Multiprocessing Real Time Vision Based System for Condition Monitoring in Solar Panels

Moath Hashim Alsafasfeh
Western Michigan University, safasfeh77@gmail.com

Follow this and additional works at: https://scholarworks.wmich.edu/dissertations
Part of the Electrical and Computer Engineering Commons

Recommended Citation
https://scholarworks.wmich.edu/dissertations/3119

This Dissertation-Open Access is brought to you for free and open access by the Graduate College at ScholarWorks at WMU. It has been accepted for inclusion in Dissertations by an authorized administrator of ScholarWorks at WMU. For more information, please contact maira.bundza@wmich.edu.
MULTIPROCESSING REAL TIME VISION BASED SYSTEM FOR CONDITION MONITORING IN SOLAR PANELS

by

Moath Hashim Alsafasfeh

A dissertation submitted to the Graduate College in partial fulfillment of the requirements for the degree of Doctor of Philosophy Electrical and Computer Engineering Western Michigan University April 2017

Doctoral Committee:

Ikhlas Abdel-Qader, Ph.D., Chair
Bradely Bazuin, Ph.D., Co-Chair
Azim Houshyar, Ph.D.
Enabling an algorithm to be executed in parallel on a multicore or a multiprocessor system has become a necessity for many real time applications. Multicore systems are widely used to improve performance and satisfy time and power demands. In this dissertation, solar energy, which has proved itself as the future clean source of energy, is also considered in real time. Optimum utilization of this energy propelled research efforts into many directions of the solar system components. However, while real time operations of Photovoltaic (PV) systems occur without any supervisory mechanisms, many internal and/or external obstacles can occur and hinder a system’s efficiency. To address fault detection in solar systems and thus provide a safer and more time efficient inspection in real time, we are proposing using videos for real time inspection and fault detection for the solar panel.

To monitor the conditions of the solar system and issue an alert when a faulty condition is detected, an integrated multicore CPU system using real time recording and analyzing of thermal and photographic videos has been developed. The system using a multiprocessing module in Python and under a multicore CPU system, inputs thermal and photographic videos into different segments and executes detection algorithms in parallel. Each segment should be processed via a separate thread. Several Pattern Recognition algorithms are investigated for real time fault detection suitability.
Two cameras are used to capture the scene of the solar panels simultaneously while mounted on a drone. The FLIR Vue Pro thermal camera was used for thermal video recording with (NTSC) frame rate, and with a resolution of 336x256 pixels. This resolution is high enough to show an accurate thermal resolution from the solar panels. For visual images, a GoPro Hero 4 Black photographic camera was used in the system; the camera has effective photo resolution of 12.0 MP, and a max video resolution of 3840x2160. These two cameras are mounted on the Yuneec Typhoon Q500 quadcopter. The recorded videos are streamed into the ground workstation where they are processed using the Python 2.7- IDE for Eclipse (Luna Service Release 2 (4.4.2)).

To validate our real time proposed system, we used a mobile solar system that was constructed primarily for this project in the Digital Image and Signal Processing Laboratory (DISPLAY) at Western Michigan University (WMU). This system is composed of two panels of SUNIVA OPTIMUS 60 Cell modules (Model OPT285-60-4-1B0); each panel is rated at 285W. The proposed system, as demonstrated by the results, has the following contributions: 1) Using the multiprocessing module in Python and the thermal and photographic video processing on multicore CPU shows execution time improvement and processor performance enhancements; the average improvement for the processing time of the detection algorithms for thermal and photographic videos was 3.1 times using 2 processes, and 6.3 times using 4 processes; and 2) a multicore real time system for the analysis of thermal and photographic videos, drone mounted, provides the capability to accurately detect defects in the solar panels and give location information in terms of panel location by longitude and latitude.
ACKNOWLEDGMENTS

I would like to express my sincere gratitude and appreciation to my advisor; Professor Ikhlas Abdel-Qader, for her encouragement, support, and guidance during my PhD journey. This dissertation would not have been possible without her supervision, suggestions, and valuable feedback. Dr. Abdel-Qader is a very kind and nice person that I am grateful to have in my life. There are no words to express her magnificence and I will never forget the support she provided during my studies at WMU.

I would like also to extend my appreciation to Dr. Bradley Bazuin for his suggestions and help toward the completion of my dissertation. His support while building the solar system and feedback on the results are greatly appreciated. Many thanks also for Dr. Azim Houshyar for his efforts and time while serving on my dissertation committee.

I would like to thank my beloved parents for their love and support all my life and for the encouragement I received from them while studying abroad. My sincere thanks also go to my dear brothers and sisters for their love and support during this journey and through my whole life.

Also, I would like to provide my great thanks and gratitude to my lovely wife, Alia. She is the powerful supporter and the daily dose of giving, and at times, of pushing to keep me going to complete this dissertation. I truly cannot find words to express my appreciation of her being with me these years. Last but not the least, to all my friends I have made along the way, many thanks as your friendship means a lot to me.

Moath Hashim Alsafasfeh
# TABLE OF CONTENTS

ACKNOWLEDGMENTS............................................................................................................ ii

LIST OF TABLES..................................................................................................................... vi

LIST OF FIGURES .................................................................................................................. vii

CHAPTER

I.  INTRODUCTION ............................................................................................................... 1

   1.1 Significance of the Research................................................................................ 2

   1.2 Statement of the Problem..................................................................................... 3

   1.3 Research Questions.............................................................................................. 3

   1.4 Organization ........................................................................................................ 4

II.  BACKGROUND DESCRIPTION ..................................................................................... 5

   2.1 Parallel Processing................................................................................................... 6

       2.1.1 Multicore System........................................................................................ 7

       2.1.2 Multiprocessor System.............................................................................. 11

       2.1.3 Instruction Level Parallelism ..................................................................... 12

       2.1.4 Data Level Parallelism .............................................................................. 14

       2.1.5 Thread Level Parallelism ........................................................................... 14

       2.1.6 Parallel Programming ................................................................................ 15

       2.1.7 Comparison between Multicore and Multiprocessor Systems .................... 17

       2.1.8 Performance Indicator Metrics ................................................................. 17
Table of Contents—Continued

CHAPTER

2.1.9 Parallel Programming Platforms ............................................................... 21

2.2 Solar Energy System ...................................................................................... 24

2.2.1 Photovoltaic (PV) Cells ............................................................................. 25

2.2.2 Solar Cell Types ......................................................................................... 30

2.2.3 PV Modules Efficiency and Cost ............................................................... 34

2.3 Fault Types in Solar Panels ......................................................................... 36

2.4 Infrared Technology for Non-destructive Testing ......................................... 41

2.5 Concluding Remarks ..................................................................................... 45

III. PERTINENT LITERATURE .............................................................................. 46

3.1 Real Time Image Processing ......................................................................... 46

3.2 Investigation of Defects in Solar Cells ......................................................... 48

3.2.1 Detect Shunting in PV Cell ...................................................................... 49

3.2.2 Faults from Solder Bonds ....................................................................... 50

3.2.3 Heating from Reverse-Bias ...................................................................... 51

3.3 Fault Detection in Solar Panel using Embedded Sensors .............................. 51

3.4 Infrared Technology ...................................................................................... 57

3.5 Concluding Remarks ..................................................................................... 60

IV. PROPOSED FRAMEWORK – MULTIPROCESSING REAL TIME VISION BASED SYSTEM FOR CONDITION MONITORING IN SOLAR PANELS ..................... 62

4.1 Multicore System for Fault Detection ............................................................. 65
## Table of Contents—Continued

**CHAPTER**

4.2 Defect Detection Algorithms ................................................................. 69

4.2.1 Morphological Transformation and Canny Edge Algorithm .......... 69

4.2.2 SLIC Super-Pixel Algorithm ............................................................... 72

4.2.3 Segmentation Based on Hot Pixels Seeds Algorithm ..................... 75

V. EXPERIMENTAL WORK AND RESULTS .................................................... 79

5.1 Data Acquisition .................................................................................... 79

5.2 Multicore System for Defect Detection in PV using Video Processing ... 84

5.3 Defect Detection in PV Solar Panel ......................................................... 94

5.3.1 Morphological Transformation with Canny Edge Detector .......... 94

5.3.2 SLIC Super-Pixel Hot Spot Detector ................................................ 96

5.3.3 Segmentation Based Hot Pixels Detection ........................................ 96

VI. CONCLUSION .......................................................................................... 104

6.1 Summary ............................................................................................... 104

6.2 Contributions ......................................................................................... 105

6.3 Future works ......................................................................................... 105

Bibliography ............................................................................................... 107
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Multicore versus Multiprocessor</td>
<td>17</td>
</tr>
<tr>
<td>2.2</td>
<td>Summary of Efficiency and Market for PV Technologies</td>
<td>35</td>
</tr>
<tr>
<td>2.3</td>
<td>The Worldwide Market Price of PV Modules, Q4 2009 to Q1 2012</td>
<td>36</td>
</tr>
<tr>
<td>3.1</td>
<td>Isofoton 106-12 PV Module Characteristics Parameters</td>
<td>54</td>
</tr>
<tr>
<td>3.2</td>
<td>RMSE Between Measured and Simulation Results</td>
<td>54</td>
</tr>
<tr>
<td>5.1</td>
<td>WMU Solar Panel System Specifications in DISPALY lab</td>
<td>80</td>
</tr>
<tr>
<td>5.2</td>
<td>Input Of Thermal Videos for Feature Detection</td>
<td>84</td>
</tr>
<tr>
<td>5.3</td>
<td>Processing Time for Canny Edge Detection Execution Using Multicore</td>
<td>84</td>
</tr>
<tr>
<td>5.4</td>
<td>Processing Time for Histogram Equalization Execution Using Multicore</td>
<td>85</td>
</tr>
<tr>
<td>5.5</td>
<td>Input of Thermal Videos for Fault Detection</td>
<td>87</td>
</tr>
<tr>
<td>5.6</td>
<td>Processing Time using Morphological Transformation and Canny Edge Detection Execution for Thermal Video Using Multicore</td>
<td>87</td>
</tr>
<tr>
<td>5.7</td>
<td>Processing Time using SLIC Super-Pixel Execution for Thermal Video Using Multicore</td>
<td>89</td>
</tr>
<tr>
<td>5.8</td>
<td>Processing Time for Segmentation Based Hot Pixels Detection for Thermal Video using Multicore</td>
<td>90</td>
</tr>
<tr>
<td>5.9</td>
<td>Input of Thermal and Photographic Videos for Defects Detection</td>
<td>91</td>
</tr>
<tr>
<td>5.10</td>
<td>Processing Time for Morphological and Canny Edge Detection Execution for Thermal and Photographic Videos Using Multicore</td>
<td>91</td>
</tr>
<tr>
<td>5.11</td>
<td>Processing Time for SLIC Super-Pixel Execution for Thermal and Photographic Videos Using Multicore</td>
<td>92</td>
</tr>
<tr>
<td>5.12</td>
<td>Processing Time for Segmentation Based Hot Pixels Detection for Thermal and Photographic Videos Using Multicore</td>
<td>93</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

2.1: Single CPU versus Multicore CPU Architecture ............................................................. 8
2.2: Execution Time on Single Core versus Multicore ........................................................... 9
2.3: Single Core Performance versus Multicore Performance ................................................. 9
2.4: Parallel Processing Control Models .............................................................................. 16
2.5: Solar Energy System Components ................................................................................ 25
2.6: Solar PV Configurations ............................................................................................... 26
2.7: Photovoltaic Panel Construction ................................................................................... 27
2.8: A Simple Solar Cell Model ........................................................................................... 29
2.9: Structure of CdTe Thin-film Solar Cell .......................................................................... 32
2.10: Structure of a-Si Thin-film Solar Cell ......................................................................... 33
2.11: Structure of CIGS Thin-film Solar Cell ....................................................................... 33
2.12: Reported Timeline of Solar Cell Energy Conversion Efficiencies ............................... 35
2.13: Line-Line Fault in PV System .................................................................................... 39
2.14: Open Fault in PV System ........................................................................................... 40
2.15: Ground Faults in PV System ....................................................................................... 40
4.1: System Description ...................................................................................................... 63
4.2: Proposed Method Flow Chart ....................................................................................... 64
4.3: Video Segmentation using ffmpeg ................................................................................ 66
4.4: Code Structure for Multiprocessing Module in Python ............................................... 67
4.5: Multiprocessing Module ............................................................................................... 68
4.6: Using of Morphological and Canny algorithm .............................................................. 70
4.7: Removing the Unwanted Pixels Using Canny Algorithm ............................................ 71
List of Figures—Continued

4.8: Thresholding Process in Canny Algorithm ................................................................. 72
4.9: Implementation of SLIC Super-Pixel Flow Chart using Python ............................ 73
4.10: SLIC Super-Pixel algorithm ..................................................................................... 74
4.11: Segmentation Based on Hot Pixels Seeds ............................................................... 77
5.1: Schematic Diagram for WMU Solar Panel System in DISPLAY Lab ..................... 81
5.2: WMU Solar Panel System in DISPLAY Lab ................................................................. 82
5.3: Drone System with Thermal and Photographic Cameras ......................................... 83
5.4: Speedup Results for Canny Edge Detection Algorithm Using Multicore ............... 85
5.5: Speedup Results for Histogram Equalization Using Multicore ......................... 86
5.6: Simulation Results for Feature Detection ................................................................. 86
5.7: Speedup Results for Morphological with Canny Edge Detection Algorithm for Thermal Video Processing using Multicore ....................................................... 88
5.8: Speedup Results for SLIC Super-Pixel Algorithm for Thermal Video Processing Using Multicore ................................................................. 89
5.9: Speedup Results for Segmentation Based Hot Pixels Detection for Thermal Video Using Multicore ................................................................. 90
5.10: Speedup Results of using Morphological with Canny Edge Detection Algorithm for Thermal and Photographic Videos Using Multicore ................................................... 92
5.11: Speedup Results for SLIC Super-Pixel for Thermal and Photographic Videos Using Multicore ................................................................. 93
5.12: Speedup Results for Segmentation Based Hot Pixels Detection for Thermal and Photographic Videos Using Multicore ................................................... 94
5.13: WMU Healthy and Defective Solar Panels: Morphological with Canny Detector Results ................................................................. 95
5.15: Results for Segmentation Based Hot Pixels Detection Algorithm ....................... 97
5.16: Output Results for an External Defect by Adhesive Paper .................................. 98
List of Figures—Continued

5.17: Output Results for External and Internal Defects by Gum and Polystyrene ............... 99
5.18: Output Results for Defective Solar Panels .......................................................... 100
5.19: Location Information for Defective Solar Panel (external and internal defects)......... 101
5.20: Location Information for Defective Solar Panel (external defects with shadow) ...... 102
5.21: Shadow Detection using SLIC Super-pixel Algorithm ........................................... 103
CHAPTER I

INTRODUCTION

The multicore processor is a computer system that has more than one core in the single Central Processing Unit (CPU). These cores are the ones that read and perform your program’s instructions on your CPU. Multicore can physically execute multiple instructions at the same time which enables the algorithm designer to run tasks in parallel. The main importance of the multicore processor is to enhance the system performance by reducing the execution time and reducing the latency communication between the computer system components. Multicore processors can deliver a throughput at lower power consumption than unicore processors. A multiprocessor is a single computer system that has more than one CPU working simultaneously. The running tasks can be allocated among all the CPUs. The main importance of the multiprocessing system is to increase the speed of the execution. The performance is improved once the execution time is reduced. All the CPUs in a multiprocessor system can share the communication bus, memory, and the input/output devices, and this sharing enables the system to process large amount of data at high speed.

Solar energy has been gaining strong momentum as a future clean source of energy. The efficiency of the solar energy system plays the most significant attribute for power generation for economic stature. Therefore, optimum utilization of this energy source propelled research efforts into many components of the solar system such as photovoltaic (PV) where improvement can lead to better system efficiency. Real time operations of PV systems occur without any supervisory mechanism but they still can have many obstacles internally and/or externally hindering efficiency. Prompt identification of faulty PV cells will allow for better system operation, timely maintenance, and enhanced safety.
Infrared image technique has a proven record in non-destructive testing, such as in detection of defects within building walls, faulty engines, plastic manufacturing quality control, and many other problems in industrial applications and electrical systems [1].

1.1 Significance of the Research

The multicore systems are widely used to improve performance and satisfy time and power demands from users. A computer system with more than one core working simultaneously does satisfy such demand. In this dissertation, the use of the multicore system is important where the defect detection process for the solar panel should take place in real time, while the two connected cameras, thermal and photographic on the drone, are recording a video of the solar panels. The output video files are processed and analyzed using the multiprocessing module in Python on the multicore CPU. Also, GPS data can be gathered from the drone in order to coordinate exactly the physical location of the defective panel. All these acquired data need to be processed in the real time in order to automate the fault detection in the solar panel and get an alert for any hazards.

The entire planet can be powered for a whole year by only the amount of energy that the sun provides to the earth in a single day [2]. The first source of energy used by humans was solar energy [2]. It was used to heat through direct contact or to dry items of clothing. Solar energy has been utilized since the 1950s to generate power for businesses, homes, and fuel technology [2]. Fault detection for the solar panels plays an important role which affects the whole system performance, reduces the risk of reducing power production, keeps the system safe from fire hazards, and keeps the system in reliable operation mode for a long life time.

This dissertation is intended to develop a multicore real time vision based system for condition monitoring in solar panels using thermal and photographic videos, and that can automate defect detection in the solar panels in real time. The proposed system will enhance
real time inspection by reducing the processing time. Also, the system provides a safer inspection method since images are obtained from a distance using a camera.

The main contribution of the work presented in this dissertation is in developing a multicore system that meets time and power demands necessary for successful a real time system capable of scanning large solar farms while streaming and processing two videos using thermal and photographic videos. Also through this research, I am able to address the need for a timely identification of faulty PV panels in real time. The proposed fault detection-multicore system is fully automated and allows the solar system to have timely maintenance and thus operate at a higher efficiency and enhanced safety conditions.

1.2 Statement of the Problem

The purpose of this work is to use the multicore system to automate and enhance the conditions monitoring for solar panels with multiple input infrared and photographic images for the solar panels or solar garden.

1.3 Research Questions

In this work, the following questions have been answered:

- Can the defects detection process for solar panels be automated by using the multicore system?
- Can the multicore system improve the fault detection process?
- Can the infrared and photographic images detect the faults in the solar panels?
- Can the proposed system determine the location of the defective solar panel?
- Can all types of defects be detected by the proposed system?
1.4 Organization

Chapter I introduces the significance, objectives, and research questions of this project. An introduction to the parallel processing, the multicore and multiprocessor systems, the importance of the multicore systems, the parallel programming, solar panels, and application of the thermal images, are presented in chapter II. Chapter III explores the previous literature of the methods of chapter II. Chapter IV introduces the proposed system for conditions monitoring, while chapter V discusses its results, followed by conclusion and summary in chapter VI.
CHAPTER II

BACKGROUND DESCRIPTION

Parallel processing is a powerful tool for algorithmic and architectural methods, which enhance the performance and provide reliability for the computer system. Most modern computers provide the ability of parallel processing. The execution time is reduced using parallel processing, which improves the system performance. Parallel processing can take place by using multicore or the multiprocessor systems. The different cores or processors help to execute different tasks in parallel at the same time. Indeed, parallel processing increases the speed of execution by enabling an algorithm to run multiple tasks simultaneously.

Photovoltaic systems (PV) are based on the principle of converting light into electricity which occurs without any supervisory mechanism. Since the efficiency of the solar energy system is the most significant attribute for power generation, timely faults detection is essential. Therefore, prompt identification of faulty PV cells will allow for higher system efficiency, timely maintenance, and enhanced safety.

In non-destructive testing, infrared imaging techniques have a proven record for fault detection in different domains such as faulty engines, quality control in manufacturing, and the detection of many other problems in industrial applications and electrical systems. Infrared image technology provides a safe tool for fault detection in the solar energy system.

This chapter represents some background and concept materials on the parallel processing in the multicore and multiprocessor systems and shows the importance of multicore system usage for reducing time execution and improving the processing performance.
The multicore architectures are represented in this chapter and the comparison between multicore and multiprocessor systems has been discussed. Parallel programming is also explained along with some of the parallel software programing platforms, such as Python, OpenMP, and CUDA. The next phase in this chapter represents the main architecture of the solar energy systems, and explains in details the solar cell types. The fault types that can occur in the solar panel are represented. Then the infrared technology is represented in the last section.

2.1 Parallel Processing

Multicore and multiprocessor systems are able to run multiple tasks at the same time. Parallel processing is the process of dividing the program instructions and run them among multicore or multiprocessor systems. Thus, the program will run in less time. High-performance uniprocessors are becoming increasingly complex, power-hungry, and expensive [3]. The trade-off exists between the use of complex processors (one or small numbers) at one extreme, and the use of simpler processors (a moderate to very large number). The latter approach leads to significant simplification of the design process [3].

The importance of parallel processing can be summarized based on the following reasons [3]:

- Solving problems faster or in higher speed, and this is important when applications have deadlines.
- Higher throughput, which means solving more instances of given problems.
- Solving larger problems which means higher computational power.

Parallel processing has some challenges; the first obstacle has to do with the limited parallelism available in programs. This limits the impact of parallelism in the speedup. The second reason relates from the relatively high cost of communication; this involves the large
latency of remote access in a parallel processor [4]. For a parallel algorithm, the main issue is the way in which the computational load is divided between the multiple processors [3].

Multiprocessing is the use of two or more CPUs within a single computer system. The multiprocessing computer is a system that has the capability to process more than one task simultaneously, which increases the system performance as compared with a single processor. Multiprocessing systems have increased importance because of the following factors [4]:

- Attempts to find and exploit more Instruction Level Parallelism (ILP).
  - Multiprocessing is a scalable and general purpose way to increase performance above the basic technology.
- A growing interest in software-as-a-service and high-end servers as cloud computing become more important.
- A growth in data-intensive applications and the availability of massive amounts of data on the Internet.
- Multiprocessing systems design provides the advantages of leveraging a design investment by replication rather than running a unique design.

Multicore and multiprocessor systems are used to achieve the aim of a multiprocessing system. These two systems are explained in detail in the following sections.

2.1.1 Multicore System

A multicore system consists of more than one core within a single CPU capable of executing multiple tasks at the same time. A multicore CPU reads and executes a program’s instructions, and a multicore can execute multiple instructions at the same time. These instructions are the ordinary CPU instructions, for example add, move, and branch. These cores are usually integrated into a single IC (integrated circuit) die, or onto multiple dies but in a
single chip package [5]. Figure 2.1 shows the main architecture difference between single CPU and multicore CPU.

The execution of multiple tasks on a single CPU is done by assigning a time slice to work on one task. Once this task is completed, the operating system will assign slices for the remaining tasks based on the process scheduler. In other words, a long-running process will cause the system to have a large run queue [6]. On the other hand, the multicore can run multiple tasks in parallel at the same time leading to significant reductions in execution time of the task at hand. Figure 2.2 shows an illustration of how performance is enhanced using equation (2.1). The performance is improved once the execution time is reduced. The performance has been improved using a multicore CPU. Figure 2.3 compares the achieved performance improvement, from 2000 to 2008, between a single core and a multicore.

\[
\text{Performance} = \frac{1}{\text{Execution Time}} \tag{2.1}
\]

Figure 2.1: Single CPU versus Multicore CPU Architecture
Gordon Moore predicated that the number of transistors on a chip will be doubled for every 18 months, which is known as Moore’s Law [7]. In other words, the processor performance will be improved two times every 18 months. Based on this, there are risks to the design complexity and to the power dissipation. Using the multicore processor on a single chip should be able to double the performance of a single faster processor [8]. The reason is the individual cores on a multicore processor are working on a low frequency and execute more tasks in parallel; in other words, they do not necessarily run at a high frequency as does the single core [6].
In the multicore systems, the execution parts of the processor were duplicated; for example, Arithmetic Logic Unit (ALU), fetches and decodes unit, instruction pipeline, and some cache memory. Thus, each core contains all the resources necessary to perform computational tasks that do not involve interacting with components outside the CPU or individual core.

The demand on improved performance in all segments of embedded system applications is rising, and tracking the performance improvements in hardware and software. Naturally, improving performance can be addressed by using the multicore or multiprocessor with parallel programming capabilities [9]. A multicore system provides simultaneous running for multiple tasks at the same time; the execution time should be improved with completion of all the tasks in a shorter time.

Multicore processors are not clocked at a higher frequency, and they can execute programs in parallel. In the meantime, multicore is generally designed to partition the task into subtasks, so that the unused cores run at lower power consumption, or they are powered up only when needed, making the system more energy efficient [6].

The system’s performance can be improved by using the parallel programming; this can be achieved by adding more parallel resources while maintaining manageable power properties [9]. Many new applications are multithreaded [10], and using a multicore for these applications will positively affect the whole system performance. In addition, the multicore system can execute re-order, pipeline instructions, split them into microinstructions, and do aggressive branch prediction [10], which means more than one instruction can be executed in the same clock cycle. This allows for the Instruction-level parallelism (ILP) that has enabled rapid increases in processor speeds over the last 15 years [10]. There are many examples of successful applications in a multicore processor, such as running an anti-virus program while
downloading software, or recording a TV show while modifying a photo [10]. On the other hand, there are many application that benefits from using a multicore or multiprocessor system, such as multimedia applications, scientific applications, and applications with thread level parallelism, web servers, web commerce, compilers, and database servers.

The cores in the multicore system can be classified into two types; the homogeneous multicore system has more than one core and all cores have the same ISA and performance. The cores can execute the same binaries. This homogenous multicore typically has full cache coherency and shares a global address space. In addition, a homogenous multicore is easier to program for parallelism [11]. A homogeneous multicore processor has reusability, has reduced design complexity, reduced verification effort, and hence it is easier to meet time-to-market criteria. A homogenous multicore improves the overall processor performance by applying a divide and conquer approach: the high computationally intensive applications are broken into less computationally intensive pieces and executed in parallel [12].

A Heterogeneous multicore has different kinds of cores which differ in ISA and performance [11]. Low power can be achieved by using a heterogeneous multicore, and it can target the issue of running a variety of specialized computations and applications to be executed on a computer, for example addressing multimedia applications that require heavy mathematical calculations [12]. The programming model for heterogeneous architectures is much more complicated than a homogeneous multicore [9]. Multicore processors could also be implemented as a combination of both heterogeneous and homogeneous cores to improve performance and optimize for specific tasks [12].

2.1.2 Multiprocessor System

The multiprocessing system contains more than one processor, and they have the ability to work simultaneously, each processor representing one CPU. Any computer system with
more than one processor is called multiprocessor. All the CPUs in the multiprocessor system have the same specifications and the Operating System (OS), and all the identical CPUs share and connect to the main memory; this system is called a Symmetric Multiprocessor (SMP). On the other hand, the multiprocessor can have different CPUs, and they are not equal; this system is called Asymmetric Multiprocessor (AMP). A multiprocessor can be classified based on the number of executed instructions; Single Instruction Multiple Data (SIMD) is a popular processor in modern graphic cards. SIMD means all the parallel CPUs share the same Instruction Set Architecture (ISA). Multiple Instruction Single Data (MISD) provides the redundancy, for example, systolic arrays or sequential execution in time on the same data element. Multiple instructions Multiple Data (MIMD) is the most popular type. MIMD means that the CPUs have separate data and instructions which means each of them can execute something different at any time. MIMD is known as a multicore processor; different cores execute different threads, operating on different parts of memory [10]. The multiprocessor systems can be different based on the memory type. The first type is a shared memory between the multiprocessors, which means there is just a one large common memory shared by all processors. One of the most popular examples is the multicore [10]. The second type is the distributed memory; in this type each processor has its own small local memory and its contents are not replicated anywhere else [10].

2.1.3 Instruction Level Parallelism

Performance can be improved by overlapping the execution of instructions, which is the use of pipelining. The overlap among instructions is called Instruction Level Parallelism (ILP), since the instructions can be evaluated in parallel [4]. ILP measures the number of the program’s instructions that can be executed in parallel. ILP can be employed by two approaches: hardware-based approach, which discovers and exploits the parallelism
dynamically; the processor decides which instructions to execute simultaneously, for example Intel Core series. The software-based approach finds parallelism statically at compilation time [4], for example Intel Itanium series.

One of the simplest ways for ILP increasing is employing parallelism among iterations of a loop, which is called Loop-level parallelism. The important method for employing loop-level parallelism is the use of SIMD in both Graphics Processing Units (GPU) and vector processors [4]. Data Level Parallelism can be employed in SIMD by operating a small to moderate number of data items in parallel.

ILP can be exposed by different techniques [4]: Instruction pipelining achieves parallelism among instructions by finding sequences of unrelated instructions that can be overlapped in the pipeline. The compiler performs the scheduling for the independent and dependent instructions. Branch predictors are one of the useful techniques for ILP which reduce control dependences.

Dynamic scheduling employs ILP by reducing the stalls that can exist in static scheduling. The hardware rearranges the execution instructions to reduce the stalls while maintaining data flow and exception behavior.

Hardware-Based Speculation is used to enhance ILP by overcoming control dependences. Hardware-Based Speculation involves speculating on the outcome of branches and executing the program as if branch predictions were always correct. Hardware-based speculation combines three ideas: dynamic branch prediction, speculation to allow the execution of instructions before the control dependences are resolved, and dynamic scheduling to deal with the scheduling of different combinations of basic blocks.

ILP may also be enhanced and employed by multiple-issue processors that allow multiple instructions to be issue in a single clock cycle, for example, Very Long Instruction
Word (VLIW), statically scheduled superscalar processors, and dynamically scheduled
superscalar processors.

2.1.4 Data Level Parallelism

Having parallel data operations achieves parallel speedup [4]. Data Level Parallelism
is a distribution of the data across different processes which operate on the data in parallel.
DLP means different processes in the multiprocessor or multicore system perform the same
operation on different data sets, for example SIMD. This process can be called a parallelizing
or a sequential program in SIMD. DLP can achieve the similar performance to ILP, or even
more, in an in-order vector processor.

DLP is implemented using three variations of SIMD; vector architecture, multimedia
SIMD instructions set extension, and GPUs [4]. A vector machine is the pipelined execution
of many data operations. Vector architectures are easier to understand and to compile than
GPUs and Multimedia SIMD, but they are specialized and very expensive.

Multimedia SIMD means basically simultaneous parallel data operations and is found
in most ISA today that support multimedia applications. Also, SIMD instructions are used to
get the highest computation rate for floating point operations on x86 computers.

GPU based systems offer higher potential performance than is found in multicore CPU.
GPUs share features with vector architecture. This environment has a system memory and
system processor in addition to the GPU and its graphics memory [4].

2.1.5 Thread Level Parallelism

Using multiple threads will increase the uniprocessor throughput which hides pipeline
and memory latencies [4]. The functional units of a single processor can be shared by multiple
threads in an overlapping fashion. A multiprocessor is a more general method to employ Thread
Level Parallelism (TLP) since the multiprocessor has multiple independent threads operating at once in parallel. Multithreading shares most of the processor core among a set of threads, duplicating only a private state, such as the registers and program counter, which means multithreading does not duplicate the entire processor as a multiprocessor does [4]. Each thread has a separate register file, separate program counter, and separate page table, and the memory can be shared through the virtual memory mechanism. Switching between different threads should be relatively quick, and the hardware should support this quick switching.

The main three hardware multithreading approaches are Fine-grained, Coarse-grained, and Simultaneous multithreading (SMT). In fine-grained multithreading, switching between multiple threads takes place on each clock cycle. This allows the interleaving and execution of instructions from multiple threads. This interleaving is done in Round-Robin (RR) fashion which means skipping any threads that are stalled at that time [4]. In Coarse-grained multithreading, threads will be switched only on costly stalls, such as level two or three cache misses. Coarse-grained has a limited ability to overcome throughput losses, especially for short stalls. SMT is the most common implementation of multithreading [4]. SMT is a variation on fine-grained multithreading. Long latency events can be hidden by SMT using TLP, thereby increasing the usage of the functional units. Register renaming and dynamic scheduling with SMT will allow multiple instructions from independent threads to be executed without regard to the dependences among them [4].

2.1.6 Parallel Programming

The programmers try to make the software run faster by using parallel programming techniques. This is done using simultaneous processors to execute a process with multiple tasks or multiple processes. The idea of using multicore with parallel processing divides the big application, the complex algorithms, or the large amount of data into smaller tasks, then
distributes them among multiple cores, and as a result, the processing time should be reduced
to speed up big application processing. Applying this idea should minimize the sharing of
system resources and balance the execution among different processes [13]. The parallel
programming is very useful for video processing applications, floating point processing, speech
and voice processing, medical imaging, and other scientific applications with real time needs
to moderate data size. A parallel programming language must provide support for the three
basic aspects; specifying parallel execution, communicating between multiple threads, and
synchronization between threads [14]. The parallel processing provides two schemes for
control; Figure 2.4 shows the two-control model of parallel processing, master-slave model
and data flow model. The master-slave is the simplest model where the master is responsible
to generate subtasks and assign them to other works (slaves) in order to execute the specific
task. However, the data flow model executes the tasks as a directed graph of data flow.

Figure 2.4: Parallel Processing Control Models: (a) Master-Slave, (b) Data Flow [13]

The recommended tool for successful parallel processing is to have an advanced debug
feature, such as the Integrated Development Environment (IDE) as well as a dedicated Real
time Operating System (RTOS), such as SYS/BIOS. In addition, the standard APIs is also a
good tool to have to simplify the multicore programming and optimize the system capabilities
[13].
2.1.7 Comparison between Multicore and Multiprocessor Systems

Multicore and multiprocessor systems are both used to enhance the system performance by executing multiple tasks at the same time. A multicore system is a MIMD type of multiprocessor system [10]. A multicore system uses only one CPU, and this CPU contains more than one core on single chip. On the other hand, a multiprocessor uses more than one CPU, and each CPU is installed on a single chip. Thus, multicore will cost less. The multicore system will run more efficiently than the multiprocessor system where both systems have the same clock speed, number of CPUs and cores and RAM [5]. Table 2.1 shows a summary of the comparison between multicore and multiprocessor.

Table 2.1: Multicore versus Multiprocessor

<table>
<thead>
<tr>
<th></th>
<th>Multicore</th>
<th>Multiprocessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Physical CPUs</td>
<td>Single or Multiple</td>
<td>Multiple</td>
</tr>
<tr>
<td>Power consumption</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Traffic</td>
<td>Has less Traffic</td>
<td>Has High Traffic</td>
</tr>
<tr>
<td>Cost</td>
<td>Cheaper</td>
<td>Expensive</td>
</tr>
<tr>
<td>Execution Speed</td>
<td>Faster running for single program</td>
<td>Faster running multiple programs</td>
</tr>
<tr>
<td>Execution pattern</td>
<td>Execute multiple instructions</td>
<td>Execute multiple programs</td>
</tr>
<tr>
<td>Configuration</td>
<td>Not needed</td>
<td>Needs a little complex configuration</td>
</tr>
</tbody>
</table>

2.1.8 Performance Indicator Metrics

Using multicore systems with parallel programming should achieve improvement on the system performance. This performance improvement should be measured to figure out how the using of multicore with parallel programing is important.

Improving performance can be achieved by taking advantage of the most important method which is parallelism. Parallelism can be used at the system level by spreading the workload of the handling requests among the processors and disks. Scalability is the ability to expand the number of processors and the memory and disk. Spreading data across many disks
for parallel reads and writes enables DLP. Taking advantage of parallelism among instructions at the level of an individual processor is critical to achieving high performance, and pipelining is the simplest way to do this. Parallelism can be employed at the level of detailed digital design; for example, modern ALU use carry-lookahead, or set-associative caches [4].

The principle of locality is one of the most important program properties. Programs tend to reuse instructions and data they have used recently; a program spends 90% of its execution time in only 10% of the code. The idea of locality is that the predication of instructions and data that a program will use in near future is based on its accesses in the recent past. The locality has two types; spatial locality says that items whose addresses are near one another tend to be referenced close together in time. Temporal locality says that recently accessed items are likely to be accessed in the near future [4].

One of the performance measuring metrics is the speedup. Speedup is comparing the run time of the best sequential program versus the run time of the parallel program [15]. This ratio in equation (2.2) is known as the speed up and it shows how much faster a program runs when parallelized.

\[
SpeedUp = \frac{\text{run time of the best sequential program}}{\text{run time of the parallel program}}
\]  

(2.2)

The speedup metric is defined by Amdahl’s law in equation (2.3). In general, Amdahl’s law says that the speedup that can be gained by using a fraction, the overall speedup is the ration of the execution times [4]. Amdahl’s law depends on two factors; the first factor is the \( Fraction_{\text{enhanced}} \) which is the fraction of the computation time in the original computer that can be converted to take advantage of the enhancement, and this value is less than or equal to 1. The second factor is \( Speedup_{\text{enhanced}} \) which is the improvement gained by the enhanced execution mode, which means how much faster the task would run if the enhanced mode were
used for the entire program. This is equal to the time of the original mode over the time of the enhanced mode.

The law defined the program speed up as a function of the fraction of a program that is accelerated and by how much that fraction is accelerated [15]. Based on Amdahl’s law, the addition of processor cores is perfectly scalable. In other words, the serial portion of the code limits the maximum benefit a program can expect form parallelizing some portion of the code [15].

\[
\text{Speedup}_{\text{overall}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}})^{\frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}} \quad (2.3)
\]

In Amdahl’s law, the task speedup cannot be more than the reciprocal of 1 minus the fraction if an enhancement is only usable for a fraction of a task. Amdahl’s law can be considered as a guide to how much an enhancement will improve performance and how to distribute resources to improve cost-performance [4]. The goal is to spend resources proportionally to where time is spent [4]. Regrading to the number of cores, the speedup that will be achieved by n cores is based on how much of the program is parallel and how much is serial.

The speedup of parallelizing any computing problem is limited by the percentage of serial portion; this what is mentioned in Amdahl’s law. Gustafson’s law argues that, the law mentions that once the problem size is increased, the processor power also tends to increase. The drastic increase in the ratio of parallel-to-serial tasks in the computational load represents an equally dramatic increase in processing requirements, which means once the computing resources increase, the problem size also increases, and thus the serial portion became much smaller as a proportion of the overall problem [16]. Gustafson modified Amdahl’s law where the size of the overall problem should increase proportionally to the number of processors (N),
while the size of the serial portion \((s)\) of the problem should remain constant as number of cores increases, as shown in equation (2.4).

\[
\text{Speedup} = N + s(1 - N)
\]  

(2.4)

Superlinear speedup is defined as the computation using \(n\) processors could be more than the same computation performed on a uniprocessor [17]. The speedup will be more than \(n\). There are many resources of superlinear speedup; reducing overhead; increasing cache size where each processor has a local cache level 1 or level 2; Hiding latency in a communication in an interconnection network; using a randomized algorithms for problem solving where a parallel randomized algorithm can simultaneously achieve many solutions to the same problem [17]; the different speeds of memory inherent in distributed memory ensembles, the shift in time fraction spent on different-speed tasks [18]; utilizing resources more efficiently, and partitioning the data in such a way that all data are fitted in caches of multiple data nodes [19].

Amdahl’s law has some limitations due to the fixed workload assumption; problem size is fixed. The amount of parallelization in a program will not be the same for every action. One case that shows the limitation of Amdahl’s law is that once what is running is not being limited by the CPU, after increasing the number of cores, the performance gain will not take place. For example, if the RAM performance is preventing the program from running faster, adding more cores will not help even if the whole program is parallel. Amdahl’s law can accurately calculate the parallel efficiency once the comparison takes place for two or more CPUs that only use the same architecture [20].

Amdahl’s law does not apply to a critical section intensive workload such as databases. Critical section is a portion of code when one thread can execute at a given time and other threads wait to execute. Thus, a set of threads will be serialized. In some programs, there is a region of limited parallelism due to the contention of thread creation/deletion or hardware
resources overhead. Amdahl’s law does not apply to these cases [21]. Gustafson’s law assumes all the world being infinitely parallel or completely serial. As Amdahl’s law, Gustafson’s law does not account for the creation/deletion overhead of threads. Gustafson’s law does not account for the critical sections as a type of serial portions [21].

2.1.9 Parallel Programming Platforms

Multiprocessing programming can be implemented and debugged using an Application Programming Interface (API) that supports multi-platform shared memory multiprocessing programming. Python multiprocessing module, Open Multi-Processing (OpenMP), and CUDA are discussed in this section.

Python has libraries used for the programming of solutions employing either multiple CPUs or multicore CPUs in a SMP or shared memory environment, or potentially huge numbers of computers in a cluster or grid environment [22]. Python has a built-in multiprocessing module which helps the programmer to write a parallel program to run several tasks at the same time on a multicore CPU. The multiprocessing module in Python is a package that supports spawning processes using an API. The parallelized programming code can be written in relatively simple code segments using the Python multiprocessing module. Using a multiprocessing module in Python avoids the Global Interpreter Lock (GIL) problem in threads by using subprocesses instead of threads [23].

A parallel application can be built in Python using different options. The main three multiprocessing classes in Python are Process, Queue, and Lock [23]. A process in Python is an abstraction that sets up another (Python) process, provides code to run, and a way for the parent application to control execution. The program in execution can be managed by means of a process [24]. The first step in the multiprocessing module in Python is importing the process class.
Each process should have a target, for example, a specific function to run tasks. The process should start by calling the `process.start`. The child process will sit idle and not terminate without using the `process.join`. The number of processes can be as the programmer needs, but there is a limit on adding more processes for increasing performance, and this is not a fact for all computers [23]. A queue object is a First in First Out (FIFO) data structure, or may operate as a process/thread safe [23]. A queue object is used for communication among the processes. The processes in Python are communicated by a message passing paradigm [24]. The message passing paradigm is based on the lack of synchronizing mechanisms as copies of data are exchanged among processes [24]. Queues are especially useful when passed as a parameter to a process' target function to enable the process to consume or return data [23]. A lock object is used by the code to claim the lock, once the processes are executing similar codes, the lock object blocking other processes until the process has finished and releases the lock [23].

Python became very popular very quickly; the simplicity and the code readability are the main reasons for the popularity. It enables the programmer to express ideas in fewer lines of code without reducing readability [25]. Python is compared with other interpreted programming languages such as Java or Perl. In this comparison, Python programs are typically 3-5 times shorter than equivalent Java programs, and it is often 5-10 times shorter than equivalent C++ code [26]. Time is not wasted on declaring the types of variable or arguments, since Python has a powerful dictionary types and polymorphic list [26]. The common programming methodologies are supported in Python, such as object-oriented programming and data structure design. The code in Python is readable by providing an elegant but not overly cryptic notation [26]. Everything in Python is an object, and Python has dynamic typing and binding. In this dissertation, Python is used because of the ease of use of the multiprocessing module and the compatibility with Open Source Computing Vision (OpenCV). OpenCV-Python is a library of Python bindings designed to solve computer vision problems. OpenCV-
Python is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language [25]. Numpy, which is the library for numerical operations with a MATLAB, can be used by OpenCV-Python. All the OpenCV array structures are converted to and from Numpy arrays, which means the ease of integration with other libraries that use Numpy such as Matplotlib and Scipy. Matplotlib is a comprehensive 2D plotting. Scipy is a fundamental library for scientific computation [25]. OpenCV provides image processing operations, video analysis, feature detection and description, object detection, core operations such as counting the clock cycles and the execution time.

OpenMP supports multi-platform shared memory parallel programming in C, C++, and Fortran. OpenMP is primarily designed for shared memory multiprocessors [14]. A directive-based approach for supporting parallelism takes place in OpenMP. A set of compiler directives are used in OpenMP to express shared memory parallelism. These directives may be offered for Fortran, C, and C++ (directives are referred to as “pragmas”) [14]. A small set of runtime library routines and environment variables are included in OpenMP. The directive-based language extensions, the runtime library routines, and the environment variables are taken together to define the API. The purpose of OpenMP is to provide a standard and portable API for writing shared memory parallel programs [14]. A fork and join model is the basic operation of OpenMP for parallel execution. OpenMP programs begin as a single process, called the master thread. The master thread executes sequentially until a parallel region is encountered. the fork process starts where the master thread forks into several parallel worker threads. Worker threads execute the instructions in the parallel region. Once they reach the end of the parallel region, the threads synchronize and join to become the single thread again [27]. OpenMP has disadvantages; the scalability is limited by memory architecture, and OpenMP is available only on SMP systems [28].
Compute Unified Device Architecture (CUDA) is a parallel programming platform created by NVIDIA. CUDA is implemented by using a Graphics Processing Unit (GPU). The computing performance increases dramatically by CUDA; this can happen by using the power of the GPU. CUDA has a toolkit which includes a compiler, math libraries and tools for debugging and optimizing the application performance [29]. GPU computing is the running of the sequential part of GPU-accelerated applications workload on the CPU while accelerating parallel processing on the GPU [29]. A unified shared pipeline is included in the CUDA architecture; this allows each ALU on the chip to be organized by a program intended to perform general purpose computations. The execution units on the GPU were allowed read and write access to the shared memory [30]. Applications of CUDA are in medical imaging, computational fluid dynamics, and environmental science [30].

2.2 Solar Energy System

Solar energy is produced by capturing the power of the sun. Solar energy is the process of generating electricity or heat from the sun’s energy for human use. The energy of the sun is renewable, clean, and costs nothing [2]. Solar energy can be a more proficient energy source and an alternative to fossil fuels. Solar energy is used to provide power for different usages, such as appliances, buildings, and homes. Solar energy generates green and cheap electricity from sunlight using solar panels [2]. Solar energy can be described in two different forms; active or passive. Active solar energy produces electricity from sun light. Active solar energy uses the solar panels to capture the energy of the sun; then the energy is converted into Direct Current (DC) or Alternating Current (AC) [2]. Passive solar energy does not use solar panels; the building or structures are constructed to capture the power of the sun using tanks or windows, which heat water or homes, and it cannot produce electricity. The most popular type of solar energy system is Photovoltaic (PV), and the research is going on in this field.
Figure 2.5 shows the main components of a solar energy system. Some arrays are sets of special tracking devices to follow sunlight all day long in order to improve the solar energy system efficiency. The solar energy system is composed of a PV panel which absorbs the light from the sun, which generates the DC and feeds it to the Maximum Power Point Tracking (MPPT). The battery is used to charge the DC and use it later as a load. The inverter is used to convert the DC to AC and feed all AC loads.

2.2.1 Photovoltaic (PV) Cells

Photovoltaic (PV) generates electricity from sunlight. A PV module has individual PV cells, and they are arranged and grouped in an array as shown in Figure 2.6. Silicon is the semiconductor material that used for the majority of solar cell production because of some advantages: silicon can be easily found in nature, it is environment friendly, it does not pollute, and it can be easily melted [31].

![Solar Energy System Components](image)

Figure 2.5: Solar Energy System Components
The solar cell is the main part which is able to capture light from the sun. Basically, a solar cell is made of semiconductor materials such as silicon. Figure 2.7 shows the construction of the PV cell [32]. Each solar cell has two wafers of doped silicon in contact, p-type and n-type, in order to form a junction, and each wafer has an electrical connection. Solar cells are wired together as a module to form a panel which is then encapsulated to protect the cells from the weather. All these cells are sited on a tough backing plate, and the grid of electrical connections lies below and above the cells of the cells. Solar cells are connected in series where electrical connection strips will go from the bottom of one cell to the top of the next cell. A non-reflective layer will be above in order to increase light absorption. A layer of tough glass will be on the top, and the whole structure will be in an aluminum frame.
The light strikes the solar cell; a certain amount of the light is absorbed, which means the energy of the absorbed light is transferred to the solar cell. Thus, the electrons will flow freely because the energy will knock electron loose [33]. Solar cells have electric fields which act to force electrons freed by light absorption to flow in a certain direction, which sources a current [33]. This current could be used externally by placing metal contacts on the top and bottom of the solar cell. The solar cell’s voltage, which is a result of its built-in electric fields, together with the current provides the power that can be produced by the solar cell.

The electron-hole pairs become free when light’s energy hits the solar cell. One electron and one hole will free by one photon with enough energy [33]. When a free electron and a free hole happen an electric field, the electron will be sent to the N side and the hole will be sent to the P side, then the electrical neutrality will be disrupted and with the existence of an external current path, electrons will flow along the path to the original P side to unite with holes that inherent the electric field sent there [33]. The current is provided by the electron flow, and the voltage is provided by the solar cell’s electric field.

In addition, the Photovoltaic effect is the main working principle of solar cells, for example the generation of a potential difference at the junction of two different materials in
response to solar radiation. Once the material absorbs light with a frequency above a material-
dependent threshold, the electrons are emitted, and this is closely related to the photoelectric
effect [34]. The photovoltaic effect can be divided into three basic processes [34]:

- **Generation of charge carriers due to the absorption of photons in the materials that form**
a junction: once material absorbs a photon, its energy is used to excite an electron from
an initial energy level to a higher energy level. Absorption of a photon takes place if the
difference between the high energy level and initial energy level is equal to the
photon energy. Once an electron is excited from the initial energy level to a higher
energy level, a void is created at the initial energy level. This void behaves like a particle
with a positive elementary charge and is called a hole. Figure 2.8 (1) shows the creation
of an electron-hole pair once the absorption of photon is taking place [34].

- **Subsequent separation of the photo-generated charge carriers in the junction:** in the
recombination processes of the electron-hole pair, the electron will fall back to the
initial energy level, as shown in Figure 2.8 (2). In this case, the energy will be emitted
as a radiative recombination (photon) or nonradiative recombination (transferred to
other electrons of holes or lattice vibrations. Performing work in an external circuit,
using the energy stored in the electron-hole pair, needs the equivalent of semipermeable
membranes on both sides of the absorber, and as a result electrons only can flow out
through one membrane and holes only can flow out through the other membrane as
shown in Figure 2.8 (3). These virtual membranes are formed by voltage potential of
n-p type materials in solar cells. In the solar cell, the electrons and holes reach the n-p
regions before they can be released as a flow of electrons or holes recombine.

- **Collection of the photo-generated charge carriers at the terminals of the junction:**
electrical contacts are used to extract the charge carriers from the solar cells so that they
can perform work in an external circuit as shown in Figure 2.8 (4). The electrical energy
is produced from the continuous solar generation of the electron-hole pairs and inherent electric field of the device. When an external circuit is connected; the electrons will flow and combine with holes at a metal absorber interface after they pass through the circuit, as shown in Figure 2.8 (5).

Figure 2.8: A Simple Solar Cell Model [34]: (1) Generation of an electron-hole pair by absorption of a photon. (2) Combination of electrons and holes, (3) Electrons and holes are separated by semi-separated membranes, (4) Driving an electrical circuit by separating process, and (5) The electrons will recombine with holes after the electrons passed through the circuit.

A solar energy system can be supplemented with additional technologies in order to increase power production efficiency which supplies many devices and appliances and meets the different needs of different applications and customer needs such as the following technologies [35]:

- Concentrating solar power (CSP): using reflective devices to concentrate the sun’s energy; heat will be produced, then used to generate electricity.

- Transpired Solar Collectors: preheat ventilation air for the building using solar energy.

- Solar water heating systems: Heat working fluid using a solar collector which faces the sun; this can be used to heat water.
2.2.2 Solar Cell Types

Solar cells include three primary categories; thin film PV, silicon based PV, and emerging PV. Commercially and commonly available solar cells are thin film and silicon. These include less expensive thin film, poly-crystalline Si and mono-crystalline which is the most expensive one. Emerging solar cells are under development. Solar cells can be classified based on the basic material used. There are three generations: the first generation uses the crystalline silicon (c-Si); single or multi crystalline, and this generation is fully commercial. The second generation includes three families; they are based on thin film, micromorph silicon (a-Si/μc-Si), amorphous (a-Si), Cadmium-Telluride (CdTe), Copper-IndiumGallium-Diselenide (CIGS), and CopperIndium-Selenide (CIS). The last generation includes concentrating PV (CPV) and organic PV which are the emerging solar cells; that is still under development [36].

2.2.2.1 Crystalline Silicon Cells (c_Si)

The first generation of solar cells is based on crystalline silicon. There are three types of crystalline silicon cells; this classification depends on how the Si wafers are made, and the three types are the following:

Multi crystalline silicon (mc-Si): multi crystalline Si is called poly-crystalline Si. The silicon wafer can be made by casting and cooling molten silicon, then by a cutting process, and the final product is called the poly-crystalline Si [37]. The process of manufacturing poly-crystalline Si is based on using a silicon wafer, which means this process is much less expensive than mono-crystalline Si. The poly-crystalline has low efficiency, but the lower cost makes it commercially applicable.

Single crystalline silicon (sc-Si): Single crystalline Si is called mono-crystalline Si. The single crystal silicon wafers, with high purity silicon, are used to make the mono-crystalline Si.
It has a high efficiency and a long lifetime, which means it is suitable for residential use. It is relatively expensive because it is made from single silicon wafers [37].

2.2.2.2 Thin-Film Solar Cells

After intensive research and development for 20 years, an alternative to silicon solar cells exists, the thin film. It has lower-cost solar conversion [36] and [37]. Thin film is cheaper to manufacture because it has simple mass-production, and it has good flexibility which makes it easy to apply to many new applications. Thin film has better performance in the shade and at high temperature. On the other hand, thin film has fast degradation and low efficiency and this makes thin film less suitable for residential applications [37]. There are three main types of thin film that have been commercially developed: Cadmium Telluride (CdTe), Amorphous Silicon (a-Si), and Copper-Indium-Selenide (CSI and CIGS) [36].

Cadmium Telluride solar cells (CdTe): The main two materials are cadmium and tellurium. Copper processing produces tellurium, and zinc mining produces cadmium. Figure 2.9 shows the structure of CdTe solar cells. CdTe exceeds the cost-efficiency of c-SI solar panels, and CdTe operates in the efficiency range of 9-11% [37]. CdTe has higher cell efficiencies and a lower production cost than other thin-film solar cells [36]. On the other hand, CdTe has a problem--the produced amount of tellurium is far lower than cadmium. Cadmium has limitations in its use because there are issues around its toxicity [36].
Amorphous Silicon (a-Si): One type of the Thin-film solar cells type is amorphous silicon (a-Si). Figure 2.10 shows the structure of a-Si solar cells. A-Si has the ability to deposit in very large layers and is cheap, which reduces the manufacturing costs, by using continuous deposition techniques [36]. A-Si is suitable for curved and flat surfaces, for example as facades and roofs, and it can be used in portable devices [36] and [37]. The most popular form of a-Si is the hydrogenated (a-Si:H), and it has low efficiency because of the Staebler-Wronski effect [37]. This issue is solved by building double or triple junction solar cells, called multi-junction.

A-Si has issues: it has poor doping, and the Si layer quality is detrimentally affected by doping. A-Si monitory carriers have small diffusion lengths. For these reasons, the previous a-Si is not suitable with the normal structure of p-n junction. The solution of this issue is that a-Si should use p-i-n junction structure [37].
Copper Indium Gallium Selenide (CIGS): This thin-film solar cell type uses a direct bandgap semiconductor, it is CuInXGa1-XSe2 [37]. CIGS has a high absorption coefficient which makes it easy to deposit on plastic or glass layers. Figure 2.11 shows the structure of CIGS solar cell. It has a potential efficiency which is better than in a-Si. CIGS is more ecofriendly than CdTe because CIGS contains fewer toxic materials [37].

2.2.2.3 Emerging Solar Cells

The third generation of the solar cells is under development study. There are three types; Concentrating PV (CPV), Organic solar cells, and Dye-sensitized solar cells (DSSC). The CPV
is used to concentrate the solar radiation onto very small solar cells made of semiconductor material [36]. Using a sun tracking system, which orients the CPV modules toward the sun, should maximize the electricity generation [36]. Organic solar cells contain polymer or organic materials. The main advantages of an organic solar cell are that they are cheap and flexible. The changing of the molecular composition can simply modify the organic solar cell. On the other hand, the organic solar cell has a low efficiency [37]. DSSC is a photoelectrochemical system. The semiconductor formed between a photo-sensitized anode and an electrolyte is the basis of the DSSC solar cell. It is an inexpensive alternative for silicon solar cells [37]. On the other hand, DSSC has low efficiency because it has narrow spectral coverage. The solution uses the nanocrystalline semiconductors [36].

2.2.3 PV Modules Efficiency and Cost

In order to enhance efficiency, it is very important to detect a fault as soon as possible to minimize harmful impact and save energy and money. Once the fault is detected in its earlier stage, the power consumption and the efficiency can be improved by taking the appropriate action and prevent any further damage such as fire or any other electric dangers that may arise and comprise safety. In the three recent decades, solar energy has made fast progress. The PV module’s efficiency is the ratio between the incident power of the solar radiation and the power produced by the module. The solar panel efficiency can be determined in equation (2.5) [39], where $I_{mp}$ is the maximum power point current; $V_{mp}$ is the maximum power point voltage; $G$ is the incident irradiance; and $A$ is the module area.

$$\eta = \frac{I_{mp}V_{mp}}{G \times A}$$ (2.5)

Solar cell efficiency is divided into four individual efficiencies; thermodynamic efficiency, reflectance efficiency, charge carrier separation efficiency, and conductive
efficiency. The product of each of these efficiencies is the overall efficiency [40]. Figure 2.12 shows the solar efficiency report from National Renewable Energy Laboratory.

![Figure 2.12: Reported Timeline of Solar Cell Energy Conversion Efficiencies [40]](image)

Table 2.2 shows the comparison between the different solar cell technologies and the difference in efficiency and market share.

**Table 2.2: Summary of Efficiency and Market for PV Technologies [36]**

<table>
<thead>
<tr>
<th>Technology</th>
<th>1&lt;sup&gt;st&lt;/sup&gt; Confirmed solar cell efficiency at AM 1.5</th>
<th>2&lt;sup&gt;nd&lt;/sup&gt; Commercial PV module efficiency at AM 1.5</th>
<th>3&lt;sup&gt;rd&lt;/sup&gt; Confirmed maximum PV module efficiency</th>
<th>Market share in 2009</th>
<th>Market share in 2010</th>
<th>Maximum PV module output Watt</th>
<th>PV module size m&lt;sup&gt;2&lt;/sup&gt;</th>
<th>Area needed per kW m&lt;sup&gt;2&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>mc-Si</td>
<td>14-18</td>
<td>13-15</td>
<td>16</td>
<td>3</td>
<td>2</td>
<td>320</td>
<td>1.4-2.5</td>
<td>8</td>
</tr>
<tr>
<td>Se-Si</td>
<td>20-24</td>
<td>15-19</td>
<td>23</td>
<td>83</td>
<td>87</td>
<td>120</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>CdTe</td>
<td>8-10</td>
<td>8-11</td>
<td>11.2</td>
<td>1</td>
<td>2</td>
<td>120</td>
<td>0.72</td>
<td>11</td>
</tr>
<tr>
<td>a-Si</td>
<td>6-8</td>
<td>5-8</td>
<td>7.1/10</td>
<td>13</td>
<td>9</td>
<td>300</td>
<td>1.4</td>
<td>15</td>
</tr>
<tr>
<td>CIGS</td>
<td>10-12</td>
<td>7-11</td>
<td>12.1</td>
<td>1</td>
<td>9</td>
<td>120</td>
<td>0.6-1.0</td>
<td>10</td>
</tr>
<tr>
<td>CPV</td>
<td>36-41</td>
<td>25-30</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OPV</td>
<td>8.3</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DSSC</td>
<td>8.8</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
There are parameters that affect the cost of the electricity generated by a PV module; the first parameter is the capital cost (CAPEX). It consists of the Balance of System (BOS) cost and the PV module cost. The raw material in the PV array, cell processing and manufacturing, silicon prices and assembly costs are determined by the PV module cost [36]. The BOS cost includes the cost of the electrical system, the battery and other storage system costs, and the structural system cost. The other parameters that affect the cost of the electricity generated by the PV module are the level of the irradiation, the cell’s efficiency, and the variable costs (OPEX). The cost reduction can be achieved by improving the cost of finance and efficiency, and the capital cost [36]. Table 2.3 shows the summary of the worldwide market price.

Table 2.3: The Worldwide Market Price of PV Modules, Q4 2009 to Q1 2012[36]

<table>
<thead>
<tr>
<th>PV module supplier</th>
<th>Factory-gate price in Europe (USD/Watt)</th>
<th>Factory-gate price in the United States (USD/Watt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>High efficiency c-Si</td>
<td>2.45</td>
<td>2.22</td>
</tr>
<tr>
<td>Japanese/Western c-Si</td>
<td>1.98</td>
<td>1.81</td>
</tr>
<tr>
<td>Chinese major c-Si</td>
<td>1.51</td>
<td>1.42</td>
</tr>
<tr>
<td>Emerging economies c-Si</td>
<td>1.45</td>
<td>1.35</td>
</tr>
<tr>
<td>High efficiency thin film (via distribution, First Solar)</td>
<td>1.26</td>
<td>1.30</td>
</tr>
</tbody>
</table>

Notes: Based on short-term contract prices for quantities of 0.5 MW or more. Spot market prices are typically more volatile. * Sunpower and Sharp ** Sharp, Kyocera, Solarworld and Schott *** Suntech, Yingli, Trina Solar and Green Energy etc. **** Chinese, Korean and Indian manufacturers.

2.3. Fault Types in Solar Panels

Solar energy systems are used to produce power and provide electricity to homes, buildings, and more. In order to get more efficient power and higher performance for the solar energy system, it is very important to detect a fault as soon as possible to minimize harmful
impact and save energy and money. The common faults in the PV module types are summarized in the following points [41], and all these faults can be considered as manufacturing faults:

- **Delamination** can occur when the adhesion between the glass, encapsulant, active layers, and back layers becomes weak.
- **Back sheet adhesion loss**
- **Junction box failure** takes place when there are no reliable soldering contacts of the string interconnects; this could cause high a resistance and consequent heating
- **Frame breakage.**

The most common failures due to external causes are summarized in the following:

- Clamping
- Connector failures
- Lighting
- Transport and installation

In the following, some faults in a solar panel are categorized as [42]:

- **Hot spots caused by the panel acting as a load:** hot spots can be taken as a sign of heat loss from the solar panel due to some electrical problems, for example in bypass diode, or in shunt resistor.
- **Defects in the anti-reflective coating:** the color of the module can be changed due to a change in the coating layer. The light that reaches the cells may be lower [42], and as a result the antireflection properties may suffer changes.
- **Bubbles in the solar modules:** when the adherence between tedlar and the back side of the module has been lost, bubbles will develop. As a result, overheating will take place which means a reduction of cell life.
➢ **Delamination over cells and interconnections**: it occurs when the tedlar adherence is lost with the subsequent unstuck layers, and as a result the water penetration will take place inside the module, and this is considered as a major problem.

➢ **Browning and yellowing**: it is caused by water penetration inside the module, because of lack of adherence between the tedlar and the module.

➢ **Newly cracked cells**: the thickness of the solar cell has been changed from 300 µm to 200 µm. In other words, the cells become more susceptible and fragile which means they are easily breakable on manipulation or storage of the module.

The following list consists some of the considered electrical faults in the solar panels [43], [44] and [45]:

➢ The shunting in cells’ bypass diode functionality
➢ Erroneous bypass diode functionality
➢ Reverse-biased heating
➢ High series resistance.
➢ Nonlinear weak diodes
➢ Resistive solder bonds
➢ Hot spots in modules

There are three types of faults in PV arrays; these are the line-line fault, open circuit fault, and ground fault. The faults inside PV arrays usually cause an overcurrent which damages PV components [46]. A line-line fault occurs when an accidental connection goes through a low resistance path between two points in the solar panel [46]. A line-line fault may occur between two adjacent strings or between two points on the same string as shown in Figure 2.13.

The potential difference between the connected points influences the magnitude of the line-line fault, so the higher potential difference means higher feedback current results, and tripping the over-current protection devices (OCPD) will increase [47]. A line-line fault can be considered as a ground fault if one of the points is on the Equipment Grounding Conductor
The Line-Line fault may not be detected once the current flow through one of the current carrying conductors (CCC) is less than the rated current of the OCPDs. This will reduce the system efficiency [48].

In the open fault, or arc fault, a current might be established due to insulation breakdown in the Current Carrying Conductors [47]. An open fault is generated in CCC from cell damage, rodent damage, solder disjoint, or corrosion of a connector. If this fault occurs at different potentials and between two adjacent conductors, it is called a parallel open fault, as shown in Figure 2.14 [47]. High frequency noise in the DC current of the PV string is produced due to the open fault. An additional sudden voltage/current drop inside the PV array occurs because of parallel open fault; therefore, the difference between parallel and series open faults can be observed by the sudden voltage/current drop associated with increased noise in the DC current. On the other hand, by disconnecting the PV array from the inverter by opening one of the terminals of each PV string, the series open faults can be distinguished [47].
Once one of the CCCs is functionally connected to ground, the PV system can be used as a grounding system. Therefore, there are two types of PV systems: ungrounded and grounded [47]. A ground fault generates an accidental low impedance path between the ground and one of the CCCs as shown in Figure 2.15. Using a ground fault detection interrupt (GFDI) device helps to detect the ground faults and interrupt the fault current [47]. Whenever the fault current amplitude continues longer than a predefined minimum time duration or higher than a threshold limit, GFDI devices either open or melt the current path [47].
Currently, the most used technique to detect faults in solar energy systems is concentrated on an inverter’s failures [49] and [50]. This method is based on detecting unexpected power loss by comparing the output with benchmark values or reference values. Once a large difference is detected, the attention message can be generated to the control unit declaring a fault, but with no information on its location.

2.4 Infrared Technology for Non-destructive Testing

The electromagnetic spectrum is a scale classifying the different forms of electromagnetic radiation [51], such as X-rays, gamma rays, radio waves, microwaves, visible rays, and invisible rays. Because the human vision system can see just the visible light, the use of infrared imaging becomes more necessary. The technique of capturing invisible infrared images and converting them into visible images is called infrared imaging. Infrared imagers and cameras have special sensors that do not need visible light to see in infrared light [51].

All warm objects with a temperature above absolute zero can produce infrared radiation because there is no molecular and atomic activity at absolute zero [51]. Molecular and atomic activity increases when the temperature increases, which means more thermal radiation and heat is produced, and more infrared radiation is emitted [51].

The targeted objects or the reflected radiation in the infrared imaging emit the radiation. Using sunlight for illumination will cause reflected radiation, or the imaging device may have infrared illuminator lasers with LED. These infrared waves absorb or reflect from any object that comes in the range of these invisible illuminators [51].

Using long wave thermal infrared imagers will help to detect and pick up the infrared radiation reflected from or emitted by warm objects. The infrared rays are directed onto an array of infrared sensors, and there can be many thousand sensors on the sensor array [51]. The infrared energy is transformed into electrical signals and then converted into an image.
Infrared light has many uses because it has the ability to cross areas that cannot be reached by visible light, and it reveals unclear objects. The military developed it to produce night vision cameras and gun sights [51]. Infrared imaging is used in many applications, such as helping firefighters and police to catch criminals in the dark, or to rescue people lost at night and in dark places. Using infrared imaging enables technicians to locate leaking chemicals as well as overheated or under-heated parts, which helps to eliminate potential hazards. Infrared imaging helps researchers in the wild to study animals in their habitat at night. On the other hand, infrared imaging can be used for medical diagnostic purposes; it is used to perform body scans. Also, satellites use infrared imaging to study earth conditions.

In 1965, the first thermal imaging camera was sold for line inspection of high voltage power lines [52]. The thermal imaging cameras became like photographic camera and supported video records which provided real time high-resolution images. The building industry is one of the sectors that soon began to use the thermal cameras, providing valuable information that is impossible to capture with a normal photographic camera [52]. A thermal camera can capture the energy loss from a building.

The intensity of radiation in the infrared part of the electromagnetic spectrum can be recorded by the thermal imaging camera, and this radiation can be converted to a visible image [52]. The heat or the thermal radiation is the primary source of infrared radiation. It lies between the microwave and the visible portions of the electromagnetic spectrum [52]. The radiation of an object can be emitted in the infrared region if that object has a temperature above zero (0 Kelvin or (-273.15) degrees Celsius) [52].

The use of thermal imaging techniques needs well-defined guidelines to determine the efficiency of the PV module with thin-film modules or crystalline solar cells in the field [53]. There are many guidelines. The first one ensures that there is a good thermal contrast for
accurate thermo-graphic measurement which is achieved from sufficient energy from the sun; a solar irradiance of \(500\text{W/m}^2\) or higher is needed, and optimally \(700\text{W/m}^2\). The second guideline is the sky should be clear because clouds produce interference through reflections and reduce solar irradiance. Furthermore, informative images can be also taken with a cloudy sky using a camera of sufficient thermal sensitivity. Also the airflow may cause convective cooling which reduces the thermal gradient. Once the potential thermal contrast is higher, then the air temperature is cooler, so early morning inspection is the best option [53].

The glass covering the panel is not transparent, so the solar cell is inspected from the front, which enables the camera to see the heat distribution on the glass surface, but only indirectly the thermal performance of the underlying cells. Typically, the thermal camera has an uncooled micro-bolometer detector; it is sensitive in the 8-14 µm waveband. Based on that, the measured temperature difference on the solar panel glass is small, and using a thermal camera with thermal sensitivity of \(< 80\text{mK}\) will make this difference visible [53].

The aluminum framework which holds the PV module shows a cold area on a thermal image, because that framework reflects the thermal radiation emitted by the sky. In other words, the framework temperature will be shown on the thermal camera below 0ºC. The minimum and maximum temperature can be adapted by using the camera’s histogram equalization, and there may be many small anomalies which will not be directly observed. Therefore, we need to achieve clear contrast. Using Digital Detail Enhancement (DDE) will automatically optimize image contrast in high dynamic range scenarios [53].

The camera with DDE will be well-suited for solar panel inspection. In order to tag faulty modules over a large area, a GPS can be used for that purpose, for example inspection of a solar garden [53]. One of the most important features that needs to be in the thermal camera is fusion. It is the feature that allows the thermal image to be superimposed with a visual image
in order to give more clarity. One important technology used to take details from the visual image to improve the thermal image is the Multi Spectral Dynamic Imaging (MSX) technology. Solar inspection with MSX will be quicker and more effective, which reduces the time and cost [53].

The ability of the material’s surface to emit energy by radiation is called emissivity; therefore, this factor is necessary to any thermal measurement and can be pre-programmed with professional thermal imaging cameras. The thermal image of the glass surface will pick up the radiated temperature of surrounding objects such as the camera, so it is important to pay particular attention when working with glass because it will give false results for measurement errors and hotspots [53]. This problem can be avoided or at least minimized by adjusting the viewing angle, and a tripod will prove a useful accessory. Larger areas can be inspected with a single camera if it provides a longer distance measurement. In order to achieve a clear infrared image over a long distance, the required minimum image resolution is 320 x 240 pixels, and the better resolution is 640 x 480 pixels [53]. Using an interchangeable lens achieves high resolution which enables the operator to switch to a telephoto lens for long distance observations [53].

Every solar cell on a PV module must be working in order to get maximum power generation, system life and the best return on the investment [53]. Thermal imaging allows abnormalities to be seen clearly and shows physical damage; it can also be used to scan the installed solar panels during normal operation. It can scan a large area in minutes, which becomes a highly time-efficient process [53]. Thermal imaging cameras have been used in development and research of solar panel technology for many years, using the cooled thermal camera. On the other hand, the uncooled cameras are typically used in quality control, maintenance, and post-production [53].
2.5 Concluding Remarks

Parallel processing on the multicore and multiprocessor systems is widely used in order to reduce the execution times for the running tasks and processes. The parallel programming on such systems enables an algorithm to run multiple tasks in parallel on multicore or multiprocessor systems, which increases the speed of the execution and improves the performance. Python provides a rich multiprocessing module which provides an easy way to run multiple processes at the same time. Indeed, OpenCV-Python provides a nice platform for computer vision processes. Solar panels are used widely (across the whole world) for electricity generation. The real operations of the PV systems occur without any supervisory mechanism, and the systems may have a low efficiency because of many obstacles. The fault detection in a solar energy system should be automated and should have a timely alert system, where the prompt identification of faulty PV cells will allow for better system operation, and enhance safety. The solar energy system has been discussed in this chapter. Furthermore, the main components of the system and the main types of solar cells are discussed. Different types of faults that may take a place in the solar system were discussed in detail. Infrared technology was explained, and this technology has been used in the dissertation as technology for data recording for the solar energy system under real time operations.
CHAPTER III
PERTINENT LITERATURE

Many researchers in computer engineering fields have used and developed the multicore and multiprocessor systems for many applications in order to reduce the execution time and improve the performance. In the image processing field, many algorithms were implemented using a multicore system. Fault detection in the solar panel has been implemented in different ways in recent years. Infrared technology has been used in different applications and sectors for the inspection process and validation in manufacturing and industry. In this chapter, I attempt to mention and survey the major works using multicore and multiprocessor for image processing. Furthermore, I mention different methods of fault detection in the solar panel using different techniques.

3.1 Real Time Image Processing

Many algorithms are used to analyze the images in order to find reliable and accurate results, and in the meantime, process the image in less time [54]. The image can be made more visible and clear by preprocessing it, and at the same stage, the algorithm parallelizing optimizes image processing time. The problems of image processing that require a large amount of processing time or need to handle a large amount of information can be solved by parallel computing [54]; for example, medical image processing requires lots of time and memory space to process, so parallel processing can be an efficient way to process these images. Parallel processing for images is based on dividing the input image into tasks and then processing these tasks simultaneously.
Saxena et al [54] represented the parallel implementation of sequential image processing algorithms using multicore architecture. For example, they presented noise reduction, histogram equalization, and segmentation using parallel processing. The main algorithm proposed by Saxena et al is based on dividing the input images into different tiles where the number of tiles is equal to the number of cores or the number of threads. Each tile will be processed by its core or thread with attention given to synchronization within the processor [54] using Intel Core i3-2350M Processor 2.30 GHz, 3 GB of RAM, Hard Disk Drive 320 GB Software. They used also MATLAB R2011a and JAVA JDK 1.6.0_21 and a 64-bit operating system. The image resolutions are 256x256, 256x768, and 128x843. The results show that the parallel processing is better than sequential processing by 1 time; the execution time is less than 2 times of the sequential programming. The results also show that for some algorithms the improvement reached 2 times.

The image segmentation process is one of the primary steps of extracting different objects that compose an image. The segmentation process is based on dividing the input image into meaningful regions. The large images need high computational time for the segmentation process [55]. Happ et al explored multicore processors in order to speed up the segmentation process of an image. They used parallel processing to improve the proposed segmentation algorithm by Baatz [56]. The main idea of the Baatz algorithm is splitting the image into tiles (regions) [56]. Each thread can process one tile which performs a local region growing use the sequential algorithm [55]. Once the image is divided into tiles and then the work divided in threads, these should impact the final segmentation results. Happ et al used the same proposed Baatz algorithm, and they used the multicore processor; they suggested that to achieve better performance. The number of threads should always be equal to the number of available cores [55]. The testing environment was on an Intel Core 2 Quad 2.40 GHz, 2 GB of RAM. The input images have three different sizes, 1000x1000, 2000x2000, and 2800x2800. The results show
speed ups to 2.5 times using 4 threads, and the speed up is improved around 1.5 time using 2 threads [55].

Kamalakannan et al proposed multithreaded color image processing algorithms including contrast enhancement using fuzzy method versus edge detection [57]. They proposed that the entire image can be partitioned into equal blocks using separate cores and perform simultaneous processing [57]. The testing environment was Core i5 Quad-core, and the input images were 10 images of different pixel size. The results show that the speed up improved by 3.4 times over a sequential approach using a four thread model [57].

Liu and Gao presented an implementation of the cubic convolution interpolation algorithm in images using parallel programming tools such as Threading Building Blocks (TBB) and OpenMP on a multicore processor platform [58]. They also presented a comparison between the sequential and parallel implementations and results. The results show that the use of Dual-core with the parallel implementation of the cubic algorithm improved the speed by 200%, and by using the Quad-core the speed up is improved about 400% compared with sequential implementation [58].

3.2 Investigation of Defects in Solar Cells

Several technologies have been proposed to identify problems in photovoltaic cells such as infrared thermal imaging (IR), photoluminescence imaging, electroluminescence imaging (EL) and current-voltage sensors. However, the non-destructive nature of thermal imaging has propelled many researchers to detect solar energy problems.

Typical device behavior can be investigated by understanding and measuring the temperature distribution [44]. Temperature distribution can be measured using an infrared imaging system such as thermal cameras. This tool can provide a suitable method for relative and absolute temperature distribution for large and small components with spatial resolution
and a high temperature [44]. Infrared images can be used to identify minute defects and thus an algorithm can be designed to provide a clear survey of the inside of the solar panel. Indeed, infrared images have been able to localize shunting in cells, bypass diode functionality, hot spots in modules, and batteries during charging, and temperature of electronic components [44].

3.2.1 Detect Shunting in PV Cell

A localized shunting path within a solar cell often occurs during cell manufacturing or due to field aging [44]. Humidity intrusion into modules causes field aging and it may result in shunting failures, especially in thin-film modules [44]. The junction diffusion process may include impurities on wafers, and this can happen during manufacturing [44]. Research uses locating and tracking of the shunting failures to investigate the affected spots, and infrared imaging can be used to detect these spots [44]. In order to localize heating using shunting spots, the current through the cell should be forced to flow in a reverse-biased condition [44]. In this case, current flows through any available shunt resistance which leads to determining the heating location. After that, infrared images can be taken after a current flow is introduced. It is recommended to record the images quickly because the image of temperature might bloom the masking because of heat transfer, and this will mask the exact location of the shunt [44]. Also, the ability of the infrared cameras to detect shunt paths is affected by the heat spread on the glass [44].

Thermo-reflectance image has high resolution and wavelength flexibility. Thermo-reflectance uses sub-micrometer spatial resolution to identify defects in solar cells [45]. It uses the relationship between the change in temperature and in the reflectivity of materials in order to give high resolution thermal images [59]. In other words, the change of temperature can be used to exploit the change in material reflectivity [45]. When the temperature variation is small,
the linear relationship between the change in temperature and in the reflectivity of materials can be used [45]. The visible light can be used in order to identify the change in reflectivity where the thermo reflectance coefficient is non-zero for most wavelengths. The spatial resolution of the thermal image can be increased with visible light because the diffraction limited spatial resolution is between 250nm and 400nm, which gives a more accurate peak temperature [45].

The glass and other obscure materials that covered the solar cell degrade the spatial resolution of IR imaging. The flexibility of the wavelength is important for a new thin-film solar cell with glass [45]. Obtaining images for the active region can be achieved by using visible light because it can pass through the glass. The thermal image can be gained using near-infrared light especially for silicon.

Thermo-reflectance has a very small effect of 1 to 10,000; therefore, obtaining a thermal image with good signal-to-noise ratio can be achieved by using the lock-in technique [45]. The propagation of device heating in a specific time can be obtained by taking a series of images [45]. The linear thermal response gets indications about the shunt defects, and in forward bias low carrier injection levels the weak diodes will be shown [45]. Thermo-reflectance images of thin-film show that in forward and reverse bias images the shunts exist; on the other hand, the weak diode is shown just in forward bias images [45]. The non-ideal diode characteristics and leak current are found in weak diodes which cause degrading in cell efficiency because the photo-current will flow through these spots [45].

3.2.2 Faults from Solder Bonds

The process of current flow through a module must go through the cell interconnect ribbons and through solder bonds [44]. Field aging may cause failed or resistive solder bonds; in this case the current should flow through a small number of solder bonds, which increases
the current intensity and helps to localize the failed solder bonds [44]. Use of infrared images helps to determine the location of the failures caused by solder bonds [44].

3.2.3 Heating from Reverse-Bias

The individual cells can heat up once they are connected in series and a short circuit may take place. This fault exists in silicon modules. The reverse-bias can occur when the module is short-circuited under outdoor solar illumination [44], because the lowest short-circuit current forced these cells to operate in reverse-bias; the result is that the cells heat up.

Minimizing reverse-biased heating and compensating for performance losses due to module shading can be achieved by using bypass diodes [44]. Also, the infrared imaging system can be used to test the bypass diode functionality during a diagnostic test or manufacturing process. On the other hand, an infrared imaging system is used to illustrate and employ the energy storage components which are located between the PV modules and battery banks. The batteries are charged by current support from a PV array, or the batteries supply current (power). In the charging process, the heat might be dissipated from the battery banks which makes a temperature conformity between individual cells [44]. The unnecessary power is dissipated from the resistive battery when the current is flowing. This can be shown as an abnormally hot area in the infrared image [44]. The non-uniform temperature and the cell-to-cell mismatching at different rates will take place because of the battery banks’ age [44].

3.3 Fault Detection in Solar Panel using Embedded Sensors

Chouder and Silvestre [49] presented an automatic supervision and fault detection method for a solar energy system based on power loss analysis. It calculates the main parameters of the solar energy system for monitoring data in real time while taking into account the environmental irradiance and temperature evolution. Their method focuses on the output power losses while present in the DC side of the solar energy system generator and captures
loss. This model defines two power loss types, thermal loss and miscellaneous loss. The system can generate a fault-signal indicator based on the values of these two loss types. These indicators show the current and voltage ratios. After analysing both, the fault-signal and the current/voltage ratios, the type of fault can be identified.

The automatic supervision and fault detection procedure, in [49], is able to detect most faults that can occur in the solar energy system. This model has a remote server, which means it supports centralization. It controls different PV systems in real time using real data received via the Internet, which means the PV system can be tested periodically and a report can also be directly generated in order to reduce computational and transmission costs. The computers are necessary to run supervision in real time for this model. This model needs to use sensors for temperature and irradiance. The other electrical characteristics, such as DC output and voltage, can be extracted from the electronics elements presented in the PV system variables. Regarding [49], most of the inverters and charge regulator offer the ability of recording this information, but it costs a lot; on the other hand, it is quick fault detection.

The model description is based on the following: the powerful tools for studying and analyzing PV systems are Matlab & Simulink and Pspice. They have advanced mathematical manipulation toolboxes which are useful in PV systems simulation. When the model is based on fault detection, it is important to choose a very accurate simulation tool such as Matlab because all the predicted variables will be compared with the measured and monitored variables, and then a decision is made about the normality/abnormality of the process behavior. Matlab provides solutions for automatic monitoring, data acquisition and communications protocol control. In this model the values of cell temperature and irradiance are set as input variables. Also in this model, the fault detection focuses on the detection of the faults generated by the DC part of the PV. The PV output is predicated based on five parameter equivalent
circuit models where the relationship between output current and voltage is shown in equation (3.1).

\[ I = I_{PH} - I_0 \left[ \exp \left( \frac{V + R_s I}{n V_t} \right) - 1 \right] - \left( \frac{V + R_s I}{R_{sh}} \right) \]  

(3.1)

Where: 
- \( I_{PH} \) cell photocurrent
- \( I_0 \) Diode reverse saturation current
- \( n \) Ideality factor
- \( R_s \) The series resistance
- \( R_{sh} \) The shunt resistance
- \( V_t \) The Threshold Voltage
- \( I \) and \( V \) The output of PV cell

In regards to [49], the effects of temperature in output voltage and current are incorporated. A nonlinear regression algorithm has been applied to both data sets, measured I–V data from the PV system and data generated by the previous model in order to minimize the quadratic equation (3.2).

\[ S(\theta) = \sum_{i=1}^{N} [I_i - I(V_i, 0)]^2 \]  

(3.2)

Where \( \theta = (I_0, I_{PH}, n, R_s, R_{sh}) \)

This extraction technique gives a set of parameters that will be used in the simulation of the PV system to supervise. The parameters’ values have been compared for applying the extraction method, using I–V measures taken at different irradiance and temperature levels, and manufactured data. Table 3.1 shows the comparison results for a single PV module Isofoton 106-12 PV [49].
Table 3.1: Isofoton 106-12 PV Module Characteristic Parameters [49]

<table>
<thead>
<tr>
<th></th>
<th>$R_s$ ($\Omega$)</th>
<th>$R_{sh}$ ($\Omega$)</th>
<th>$N$</th>
<th>$I_0$ (A)</th>
<th>$I_{PH}$ (A)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Extracted parameters</strong></td>
<td>0.33</td>
<td>198.88</td>
<td>1.038</td>
<td>1.297e-7</td>
<td>6.6</td>
</tr>
<tr>
<td><strong>Manufacturer data</strong></td>
<td>0.08</td>
<td>200</td>
<td>1.63</td>
<td>6.19e-7</td>
<td>6.54</td>
</tr>
</tbody>
</table>

After introducing the environmental irradiance and module temperature evolution into the simulation model, the simulation of the solar energy system can be described. This model has been tested using operational data from a grid-connected PV system of 3.2 kWp [49]. The environmental conditions such as module temperature and irradiance have to be linked to the simulation model. Regarding [49], the comparison was done between power evolutions from monitoring data and the output current and simulation results. The error values are shown in Table 3.2 to quantify the deviation of the predicted values [49].

Table 3.2: RMSE Between Measured and Simulation Results [49]

<table>
<thead>
<tr>
<th>RMSE%</th>
<th>Output DC</th>
<th>Output DC Voltage</th>
<th>Output DC Power</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.106</td>
<td>3.319</td>
<td>4.269</td>
</tr>
</tbody>
</table>

This method depends on the climate. It attempts to establish remote monitoring and fault detection in a small solar energy system. The disparity between optimum and actual was used to indicate the presence of some defects [60] and [61].

This idea reduces computational and simulation costs because the process is centralized; no climate sensors are needed, but other costs such as data loggers and communication costs are introduced. Regarding [49], the calculated system’s energy yields do
not have the same accuracy as yields calculated from real monitored data, irradiance on the PV modules surface, and temperature of solar cells.

Previous studies provided simulation tools to detect faults in the PV modules. These studies have not been concentrated on real time data, and they depend on the simulation results [62]. Most of these approaches do not use standard software tools [63] and [64] and are just focused on determinate incidences: inverter failure or shadowing [65], PV generator maximum power point, or PV module fails in determinate branches of the PV array [62].

Comparing current operation characteristics is one of the methods for fault detection in the solar energy system. Solar panels may have faults due to different reasons such as snow accumulation, shading, dust, cracked cells, the shunting in cells’ bypass diode functionality, bypass diode functionality, reverse-biased heating, high series resistance, nonlinear weak diodes, or resistive solder bonds, which affect productivity. Keeping track of these faults needs real-time monitoring and proper maintenance to ensure normal operation and highest possible efficiency. Fault detection in solar panels becomes more complicated with a large number of panels; because of that, it is hard to identify and locate the fault without costly physical inspection.

Detecting the array faults in the solar panels can be accomplished by comparing current operation characteristics with recorded reference characteristics that account for time, season and climate which will affect the performance. Regarding [50], due to the complexity of these numerous conditions, the need for reference profiles and the need for generating reference profiles, as well as the comparison of the parameters with the reference profiles, this process can be very complicated and the result could be inaccurate. The summary of the invention is in the following steps [50]:
1- A monitoring device to detect and identify a fault of a solar panel and an inverter in a solar array.

2- Generation of a normal profile by extracting median values of operation profiles from multiple solar panels in a solar array; comparison of an individual operation profile against a normal profile to determine a fault in a solar panel.

3- Detection of a fault in a combination of a solar panel and an inverter; and detection of a fault in a solar panel.

4- Identification of a fault in an inverter; and the storage of faulty profiles in the database for particular faults.

5- Comparison of an operation profile from a faulty solar panel with a number of faulty profiles in a database to identify the type of the fault; and generation and report of a message with a fault and a suggestion of a corrective action.

The main idea in this method is accessing an operation file for the current solar panels of the solar array, then generating a normal profile for that panel. The comparison between the normal profile and operation profile from each of the plurality of panels then determines what the difference is between them. Then monitoring devices decide if the fault exists or not [50].

One of the methods used to detect the fault in solar panels is reverse current fault. The idea is to detect the abnormal current flow [43]. The solar energy system combines multiple power resources to provide a single output, such as systems for combining the power generated by multiple solar panel arrays [43]. It generates power from multiple solar panels. In this case it is important to detect the faults generated by reverse currents [43]. The fault detection based on the current flow checks for the current flow in the opposite direction. Using a plurality of photovoltaic elements provides a system for sensing and interrupting a reverse current. Each conductor is associated with a sensor. The sensor is able to detect the current flow in the
respective conductor, and it is configured to provide an output of the detected direction of the current.

In addition, a plurality of switches may each be included in electrical communication with a respective sensor to receive the detected direction of the current. The switches may be configured to disconnect the current from the respective conductor of the first polarity responsive to the detected direction, which may be a reverse direction current flow. The switches may include a plurality of relays and/or may include normally open relays, and the sensors may include Hall Effect sensors. It is used to respond to a magnetic field, the Hall Effect sensor converts variations of the output voltage. The Hall sensor is used for speed detection, current sensing, switching, and positioning.

The receiving of the detected direction of the current can be through a comparator which is used in electrical communication with a specified sensor and switch [43]. The comparator is able to provide an output to a specific switch which is indicative of a current flow back toward the conductor; then the switch can be turned off in case of reverse current. In order to protect the circuit, the protection elements are connected with a specific conductor and may be able to create an open circuit once the current exceeds the threshold. Using the busbar with each conductor, the direct current from the solar energy system elements is combined to form a single current [43].

3.4 Infrared Technology

Thermal imaging techniques are used for solar cells and other electronic devices. In 1996, the integrated circuit could be mapped by a thermal technique using liquid crystal; this was reported by Csendes. A layer of nematic liquid crystal covered the device under test; the phase transition undergoes can be viewed as the sample is heated [66]. The same techniques were applied to solar cells; the temperature-sensitive polymer-dispersed cholesteric liquid
crystal foils covered the solar cell whose shunts can be localized by detecting the temperature variations in reverse-bias [66].

In 2004, a temperature-controlled stage with vacuum chunk was used for fast shunt analysis in the solar cells. The solar cell has a layer of liquid crystal foil, and images allow for the localization of where the temperature changes in reflectivity of the liquid crystal over layer [66]. In 1998, fluorescence microthermography (FMT) was the same technique which can be used to locate hot spots in circuits and devices. FMT is based on the efficiency of the temperature-dependent luminescence of rare earth chelate dyes [66]. In 2001, the non-destructive testing was done using infrared methods, for example, IR sensor and optics, thermal emission, and imaging and analysis. In 2000, a survey of the infrared imaging applications was done for shunts in solar cells, such as reverse-bias heating, by-pass diode functionality, resistive solder bonds, and other components of balance of system [66]. In 2002, the cracked solar cells were inspected by a thermographic imaging [66]. In 2011, the IR camera was used in the field to monitor the solar modules’ performance [66].

The proposed methods for using thermal images for fault detection in solar panels are discussed. Gao et al proposed a method for defect detection and solar panel recognition using infrared images via a camera mounted on a moving cart [67]. They used optical flow to establish frame-to-frame association to count the number of panels within any given array. They recognize the panels using Hough Transform (HT) and DBSCAN clustering technique to identify a hot panel by comparing it with its neighbours. Several drawbacks were reported such as when the top row is missed, the first column of an array cannot be recognized, or challenges with image registration in relation to optical flow results[67].

Tsai et al used Independent Component Analysis (ICA) basis images to detect a defective solar cell using Electroluminescence (EL) image [68]. Defects were presented as a
dark spot in EL images, such as breaks, finger interruptions, and micro-cracks. All of these defects have been detected using EL images. They proposed to use the cosine function as the distance measure and minimum reconstruction error. The drawback of their work is that the results are all based on a global approach and small localized defects can be missed [68].

Tsanakas et al proposed the use of standard thermal image processing and the Canny edge detection. This method was proposed for module-related faults that have hot spot heating. The main objective was the applicability of thermal image processing and edge detection to detect the defective PV modules. The proposed method did not classify the fault types. Another limitation is the unwanted grey-level variations that are caused by any specular object present in the background, which conflicts the with the actual variations related to hot spots, and thus the false alarm will take place. There are further limitations referring to emissivity uncertainties [69].

Thermal imaging is used in many applications, such as in quality control in the automobile industry for assessing the quality of all front and rear windows, or in an alert system against potential fire hazards, such as in a large warehouse. Thermal images are used for searching for weak points in the material, for example the heating wires in the rear window [70], as well as for checking the quality of a heated front window. Air conditioning in the automobile can be tested using the infrared cameras which acquire valuable temperature data [70]. In a warehouse, the thermal imaging camera can be used to detect casual fires. In early stages, the camera identifies fire pockets which trigged the fire alarm [70]. Thermal images are used to identify hot spots from within, completely non-destructively and non-invasively.

Thermal imaging was used to detect and track user interactions on arbitrary surfaces, for example, multi-finger signs, shape-based gestures and pressure based gestures [71]. The thermal imaging system can be used as a sensing solution for enhancing user surface interaction
The segmentation and detection of routine interaction in real time can be done by using a thermal imaging system with computer vision techniques [70].

Thermal imaging cameras and infrared thermometers have been used in many applications such as electronic circuit board assembly, plastic processing, airport maintenance, and many other applications. In the plastics industry, thermal imaging is used to measure the temperature. Using an infrared sensor system with non-contact temperature measurement is very important for control and process monitoring, which achieve high levels of quality because all the products undergo thermal processes [53]. The temperature measurement process of a single point can be achieved by using infrared thermometers which detect the critical point of the process. Thermal cameras help to visualize thermal events which optimize and monitor production processes [53].

Manufacturing of printed circuit boards and electronic components uses non-contact temperature measurement in order to check the quality. Infrared thermal imagers are used to do real time analysis of the thermal behavior. The video recording at 120Hz can be done using USB2.0 interface [53]. This recording allows us to analyze the thermal activities in slow motion in a test procedure, and individual images can be taken for more analysis. An alarm can be triggered to get indications about maximum and minimum temperature [53].

3.5 Concluding Remarks

In general, most previous works used a multicore system to processes images, while in this dissertation the thermal and photographic video processing is taking place. Most fault detection methods in the PV system are based on the use of sensors for current and voltage. There are researchers using only thermal images for fault detection in the solar panel, but many of their approaches did not determine the fault type or get an alert that specifies the type of the detected fault. Current approaches do not use the photographic camera simultaneously with the
thermal camera in order detect the external and the internal defects. The fault detection approaches did not benefit so much from the use of multicore or multiprocessor systems, especially the defect detection system which should be automated and work in real time operation of the PV system.
CHAPTER IV
PROPOSED FRAMEWORK – MULTIPROCESSING REAL TIME VISION BASED SYSTEM FOR CONDITION MONITORING IN SOLAR PANELS

The proposed system is based on using a multiprocessing module in Python, under a multicore CPU system, in order to divide the input thermal and photographic videos into different segments. Each segment should be processed in a specific process which uses fault detection algorithms in order to detect the faults in the solar panel. Each process is able to read all the frames in the specific segment and implement the fault detection algorithm simultaneously.

Figure 4.1 shows the system description. The proposed system consists of two cameras, thermal and photographic. The two cameras are able to capture the scene of the solar panels simultaneously. The cameras are connected on the drone which is able to fly over the solar panels. The recorded video on both cameras can be watched on the ground stations for thermal and photographic cameras. The recorded videos are processed on the multicore CPU in order to divide the input video into segments and start the running of fault detection algorithms in order to detect and locate the defective panels.
The proposed system is running on the multicore CPU using a multiprocessing module in Python. The main steps for the proposed system are shown in Figure 4.2.
Fly the drone over the solar panels and start recording on thermal and photographic cameras

Read the input video

Choose the number of processes for thermal and photographic videos processing

Call the Multiprocessing module to start the video segmentation process

Call *ffmpeg* to divide the input photographic video

Save the photographic video segments in a specific path

Call *ffmpeg* to divide the input thermal video

Save the thermal video segments in a specific path

Call the Multiprocessing module and send each thermal and regular segment to a specific process

Each process starts fault detection operation on a specific thermal and photographic segments simultaneously

Does the solar panel have defects?

No

Save the output results

Exit from Multiprocessing module

Yes

Analysis the frame to determine the location of the defective panel on the map, then send an alert.

Save the output results

Exit from Multiprocessing module

Figure 4.2: Proposed Algorithm Flow Chart
4.1 Multicore System for Fault Detection

In real time applications, the time constraints can affect the system operations. The reduction of the processing time for real time application is important. This reduction can be achieved by enabling an algorithm to be executed in parallel on multicore or multiprocessor systems. The processing flexibility and the processing real time character become possible by using the multicore processor [72]. A multicore system adds processing power with minimal latency which delivers significant performance benefits for software. This trend is shaping the future of software development toward parallel programming [73]. This benefit will be clear in applications which have huge input data and work in real time.

The thermal infrared imaging system has an imaging mechanism that produces a large noise in the images, and the edge has a great uncertainty. With the development of image acquisition technology, the image can get higher resolution, which means the image processing time will increase. The high-resolution images need a multicore system in order to process them in less time in the real-time [74].

In the proposed system, after recording the video using thermal camera, the input video is captured in Python using OpenCV, which helps to determine the number of frames and the length of the input video. Figure 4.3 shows the main steps for video segment process in order to process each segment in a different process.
Figure 4.3: Video Segmentation using ffmpeg

The duration of the input video is determined using ffmpeg in order to calculate the cutting interval using equation (4.1). ffmpeg should be installed with Python.

\[
\text{cutting period} = \frac{\text{input file duration}}{\text{number of processes}}
\]  

(4.1)
The cutting process is started in a While loop. It needs to determine the starting time which is initialized by zero, then increased by the cutting period as shown in equation (4.2). Then the input file duration is decreased by the cutting period as shown in equation (4.3).

\[ \text{from}_\text{time} = \text{from}_\text{time} + \text{cutting}_\text{period} \]  
(4.2)

\[ \text{input}_\text{file}_\text{duration} = \text{input}_\text{file}_\text{duration} - \text{cutting}_\text{period} \]  
(4.3)

The video portioning process is done using ffmpeg which uses the following command. This command is embedded in Python code.

```python
os.system('ffmpeg -ss ' + str(from_time) + ' -t ' + str(cutting_period) + ' -i ' + input_video + ' -c copy ' + output_dir + '/' + file_names + str(file_num) + '.mp4')
```

Once the video is divided into multiple segments, and the segments are stored in a specific path, the multiprocessing module is called in order to initialize the number of processes. Each process is initialized as shown in Figure 4.4. The running diagram for the multiprocessor system is shown Figure 4.5.

```python
if __name__=='__main__':
    tic3=time.time()
    #call video segmentation function
    video_ffmpeg()
    #call and start all processes to run simultaneously
    p1=Process(target=fun1)
p1.start()
p2=Process(target=fun2)
p2.start()
p3=Process(target=fun3)
p3.start()
p4=Process(target=fun4)
p4.start()
    #wait for this process to complete
    p1.join()
p2.join()
p3.join()
p4.join()
    #status code produced when the process exits
    p1.exitcode
    p2.exitcode
    p3.exitcode
    p4.exitcode
```

Figure 4.4: Code Structure for Multiprocessing Module in Python

67
Figure 4.5: Multiprocessing Module

All these processes will start running simultaneously while each process calls the specified video segment by capturing the input segment using OpenCV. Each process has a While loop to read each frame from the specified video segment. During the reading process, processes have also the capability to start the image processing operations for the fault detection algorithm. All the processes are running at the same time with the same operations. Any process that completes its specific task should exit from the execution with no waiting for another process to tackle.
4.2 Defect Detection Algorithms

In this dissertation, image processing techniques were implemented in order to detect the defects in the PV module. We implemented algorithms which are able to detect different defects in the PV modules. Each algorithm is implemented separately on different input data, and each algorithm is explained in the following sections.

4.2.1 Morphological Transformation and Canny Edge Algorithm

The first proposed framework is shown in Figure 4.6. Starting with converting the input image to grayscale in order to set the threshold values which help in edge detection for the binary images, we seek to identify regions of interest in the input thermal image; since defective solar cells show a strong yellow or white color with sharp edges in thermal images, it should be straightforward to recognize regions of interest.

In this algorithm, after converting the input image to a binary image, the thresholding process is applied to each frame. The threshold value, $Th$, for the photographic and thermal frames to be determined adaptively and at some experiments, was recomputed for each frame. Morphological transformations were used after assigning a kernel (structuring element) [75]. The first morphological transformation technique is Erosion. The main idea of erosion is a pixel will be considered 1 only if all the pixels under the kernel are 1; otherwise, it is eroded [75]. In other words, all the pixels near the boundary will be discarded depending upon the size of the kernel, which means white region decreases in the image.

The second morphological transformation technique is dilation; the idea of dilation is the opposite of erosion: a pixel is 1 if at least one pixel under the kernel is 1. In other words, dilation increases the white region in the image or the size of foreground object increases [75]. Then we used the Canny edge detection algorithm to detect the defective cells in the solar
The Canny edge detection algorithm is an optimal edge detection algorithm [76] that has been used in many practical problems with excellent performance results.

\[
E_{\text{Gradient}}(G) = \sqrt{G_x^2 + G_y^2}
\]  

(4.4)

Figure 4.6: Using of Morphological and Canny algorithm

We apply the Canny algorithm to identify significant intensity discontinuities in the image. The main idea is finding the direction of the gradient at each pixel. This can be done by finding the first derivative for the horizontal and the vertical directions using the Sobel filter. The equations (4.4) and (4.5) show the edge gradient and the angle calculations for each pixel respectively [77]. The Gradient direction is perpendicular to the edges; its value is rounded to one of four angles representing diagonal directions, horizontal or vertical [77].
After computing the image gradients, the unwanted pixels should be removed by scanning the image in order to identify which pixels do not constitute the edges [77]. Figure 4.7 shows the main concept of removing the unwanted pixels.

![Figure 4.7: Removing the Unwanted Pixels Using Canny Algorithm [77]](image)

The vertical direction is the edge, and the horizontal direction is the gradient direction. Point A is on the edge, points B and C are in the gradient direction. Point A is checked with neighboring points, points B and C, to determine if they can form as a local maximum. If they can, these two points can be considered for the Thresholding process. If point B and C are not in the local maximum, they will be set to zero (suppressed) [77].

The last step is the thresholding of the edges. This can be done by using two values for thresholding, minimum \((T_{h_{\text{min}}})\) and maximum \((T_{h_{\text{max}}})\) values. Comparing computed gradients with these two Thresholding values, edges are identified under the conditions in equation (4.6).

\[
Edges_{\text{Declaration}} = \begin{cases} 
\text{Intensity}_{\text{gradient}} > T_{h_{\text{max}}}, & \text{Sure\_Edge} \\
\text{Intensity}_{\text{gradient}} < T_{h_{\text{min}}}, & \text{Sure\_not\_Edge} 
\end{cases}
\] (4.6)
Edges are declared when they fall within the range as shown in Figure 4.8, since higher values than $Th_{\text{max}}$ are considered sure edge, such as A in Figure 4.8, while gradients below $Th_{\text{min}}$ are also discarded as not significant edges. The others located between the maximum and minimum values, such as B and C in Figure 4.8, will be compared with sure edges to see if they have any connection between them. They are considered as an edge if they have a connection with a sure edge, such as C. Otherwise, they are discarded such as B.

Figure 4.8: Thresholding Process in Canny Algorithm [77]

4.2.2 SLIC Super-Pixel Algorithm

The SLIC super-pixel technique is based on a spatial localization of K-mean clustering version. SLIC is an excellent tool for decomposing an image into small homogeneous regions, that is to locally group pixels and thus to provide a perceptual understanding of content [78]. Super-pixel reduces the complexity of the images from hundreds of thousands of pixels to only a few hundred [78]. SLIC can include a pre-processing phase including Gaussian smoothing filter to minimize outliers that would skew the results. Figure 4.9 shows the flowchart for SLIC super-Pixel algorithm.
SLIC is a proposed algorithm to efficiently generate superpixels by Achanta et al [79]. Figure 4.10 shows the main steps for the SLIC algorithm. The main parameter of the SLIC algorithm is the desired number of approximately equally-sized superpixels, $k$. The first step in SLIC is initializing cluster centers ($C_k$) by sampling pixels at regular grid step using equation (4.7), where $N$ is the number of pixels. In the next step, the cluster is moving to seed location,
to the lowest gradient position in a 3x3 neighborhood. For each pixel in the 2Sx2S region around \( C_k \) for each cluster center, the distance is computed between the pixel and the cluster center using equation (4.8).

Figure 4.10: SLIC Super-Pixel algorithm
\[ S = \sqrt{\frac{N}{k}} \]  

(4.7)

\[ D = \sqrt{\left(\frac{d_s}{s}\right)^2 m^2 + \left(d_c^2\right)} \]  

(4.8)

SLIC corresponds to clusters in \textit{labxy} color space, which means the spatial distance and color distance should be calculated using equation (4.9) and equation (4.10) respectively. The two distances are combined in equation (4.11) in order to normalize spatial proximity and color proximity by their respective maximum distances with a cluster, \( N_s \) and \( N_c \).

\[ d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \]  

(4.9)

\[ d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \]  

(4.10)

\[ D' = \sqrt{\left(\frac{d_s}{N_s}\right)^2 + \left(\frac{d_c}{N_c}\right)^2} \]  

(4.11)

The expected maximum spatial distance \( N_s \) within a given cluster should be equal to the sampling interval \( S \). The maximum color distance \( N_c \) is considered as a constant value \( m \) in equation (4.8) because the color distance can vary from image to image and cluster to cluster. Once the pixel is assigned to the nearest cluster, the new cluster centers will be computed and recalculate the distance until the residual error between the new cluster center and the previous cluster center is less than threshold value.

4.2.3 Segmentation Based on Hot Pixels Seeds Algorithm

The segmentation process divides an image into constitutive objects or parts [80]. Segmentation usually allows for object classification, pattern recognition, and/or the
identification of clusters [80]. Most segmentation algorithms are based on similarity or discontinuity features, such as edges and lines between different pixels [80].

The proposed segmentation based hot pixel detection algorithm determines a seed pixel $S_p$ selection in the input image. Figure 4.11 shows the main steps of the proposed algorithm. The pre-processing processes, Gaussian filter, and histogram equalization for the input images could solve the low contrast problem which makes thresholding more difficult.

After image pre-processing, the value of the hottest pixel is determined by finding the highest pixel value using equation (4.12). Then check all the neighboring pixels if they are related to the hot pixel, calling them seed pixels $S_p$, or to the background $B_p$ based on equation (4.13).

\[
\text{HotPixel} = \text{MAX} \left( \text{pixel[\text{row, column}]} \right)
\]  \hspace{1cm} (4.12)

\[
S_p = \begin{cases} 
\text{pixel[\text{row, column}]} \geq (\text{HotPixel} - \text{margin}) , & \text{Sure}_S_p \\
\text{pixel[\text{row, column}]} < (\text{HotPixel} - \text{margin}) , & \text{Sure}_B_p 
\end{cases}
\]  \hspace{1cm} (4.13)

Once the seed pixel $S_p$ is detected, the mean value $\mu_{S_p}$ is calculated using equation (4.14). $\mu_{S_p}$ is computed for each seed region with 8 neighboring pixels of the $S_p$.

\[
\mu_{S_p} = \frac{\Sigma_{\text{row}=0, \text{column}=0}^{9,9} \text{pixel[\text{\text{row, column}}]}}{9}
\]  \hspace{1cm} (4.14)
Figure 4.11: Segmentation Based on Hot Pixels Seeds
At the same time, an average value for all hot pixels $\mu_{\text{hot\_pixels}}$ is computed for each thermal frame, however, for the photographic frames, $\mu_{\text{hot\_pixels}} = 127$ was used which is a value that worked well in most cases. This actually brings parameter selection as one of the issues to be investigated further and hopefully develop an adaptive method for its selection. The actual seed pixel $\text{Act\_S}_p$ is determined using equation (4.15).

$$\text{Act\_S}_p = \begin{cases} 
\mu_{S_p} \geq \mu_{\text{hot\_pixels}} \quad \text{Sure}_{\text{Act\_S}_p} \\
\mu_{S_p} < \mu_{\text{hot\_pixels}}, \quad B_p 
\end{cases} \quad (4.15)$$

For each actual hot pixel, the Minimal Deviation Distance (MDD) is calculated using equation (4.16), where the standard deviation is computed using equation (4.17).

$$MDD = \min \left( (\Omega_{\text{Act\_S}_p})^2 \right) \quad (4.16)$$

$$\Omega_{\text{Act\_S}_p} = |S_p - \text{pixel [row, column]}| \quad (4.17)$$

For the background pixel, the mean value $\mu_{B_p}$ is computed for each pixel with its’ 8 neighbors, then delta value $\delta$ is computed using equation (4.18). MDD value is used to determine if the $B_p$ is defected based on equation (4.19).

$$(\delta = \left| \text{mean}_{B_p} - \mu_{S_p} \right| ) \quad (4.18)$$

$$\text{Defected}_{B_p} = \begin{cases} 
\delta \leq \text{MDD} , \quad \text{defected pixel } B_p = 1 \\
\delta > \text{MDD} , \quad \text{not\_defected pixel } B_p = 0 
\end{cases} \quad (4.19)$$

If MDD is greater than ($\delta$), then the background pixel is considered as a defected (hot) pixel (255); otherwise it is considered as a (zero) pixel.
CHAPTER V

EXPERIMENTAL WORK AND RESULTS

In this chapter, the results show that using the multiprocessing module in Python, on multicore CPU, for thermal and photographic videos processing shows execution time improvement and processor performance enhancements. Using the multicore system improves the thermal image processing techniques. The multicore system can be used to automate the fault detection in the solar panels using infrared images. In addition, using the multicore system for parallel processing should reduce power consumption and resource usage for the computer system.

5.1 Data Acquisition

To validate our proposed work, we used images taken for our solar system in the Digital Image and Signal Processing (DISPLAY) lab at Western Michigan University (WMU). This system is composed of two panels of SUNIVA OPTIMUS 60 Cell modules (Model OPT285-60-4-1B0); each panel produces 285W.

Table 5.1 shows the main specifications of the solar panels in DISPLAY lab. The whole system was built in our lab taking care of all the required components, connections, measurement, and mounting. Figure 5.1 shows the schematic diagram for the solar system while Figure 5.2 shows the complete installed system in DISPLAY.
In addition to our own data, we used images taken for solar panels at the main campus of WMU and others from online sources, with permission.

Table 5.1: WMU Solar Panel System Specifications in DISPALY lab

<table>
<thead>
<tr>
<th>SUNIVA OPTIMUS 60 Cell Modules (Model OPT285-60-4-1B0)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Electrical Specifications (Nominal)</strong></td>
</tr>
<tr>
<td>Max Power</td>
</tr>
<tr>
<td>Open Circuit Voltage (Voc)</td>
</tr>
<tr>
<td>Short Circuit Current (Isc)</td>
</tr>
<tr>
<td>Max Power Voltage (Vmp)</td>
</tr>
<tr>
<td>Max Power Current (Imp)</td>
</tr>
<tr>
<td>Max Series Fuse Current</td>
</tr>
<tr>
<td>Maximum System Voltage</td>
</tr>
<tr>
<td><strong>Mechanical Specifications</strong></td>
</tr>
<tr>
<td>Cells</td>
</tr>
<tr>
<td>Module Dimensions</td>
</tr>
<tr>
<td>Module Thickness</td>
</tr>
<tr>
<td>Weight</td>
</tr>
<tr>
<td><strong>Limits</strong></td>
</tr>
<tr>
<td>Operating Module Temperature</td>
</tr>
<tr>
<td>Storm Resistance/Static Load</td>
</tr>
</tbody>
</table>
Figure 5.1: Schematic Diagram for WMU Solar Panel System in DISPLAY Lab
The FLIR Vue Pro thermal camera was used for thermal video recording with (NTSC) frame rate, and with resolution of 336x256 pixels. This resolution is high enough to show an accurate thermal resolution from the solar panels. GoPro Hero 4 Black photographic camera will be used in the system; the camera has effective photo resolution 12.0 MP, and the max video resolution 3840x2160. These two cameras are connected on the Yuneec Q500 quadcopter as shown in Figure 5.3. The FLIR Vue Pro thermal camera is shown in figure 5.3 (a), the GoPro photographic camera is shown in Figure 5.3 (b), ST-10 is the personal ground station as shown in Figure 5.3 (c). ST-10 used to monitor the photographic camera photos and videos, and ST-
used to control the drone. Flysight Black Pearl RC801 FPV Monitor shown in Figure 5.3 is used to monitor the captured scene by the FLIR thermal camera.

![Drone System with Thermal and Photographic Cameras](image)

Figure 5.3: Drone System with Thermal and Photographic Cameras: (a) FLIR Thermal Camera, (b) GoPro Photographic Camera, (c) ST-10, and (d) FPV monitor.

We applied some external defects on our system in DISPLAY lab, such as Polystyrene behind the panel, adhesive paper on the front glass, a piece of gum, and a piece of Polystyrene in the back of the panel to mimic small defect, and we used an input image from an online source.

The input data were processed using the Python 2.7 on the Eclipse IDE platform. Python was installed on a Windows 10 environment, and other modules, extensions and libraries were installed using a pip command, for example, Multiprocessor module, Pillow, NumPy, and matplotlib. Python provides multiple modules for image processing which used OpenCV. We also used Windows 10 Home, Intel(R) Core(TM) i5-4210M CPU @2.60 GHz with 8 Giga Byte RAM to implement the offline system.
5.2 Multicore System for Defect Detection in PV using Video Processing

The results show that the impact of using a multicore system for thermal video processing to detect features for each frame in the thermal video is the reduction of the execution time. The use of a multicore system achieved a speed up processing and analysis operations of thermal video for the purpose of feature detection and classification. The multicore CPU was used to run image processing techniques for feature detection, such as Canny edge detection and Histogram equalization. The results are recorded for different scenarios; using one process, two processes, or four processes for feature detection in thermal videos. Table 5.2 shows the input thermal videos for feature detection purposes; these thermal videos have been processed using a multiprocessing module on Python. The Canny edge detection and Histogram equalization are processed for each frame.

<table>
<thead>
<tr>
<th>Input Video</th>
<th>Video Size (MB)</th>
<th>Number of Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video_1</td>
<td>12</td>
<td>826</td>
</tr>
<tr>
<td>Video_2</td>
<td>25.5</td>
<td>1798</td>
</tr>
<tr>
<td>Video_3</td>
<td>38.5</td>
<td>3103</td>
</tr>
</tbody>
</table>

Table 5.2: Input Of Thermal Videos for Feature Detection

Table 5.3 presents the processing time of Canny edge detector using one process, two processes, and four processes with the speed ups illustrated in Figure 5.4. Using a multicore system shows that the processing time was improved 1.93 times using 2 processes, and 2.6 times using 4 processes.

<table>
<thead>
<tr>
<th>Input Video</th>
<th>One Process P1 (Sec)</th>
<th>Two Processes P2 (Sec)</th>
<th>Four Processes P4 (Sec)</th>
<th>Speed up (P1/P2)</th>
<th>Speed up (P1/P4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video_1</td>
<td>59.37</td>
<td>28.61</td>
<td>21.55</td>
<td>2.08</td>
<td>2.75</td>
</tr>
<tr>
<td>Video_2</td>
<td>118.77</td>
<td>62.1</td>
<td>44.69</td>
<td>1.91</td>
<td>2.66</td>
</tr>
<tr>
<td>Video_3</td>
<td>190.17</td>
<td>105.81</td>
<td>83.22</td>
<td>1.80</td>
<td>2.29</td>
</tr>
</tbody>
</table>
In Table 5.4, we present the processing time of Histogram equalization using one process, two processes, and four processes with the speed ups illustrated in Figure 5.5. Using a multicore system shows that the processing time was improved 2.23 times using 2 processes, and 2.73 times using 4 processes.

Table 5.4: Processing Time for Histogram Equalization Execution Using Multicore

<table>
<thead>
<tr>
<th>Input Video</th>
<th>One Process P1(Sec)</th>
<th>Two Processes P2 (Sec)</th>
<th>Four Processes P4 (Sec)</th>
<th>Speed up (P1/P2)</th>
<th>Speed up (P1/P4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video_1</td>
<td>54.28</td>
<td>21.9</td>
<td>18.77</td>
<td>2.48</td>
<td>2.89</td>
</tr>
<tr>
<td>Video_2</td>
<td>109.62</td>
<td>50.57</td>
<td>42.16</td>
<td>2.17</td>
<td>2.60</td>
</tr>
<tr>
<td>Video_3</td>
<td>181.4</td>
<td>88.82</td>
<td>67.02</td>
<td>2.04</td>
<td>2.71</td>
</tr>
</tbody>
</table>
The previous results have been discussed in [81]. The output results for Canny edge detector and for the Histogram using one process or four processes are found to be similar, as shown in Figure 5.6.

Figure 5.5: Speedup Results for Histogram Equalization Using Multicore

Figure 5.6: Simulation Results for Feature Detection: (a) input frame, (b) and (c) Canny edge detector using one process and four processes respectively, (d) and (e) Histogram equalization using one process and four processes respectively.
In addition, the results show that the impact of using a multicore system for thermal video processing to detect defects in the solar panel is the reduction of the execution time. The results are recorded for different scenarios; using one process, two processes, or four processes for fault detection algorithm in PV systems using thermal images. Table 5.5 shows the input thermal videos; these videos have a different size which means a different number of frames. These thermal videos have been processed using a multiprocessing module by Python to show how the execution time was reduced, which improves the whole system’s performance.

Table 5.5: Input of Thermal Videos for Fault Detection.

<table>
<thead>
<tr>
<th>Input Video</th>
<th>Video Size (MB)</th>
<th>Number of Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video_1</td>
<td>3</td>
<td>464</td>
</tr>
<tr>
<td>Video_2</td>
<td>12</td>
<td>826</td>
</tr>
<tr>
<td>Video_3</td>
<td>25</td>
<td>2378</td>
</tr>
</tbody>
</table>

In Table 5.6, we present the processing time of using morphological transformation with canny edge detector to detect the defects in the solar panel. This execution was done by using one process, two processes, and four processes with the speed ups illustrated in Figure 5.7. Using a multiprocessing module shows that the processing time was improved 3.8 times using 2 processes, and 5.02 times using 4 processes.

Table 5.6: Processing Time using Morphological Transformation and Canny Edge Detection Execution for Thermal Video Using Multicore.

<table>
<thead>
<tr>
<th>Input Video</th>
<th>One Process P1 (Min)</th>
<th>Two Processes P2 (Min)</th>
<th>Four Processes P4 (Min)</th>
<th>Speed up (P1/P2)</th>
<th>Speed up (P1/P4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video_1</td>
<td>2.33</td>
<td>0.61</td>
<td>0.47</td>
<td>3.82</td>
<td>4.96</td>
</tr>
<tr>
<td>Video_2</td>
<td>4.39</td>
<td>1.12</td>
<td>0.95</td>
<td>3.92</td>
<td>4.62</td>
</tr>
<tr>
<td>Video_3</td>
<td>11.37</td>
<td>3.2</td>
<td>2.07</td>
<td>3.56</td>
<td>5.49</td>
</tr>
</tbody>
</table>
Figure 5.7: Speed up Results for Morphological with Canny Edge Detection Algorithm for Thermal Video Processing using Multicore

In Table 5.7, we present the processing time for using SLIC super-pixel for different size of segments, 150 and 500, to detect the defects in the solar panel. This execution was done by using one process, two processes, and four processes with the speed ups illustrated in Figure 5.8. Using a multiprocessing module shows that the processing time was improved 6.9 times using 2 processes, and 27.3 times using 4 processes. The speedup we achieved with the SLIC super-pixel algorithm is a very significant improvement. This is due to the nature of the algorithm which is based on clustering of hot regions. The algorithm has two different super-pixel segment sizes, 150 and 500 with maximum 10 iterations for k-mean causing execution time to be too long. The problem size, the number of processed frames, is large and once the execution time for each frame is long, the speedup using simultaneous processes resulted in a high superlinear speed up as shown in table 5.7.

When the problem size is divided into portions and executed among processes simultaneously, the execution time will have a significant reduction leading to this high superlinear speedup. In addition, the resource utilization will be more effective once the
problem is divided into portions; for example, the cache effect will take place once the problem is divided into more than one process via multicore CPU and run simultaneously.

Table 5.7: Processing Time using SLIC Super-Pixel Execution for Thermal Video Using Multicore.

<table>
<thead>
<tr>
<th>Input Video</th>
<th>One Process P1 (Min)</th>
<th>Two Processes P2 (Min)</th>
<th>Four Processes P4 (Min)</th>
<th>Speed up (P1/P2)</th>
<th>Speed up (P1/P4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video_1</td>
<td>234.28</td>
<td>38.45</td>
<td>8.28</td>
<td>6.09</td>
<td>28.29</td>
</tr>
<tr>
<td>Video_2</td>
<td>702.94</td>
<td>91.25</td>
<td>26.76</td>
<td>7.70</td>
<td>26.27</td>
</tr>
</tbody>
</table>

Figure 5.8: Speed up Results for SLIC Super-Pixel Algorithm for Thermal Video Processing Using Multicore

In Table 5.8, we present the processing time of using segmentation based on hot pixel seeds algorithm to detect the defects in the solar panel. This execution was done by using one process, two processes, and four processes with the speed ups illustrated in Figure 5.9. Using a multiprocessing module shows that the processing time was improved 3.84 times using 2 processes, and 6.24 times using 4 processes.
A multicore system has been used for simultaneous thermal and photographic videos processing to detect defects in the solar panel with a reduction of the execution time. Thermal frames and photographic frames are processed for the same panel at the same time. The results are recorded for different scenarios; using one process, two processes, or four processes for fault detection algorithm in PV systems using thermal images.

Table 5.9 shows the input thermal and photographic videos; these videos have a different size which means a different number of frames. These videos have been processed using a multiprocessing module by Python to show how the execution time was reduced, which improves the whole system’s performance. The processing time is recorded after the segmentation process is completed.
Table 5.9: Input of Thermal and Photographic Videos for Defects Detection.

<table>
<thead>
<tr>
<th>Input Video</th>
<th>Thermal Video</th>
<th>Photographic Video</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Video Size (MB)</td>
<td>Number of Frames</td>
</tr>
<tr>
<td>Video_1</td>
<td>4.76</td>
<td>456</td>
</tr>
<tr>
<td>Video_2</td>
<td>7.66</td>
<td>856</td>
</tr>
<tr>
<td>Video_3</td>
<td>11.4</td>
<td>1440</td>
</tr>
</tbody>
</table>

In Table 5.10, we present the processing time of using morphological transformation with canny edge detector to detect the defects in the solar panel using thermal and photographic videos. This execution was done by using one process, two processes, and four processes with the speed ups illustrated in Figure 5.10 Using a multiprocessing module shows that the processing time was improved 3.5 times using 2 processes, and 4.2 times using 4 processes.

Table 5.10: Processing Time for Morphological and Canny Edge Detection Execution for Thermal and Photographic Videos Using Multicore.

<table>
<thead>
<tr>
<th>Input Video</th>
<th>One Process P1 (Min)</th>
<th>Two Processes P2 (Min)</th>
<th>Four Processes P4 (Min)</th>
<th>Speed up (P1/P2)</th>
<th>Speed up (P1/P4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video_1</td>
<td>4.78</td>
<td>1.34</td>
<td>1.07</td>
<td>3.57</td>
<td>4.47</td>
</tr>
<tr>
<td>Video_2</td>
<td>8.01</td>
<td>2.18</td>
<td>1.93</td>
<td>3.67</td>
<td>4.15</td>
</tr>
<tr>
<td>Video_3</td>
<td>12.56</td>
<td>4.01</td>
<td>3.07</td>
<td>3.13</td>
<td>4.09</td>
</tr>
</tbody>
</table>
In Table 5.11, we present the processing time of using SLIC super-pixel for different size of segments, 50 and 200 with maximum 10 iterations for k-mean, to detect the defects in the solar panel using thermal and photographic videos. This execution was done by using one process, two processes, and four processes with the speed ups illustrated in Figure 5.11. Using a multiprocessing module shows that the processing time was improved 3.2 times using 2 processes, and 8.2 times using 4 processes.

Table 5.11: Processing Time for SLIC Super-Pixel Execution for Thermal and Photographic Videos Using Multicore

<table>
<thead>
<tr>
<th>Input Video</th>
<th>One Process P1 (Min)</th>
<th>Two Processes P2 (Min)</th>
<th>Four Processes P4 (Min)</th>
<th>Speed up (P1/P2)</th>
<th>Speed up (P1/P4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video_1</td>
<td>64.18</td>
<td>21.84</td>
<td>9.02</td>
<td>2.94</td>
<td>7.12</td>
</tr>
<tr>
<td>Video_2</td>
<td>144.95</td>
<td>46.3</td>
<td>17.96</td>
<td>3.13</td>
<td>8.07</td>
</tr>
<tr>
<td>Video_3</td>
<td>567.57</td>
<td>163.99</td>
<td>60.64</td>
<td>3.46</td>
<td>9.36</td>
</tr>
</tbody>
</table>
In Table 5.12, we present the processing time of using segmentation based on hot pixel seeds algorithm to detect the defects in the solar panel using thermal and photographic videos. This execution was done by using one process, two processes, and four processes with the speed ups illustrated in Figure 5.12. Using a multiprocessing module shows that the processing time was improved 2.7 times using 2 processes, and 6.4 times using 4 processes.

Table 5.12: Processing Time for Segmentation Based Hot Pixels Detection for Thermal and Photographic Videos Using Multicore

<table>
<thead>
<tr>
<th>Input Video</th>
<th>One Process P1 (Min)</th>
<th>Two Processes P2 (Min)</th>
<th>Four Processes P4 (Min)</th>
<th>Speed up (P1/P2)</th>
<th>Speed up (P1/P4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video_1</td>
<td>64.78</td>
<td>25.41</td>
<td>8.9</td>
<td>2.56</td>
<td>7.28</td>
</tr>
<tr>
<td>Video_2</td>
<td>107.58</td>
<td>36.1</td>
<td>18.6</td>
<td>2.98</td>
<td>5.78</td>
</tr>
<tr>
<td>Video_3</td>
<td>196.1</td>
<td>76.06</td>
<td>31.67</td>
<td>2.58</td>
<td>6.19</td>
</tr>
</tbody>
</table>
5.3 Defect Detection in PV Solar Panel

5.3.1 Morphological Transformation with Canny Edge Detector

The morphological transformations and Canny edge detector were implemented to detect the hot spots in the solar panel, as shown in Figure 5.13.
Figure 5.13: WMU Healthy and Defective Solar Panels: Morphological with Canny Detector Results: (a) Installed solar panel in DISPLAY lab at WMU, (b) Thresholding output, (c) Dilation output, (d) Erosion output, (e) Canny algorithm with morphological output, and (f) Canny without morphological transformation.

The input thermal image (a) in Figure 5.13 shows the solar 2 panels system in the DISPLAY lab at WMU. The panel on the left is healthy while the one on the right is defected. We have also imposed an external defect by putting Polystyrene behind the panel, on the sides causing heating to take place as hot spots. The thresholded image is shown in Figure 5.13 (b) while the morphological process result is shown in Figure 5.13 (c) and (d) for dilation and erosion respectively. Canny edge detector with morphological transformation is clearly included in the resulting edges defected region, as shown in Figure 5.13 (e). Canny algorithm without using thresholding and morphological transformation is shown in (f) showing a typical
edge and not isolating the hot spot. Hot spots indicate that the solar panel is working under abnormal conditions.

5.3.2 SLIC Super-Pixel Hot Spot Detector

The second section of the results shows the use of SLIC in order to determine the defected areas in the solar panels. Figure 5.14 shows how the defected areas in the solar panels are segmented in regions using the SLIC superpixel algorithm based on K-mean clustering for different size of segments. The hot regions are surrounded with big areas for the input image taken from the main campus at WMU in (a), 50 segments in (b), 150 segments in (c), and 500 segments in (d).

![Figure 5.14: Hot Spots Detection in Solar Panel Using SLIC Super-Pixel Algorithm: (a) Input image from solar panels at WMU, (b) 50 segments, (c) 150 segments, and (d) 500 segments.](image)

5.3.3 Segmentation Based Hot Pixels Detection

The last section in the results shows the use of the seed selection algorithm which is based on determining the seed pixel (hot pixel) and using the mean and standard deviation
values in order to determine the defected pixels and the background pixels. Figure 5.15 shows the results of the implemented algorithm where the hot spots have been detected. The input image in (a) is pre-processed using the Gaussian filter and histogram equalization in order to avoid the low contrast pixels as shown in (b) and (c). Then the algorithm which chooses the seed pixels is implemented and it determines if the background pixels are related to the seed (defected or not) as shown in (d). The dilation morphological transformation is applied after the algorithm to fill the holes in the output images as shown in (e). The algorithm is applied on the WMU solar system that we installed in our lab, and the algorithm was able to detect the hot spots in the solar panel as shown in Figure 5.15. After applying the erosion morphological transformation, the defected solar panel is shown clear on the right side of (e), and there is no significant white area on the left side to give any indications of defects.

Figure 5.15: Results for Segmentation Based Hot Pixels Detection Algorithm: (a) Installed solar panel in DISPLAY lab at WMU, (b) Gaussian filter output, (c) Histogram equalization output, (d) Algorithm output, and (e) Erosion morphological output.
External defects have been applied on our solar panel system in DISPLAY lab. The first defect was an Adhesive paper; it is used to write WMU on the panel glass. Figure 5.16 shows how the defects have been detected using the three algorithms and show the comparison between them.

![Image of output results for an external defect by adhesive paper](image)

Figure 5.16: Output Results for an External Defect by Adhesive Paper: (a) Installed solar panel in DISPLAY lab at WMU, (b) Morphological and Canny edge algorithm, (c) Segmentation based Hot pixels detection algorithm, and (d) SLIC Super-Pixel algorithm.

In Figure 5.17, we show the thermal images for the solar system in DISPLAY lab. A piece of gum was put on the solar panel glass to mimic a small external defect; this defect is
marked by (D1) in Figure 5.17 and detected as a hot spot. This defect has been detected as shown in Figure 5.17. In the same experiment, a small piece of Polystyrene behind the panel is put behind the solar panel; this defect is represented by (D2) in Figure 5.17. You can observe that the Polystyrene glued at the back of the panel has been detected.

Figure 5.17: Output Results for External and Internal Defects by Gum and Polystyrene: (a) Installed solar panel in DISPLAY lab at WMU (D1 is gum defect, D2 is Polystyrene defect), (b) Morphological and Canny edge algorithm, (c) Segmentation based hot pixels detection algorithm, and (d) SLIC Super-Pixel algorithm
An input thermal image was taken from online source [82], after getting the permission from the source, this image has some defective solar panels. The image was used to test the ability of these algorithms for fault detection. Figure 5.18 shows the results of applying the detection algorithms on the input image.

Figure 5.18: Output Results for Defective Solar Panels [82]: (a) Input image, (b) Morphological and Canny edge detector algorithm, (c) Segmentation based hot pixels detection algorithm, and (d) SLIC Super-Pixel algorithm

Figure 5.19 shows the output results for Morphological and Canny edge detector algorithm where the defects have been detected and the location of the defective panel is determined by longitude and latitude, and external (E) and internal (I) defects have been detected as shown in Figure 5.19 (d).
Figure 5.19: Location Information for Defective Solar Panel (external (E) and internal (I) defects): (a) Input frame, (b) Histogram equalization, (c) Canny detector output without morphological transformation, and (d) Morphological and Canny detector output with location information.

Figure 5.20 shows the output results for the segmentation based on hot pixels’ algorithm where the defects have been detected and the location of the defective panel is determined by longitude and latitude as shown in Figure 5.20 (e). The use of a photographic camera is important in order to detect some external defects, such as shadow; the shadow could hide some hot spots as shown in Figure 5.20 (e).
Figure 5.20: Location Information for Defective Solar Panel (external defects with shadow): (a) Input frame, (b) Gaussian filter output, (c) Thresholding output, (d) Segmentation based hot pixels detection algorithm output, and (e) Morphological output.

Photographic images can be used in order to determine some of external defects that cannot be shown as hot spots. Figure 5.21 shows the output results of SLIC super algorithm where the shadow can be detected.
Figure 5.21: Shadow detection using SLIC super-pixel algorithm: (a) Input image, (b) Shadow detection
CHAPTER VI
CONCLUSION

6.1 Summary

There are many desired characteristics of any real time system such as the system has high reliability, provides results in a real time, and has low power consumption. Solar systems work in real time but efficiency can be at risk if fault detection is not performed in real time but also with least cost.

The proposed system provides a safer inspection method by using the drone which can monitor the real-time operations of the solar panel from a distance with minimum human intervention. Also, the framework can be used for monitoring different real-time operations, such as monitoring cars on the roads, counting the number of pedestrians, construction operations, and finding missing persons in wild areas.

In this dissertation, I presented a new framework that is able to monitor day-to-day real-time operations of a solar energy system. I used a multicore CPU as a tool that enhances and improves the processing time for fault detection in a fully operating solar system. Internal and external faults were detected in real time which will reduce any associated hazards and increase the solar system’s efficiency and reliability. The framework has a multicore CPU able to process the thermal and photographic videos in real-time conditions, and using fault detection algorithms, was able to detect a variety of faults and to send an alert of the fault type along with latitude and longitude information of the defective solar panel.
Experimental results show that the use of multicore CPU improves the processing time for the captured thermal and photographic videos by segmenting the videos and running more than one process for the fault detection algorithm. The average improvement for the processing time of the detection algorithms for thermal and photographic videos was 3.1 times using 2 processes, and 6.3 times using 4 processes.

6.2 Contributions

This dissertation presents a framework that is able to use a multicore CPU and thermal and photographic cameras mounted on a drone, to monitor in real-time the conditions of an up and running solar system. The proposed framework is able to read and analyze the real-time operations of the solar system by recording videos for the real time operations of the solar panel. The captured videos are processed using a multicore CPU which improves the processing time. The achieved average speedup for the video processing in order to detect the faults was 3.1 times using 2 processes and 6.3 using 4 processes.

Also, using the multicore CPU allowed the proposed system to execute the fault detection algorithms in parallel which reduces the execution time and improves the performance. The framework is able to determine the location of the solar panel by determining the longitude and latitude by using the drone’s GPS system.

6.3 Future Works

This dissertation provides a multicore CPU real time condition monitoring system for solar panel fault detection. There are many ideas that to be further investigated based on the work presented in this dissertation. These may include:

1- Add a preprocess phase to allow for solar panel recognition and labeling prior to fault detection. This is a very important phase in order to count and identify the panels in solar gardens.
2- Build a multicore System on Chip (SoC) that is mounted on the drone which is able to process the algorithm using captured frames and send an alert directly to the control unit while the drone is flying.

3- Use the infrared images in order to quantify the electrical characteristics, such as current, voltage, power, and efficiency and embed them in a report.

4- Establish per solar farm a manual which contains signatures for different types of faults and assign the hazard level of the fault.

5- Fully automate some of the image parameter selection algorithms.
BIBLIOGRAPHY


