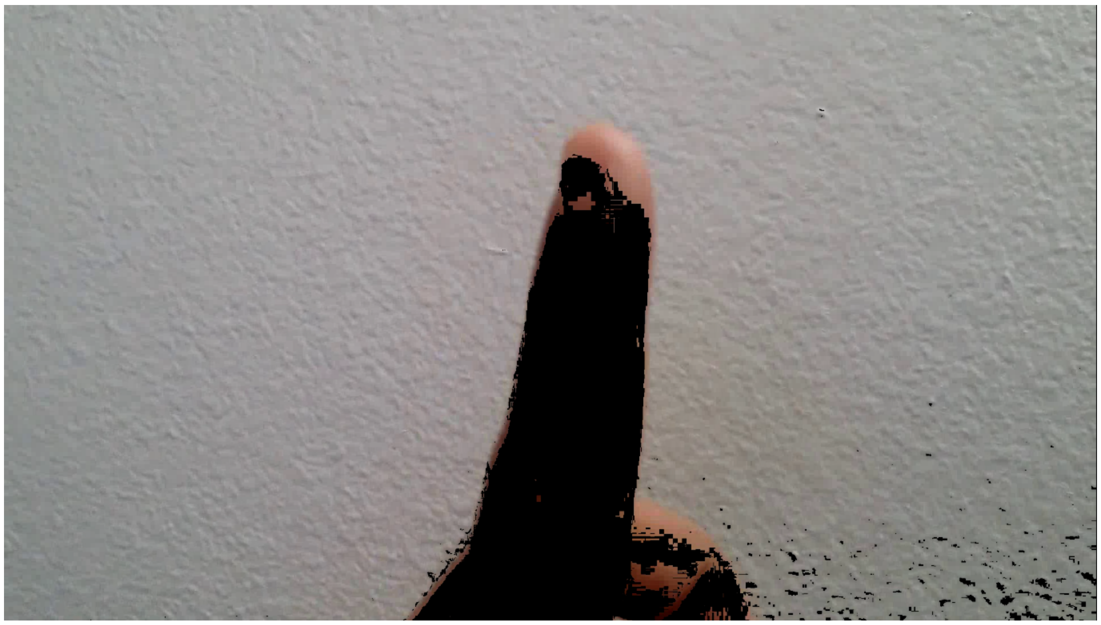
Figure 5.2: Static Object Detection in Normal

5.3 Gray Image

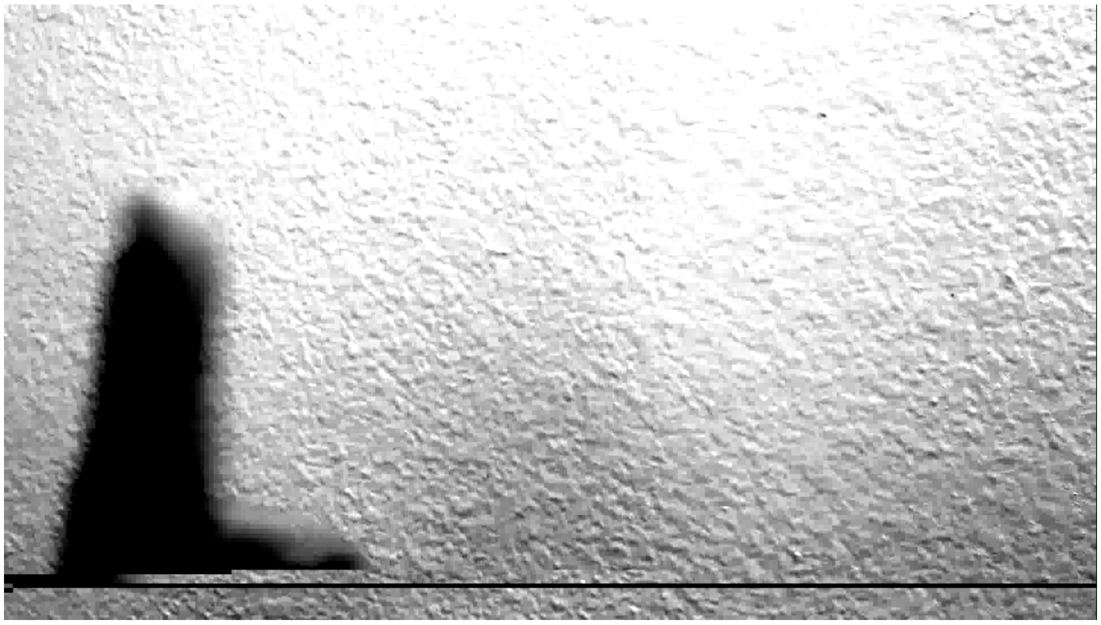
Figure 5.3: Gray with Motion

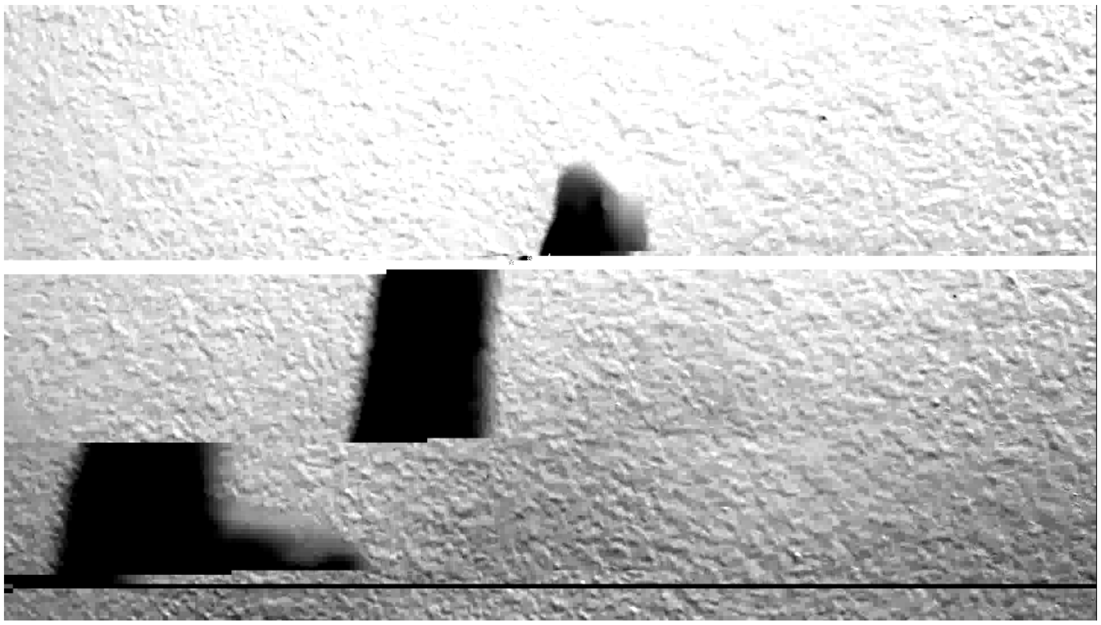
Figure 5.4: Static Object Detection in Gray

5.4 Blue Image

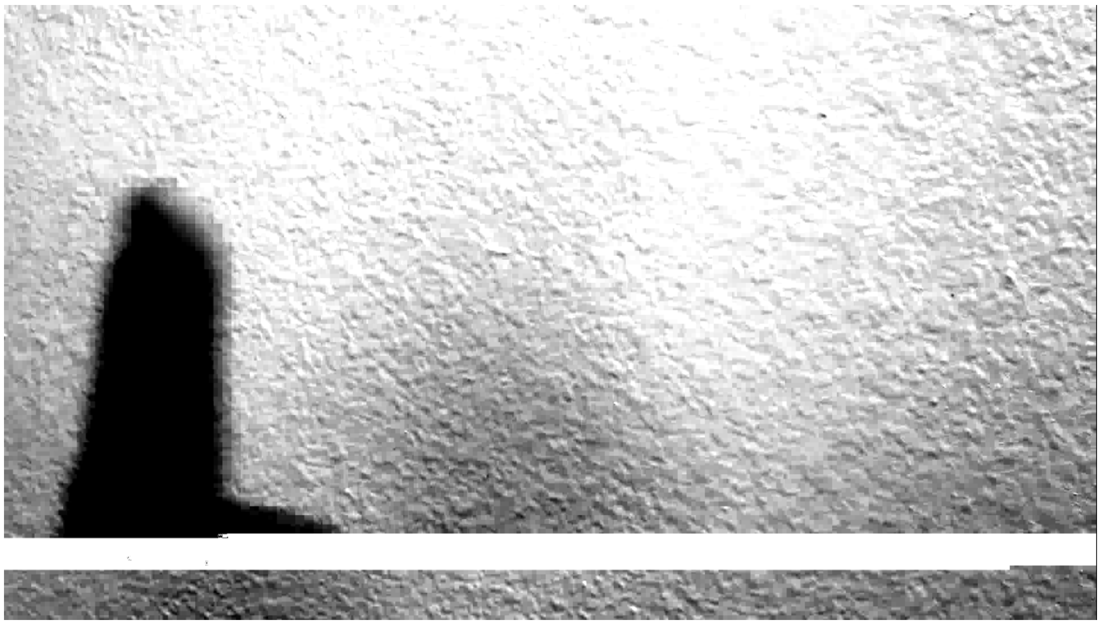

Figure 5.5: Blue with Motion

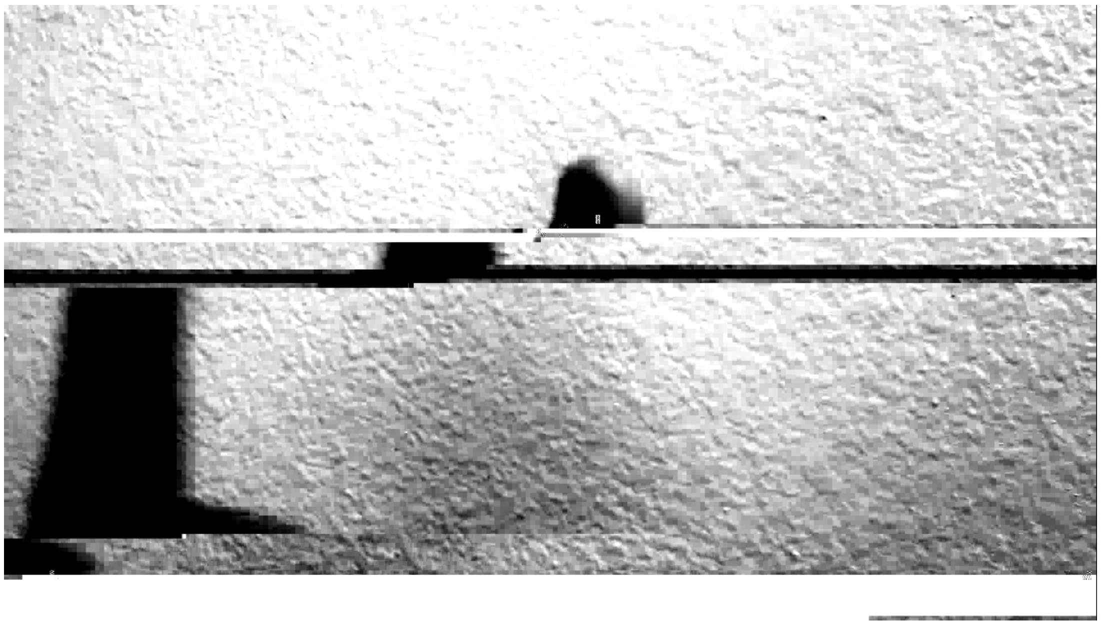
Figure 5.6: Static Object Detection in Blue

5.5 Green Image

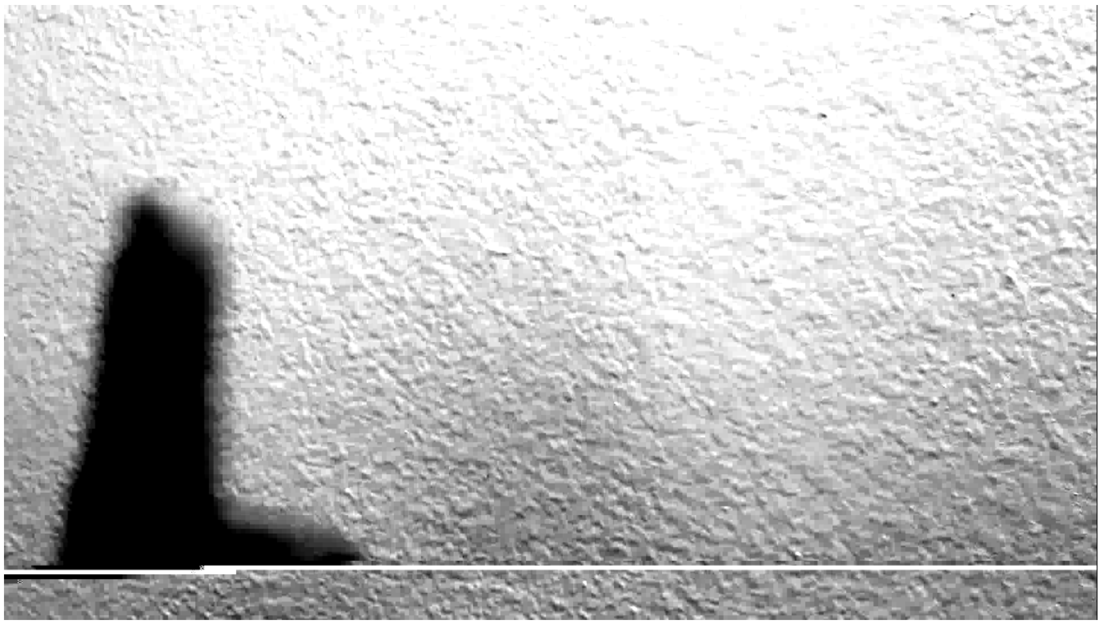

Figure 5.7: Green with motion

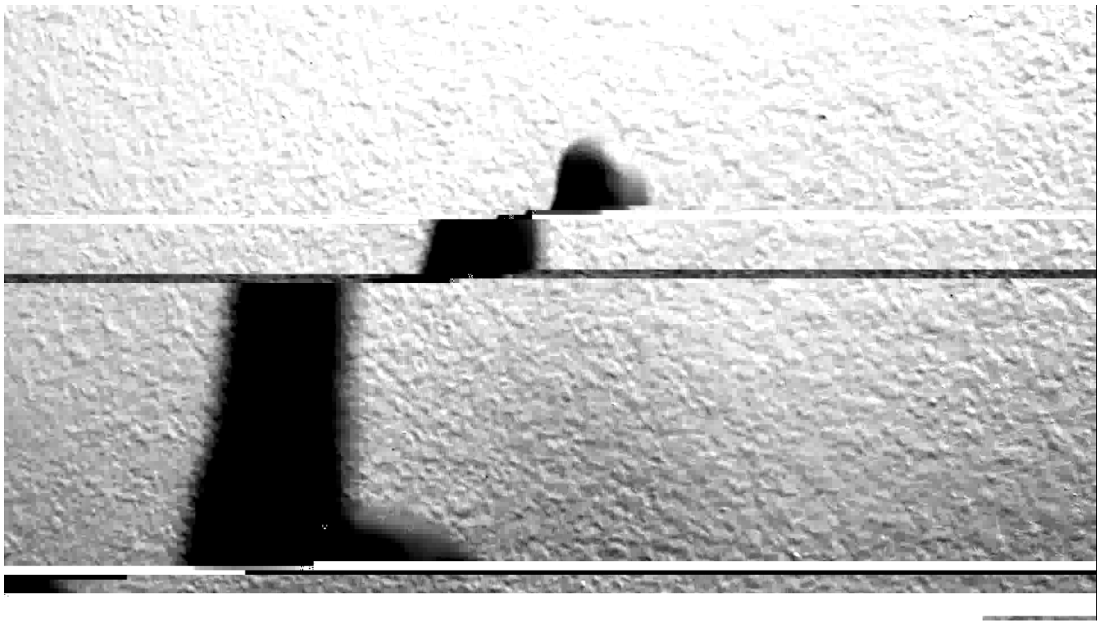
Figure 5.8: Static Object Detection in Green

5.6 Red Image

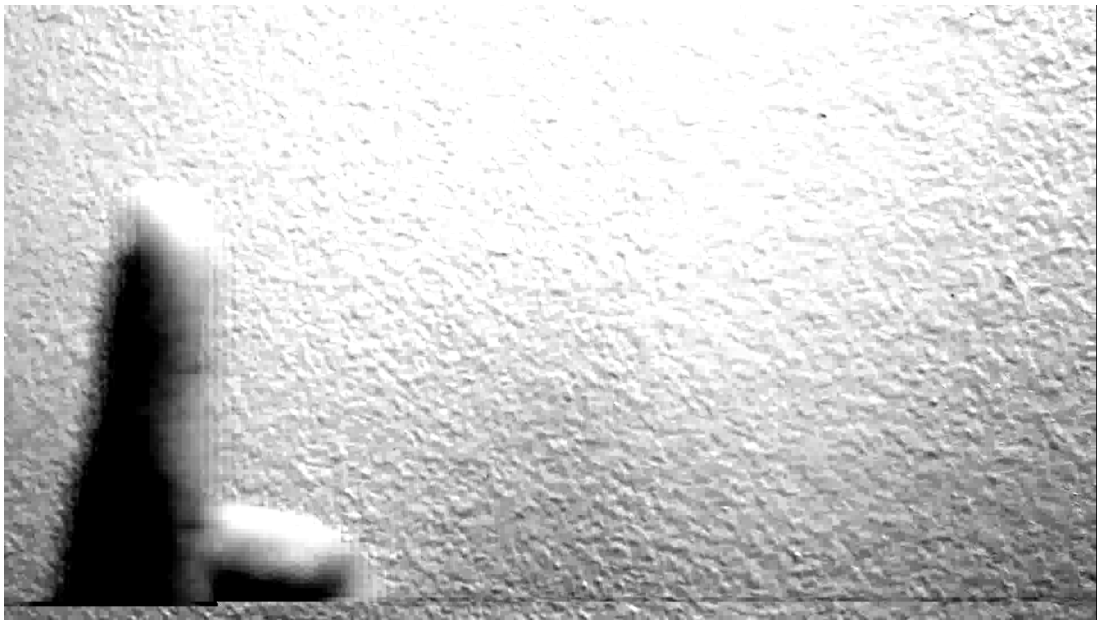

Figure 5.9: Red with Motion

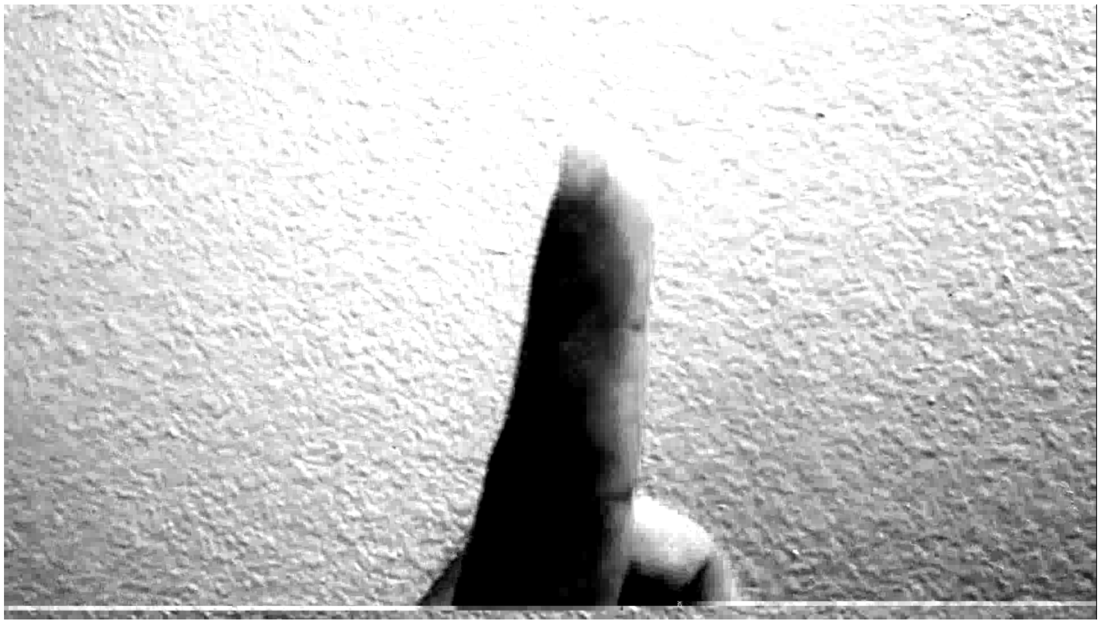
Figure 5.10: Static Object Detection in Red

5.7 Result Analysis

From the set of results, we can see that our algorithm performs best when there is an unaltered Normal Image. This is because, Normal image has information on all the Red, Blue and Green channel of the pixel, *i.e*, it is a 3 channel image. But we can see that on all the other channel, frames are flickered. Since these single channel images do not have information on all the channels, frame difference does not perform optimally with these images. This result overall shows that the frame differencing method does not perform well with the channel images which only consist of single channel data, but it performs well for the Normal Image which contains information of all the channels.

Chapter 6

Conclusion and Future Work

In this research, we learned that image processing and a parallel algorithm can go side by side. We proved this by using a primitive method of frame difference and thresholding, because these methods are intuitive when they are parallelised,

For future work, we can expand our parallel algorithm to incorporate more advanced algorithms in image processing. For Example, we can implement parallel programming with adaptive algorithms. In that scenario, instead of breaking videos by frames, we can use the approach of breaking a frame by pixels where each node works for a particular set of pixels in a frame, In this way they can maintain the historical information of the set of pixels which is essential for adaptive algorithms.

Bibliography

[1] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: portable parallel programming with the message-passing interface.* MIT press, 1999, vol. 1.

[2] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2. IEEE, 1999.

[3] B. Wilkinson and M. Allen, *"Parallel Programming", Techniques and Applications Using Networked workstations and Parallel Computers*, M. Horton, Ed. Prentice-Hall, Inc., 1999.

[4] "Chapter 4: Distributed and parallel computing," [Online; accessed 24-September-2015]. [Online]. Available: http://wla.berkeley.edu/~cs61a/fa11/lectures/communication.html

[5] M. Piccardi, "Background subtraction techniques: a review," in *Systems, man and cybernetics, 2004 IEEE international conference on*, vol. 4. IEEE, 2004, pp. 3099–3104.

[6] B.-H. Park and H. Kargupta, "Distributed data mining: Algorithms, systems, and applications," 2002.

[7] I. S. Dhillon and D. S. Modha, "A data-clustering algorithm on distributed memory multiprocessors," in *Large-Scale Parallel Data Mining.* Springer, 2002, pp. 245–260.

[8] J. A. Hartigan, "Clustering algorithms," 1975.

[9] "Reading and writing images and video," [Online; accessed 24-September-2015]. [Online]. Available: http://docs.opencv.org/2.4/modules/highgui/doc/reading_and_writing_images_and_video.html#reading-and-writing-images-and-video

Curriculum Vitae

Graduate College

University of Nevada, Las Vegas

Tirambad Shiwakoti

Degrees:

Bachelor of Computer Engineering 2011

Tribhuvan University, Institute of Engineering, Pulchowk Campus, Nepal

Thesis Title: Parallel Static Object Detection

Thesis Examination Committee:

Chairperson, Dr. Ajoy Datta, Ph.D.

Committee Member, Dr. Yoohwan Kim, Ph.D.

Committee Member, Dr. John Minor, Ph.D.

Graduate Faculty Representative, Dr. Venkatesan Muthukumar, Ph.D.