# A Crop Water Stress Monitoring System Utilising a Hybrid e-Infrastructure

Gaojie Sun*
School of Engineering
The University of Melbourne
Melbourne, Australia
Email:gaojies@student.unimelb.edu.au

Hongzhen Xie*
School of Engineering
The University of Melbourne
Melbourne, Australia
Email:hongzhenx@student.unimelb.edu.au

Richard Sinnott
Computing and Information Systems
The University of Melbourne
Melbourne, Australia
Email:rsinnott@unimelb.edu.au

***Abstract*—There are many challenges involved in irrigation systems in agricultural environments. Many approaches simply support crop watering as a simple standardised solution - watering the fields uniformly. However, depending on the topology of the ground and the crop species that exists, considerable improvements can be achieved that can greatly impact on the overall crop yield. Too little watering or too much watering in the wrong areas can impact the overall output. With the rapid growth in unmanned aerial vehicles (UAVs) and drones, image capture technology is now readily available. If the water-stressed areas could be identified, irrigation systems could offer targeted irrigation to only those crops in need. To support this, a crop water stress monitoring system has been designed utilising the thermal images of the crops. The analysis of such data can be computationally demanding when large crop fields are considered, hence the system has been designed to use multi-core high performance computing and Cloud-based systems. This paper outlines the requirements and design of the crop water stress monitoring system and how it provides an efficient and effective analysis of crop water stress index. The paper also benchmarks the algorithms across the hybrid infrastructure.**

***Keywords*—Irrigation, water stress, crop water stress index, parallelisation**

## I. Introduction

Intelligent irrigation of agricultural crops is critical. Agriculture is one of the largest consumers of water. Therefore, efficient and effective irrigation can not only increase crop production but also support water conservation [1]. In many areas of Australia, the driest inhabited continent, annual water demand of crops is often much higher than annual precipitation. Considering the water efficiency of the irrigation system, a more water-efficient approach is highly desirable. In most situations, different areas of crops have different water stress conditions with different levels of water requirements. Water efficiency can be improved by precisely irrigating water-stressed crops and reducing the irrigation for these areas with low water requirements without sacrificing the overall yield.

For estimation of crop water stress conditions, the crop's canopy temperature is a key indicator of water requirements [2]. To be more specific, when crops are water-stressed, in order to reduce the loss of the moisture, the stomas of the leaves close to prevent transpiration [3]. This interrupts the energy dissipation and leads to a rise in leaf temperature [4]. As such, when the water stress of a crop increases, its transpiration will decrease and the temperature will rise. Thus it is possible to create a crop water stress index (CWSI), which indicates the underlying water stress conditions, merely by monitoring the crop's canopy temperature. There are now many devices (UAVs and drones) that are equipped with thermal imaging capabilities, however, the associated data can be voluminous and the processing demands can be significant. Identifying slight temperature changes in hectares of crops of multi-terabytes of data can result in large-scale computational processing demands. The primary end users of advanced agricultural irrigation systems (farmers) are not typically savvy in large-scale data processing systems. The focus of this paper is to develop a scalable infrastructure that supports the variable data processing demands in identifying water-stressed crops. This has direct implications for many key agricultural areas of Australia (and elsewhere), e.g. wine-growing, fruit-growing as well as more traditional grain-growing farming enterprises.

One major challenge is the collection of the crop's canopy temperature. It is important to put forward a highly available and economical method for common crop growers. The traditional methods of measuring the canopy temperature need access to and use of large amounts of climatic data [5], and as such they are time-consuming and labour intensive. They often involve many field technicians traversing fields with handheld thermometers to collect the data or using visual observation [6]. For gardeners, this may be sufficient but for large scale crop growers, this will not be adequate.

Moreover, this method usually cannot ensure a high degree of accuracy due to the fact that the data may be obtained from a subset of the farming area, and the localised environmental conditions may change especially for farms with varying ground topologies. To overcome this problem, an automated system has been designed to collect the canopy temperature data. This system uses data captured from drones equipped with a thermal imaging camera to take images of the entire crop area within minutes [7]. Depending on the size of the fields, however, the data can run to many terabytes.

After acquiring the drone-based thermal images, the data needs to be used to create a crop water stress index [8]. Software (MatLab) has been written to support this, however this was targeted to run on a single laptop and would not scale to the kinds of data that farmers really need support for or with the kind of resolution that would identify specific areas of crops in water need. However since the image analysis program is computationally intensive especially for large fields

*These authors contributed equally to this work.

or when multiple users use the analysis system, scalability is essential. To tackle this, the MatLab algorithms were deployed on targeted HPC systems at the University of Melbourne and ported to Cloud-based platforms across Australia. End users are offered a simple web interface for interacting with these resources. This paper focuses on the large-scale infrastructure required for building the thermal image analysis system used to identify the water stress status of crops and to locate water-stressed crops. The paper also covers benchmarking of the system on hybrid infrastructures.

The rest of this paper is structured as follows. Section II covers the background and related work. Section III presents the image analysis algorithm design. Section IV presents an overview of the architecture of the system. Section V gives a case study of the system. Section VI focuses on a comparison of the performance of the hybrid e-Infrastructure. Finally, section VII draws conclusions on the work as a whole and identifies areas of future work.

## II. RELATED WORK

It is indicated in [9] that water stress can greatly influence plant height and the overall weight. Under water stress conditions, plants will reduce their size and quality, losing their market value through yield reduction. Thus, irrigation of crops is an important element in providing sufficient crop yields. This is especially needed when there is insufficient or excess rainfall.

Many different irrigation scheduling methods have been used worldwide [10]. These are based on weather data, the visible changes in plant characteristics, the appearance of the soil, or soil moisture monitoring which uses tensiometer devices to measure the moisture level of the soil. However, for large farms, considerable resources (money) is required to put so many devices in farms to enhance the accuracy of measuring water stress levels.

Another way to tackle this is by using thermal remote sensing which converts the invisible radiation patterns of leaves into visible images and subsequently using this information to determine particular thermal properties and features of crops [11]. These thermal images are usually caught by drones, a type of low-cost autonomous aerial vehicles that brings more business potential in near future. The main prototype is using this inexpensive vehicle to fly pre-programmed missions and taking thermal images of particular areas[12]. The analysis of thermal images has crossed over various areas. For example, a new method was proposed in [13] to estimate fruit yield during the growing season by analyzing temperature gradient between fruits and their background. Thermal images have also been proved to be useful tools to detect disease of plants with the help of analysis of the maximum temperature difference within a leaf or a canopy [14-16]. However, the scaling of the solutions has hitherto not been achieved for large-scale agricultural enterprises.

It is known that once plants are under water stress conditions, their stomata begin to close and cease to transpire, which can cause the plant to ”heat up” and the canopy temperature to rise [11]. Based on this theory, an algorithm was developed to assess crop water stress and irrigation scheduling information, using an optical image taken from the plant canopy which was aligned with the underlying infra-red (IR) image [17]. This was then used to extract the leaf area using Gaussian mixture distribution extraction techniques to get the distribution of the temperature of the leaves. Thus the crop water stress index can be obtained based on the leaf temperature distribution.

A major challenge is scaling these algorithms up to large-scale agricultural farms (where they are really needed). Several applications have already been implemented to process agricultural data on HPC and Cloud. A method introduced in [18] uses Cloud infrastructures to provide advice and recommendations regarding the analysis of data on the Cloud with the consideration of performance in particular. A web-based system described in [19] supports the processing of geospatial tasks on the Cloud by providing high-performance service-oriented computing capabilities. [20] explores rapid processing methods and strategies for remote sensing images, and makes comparisons with other computing paradigms.

We recognise that a range of infrastructures are now available and any solution must accommodate such heterogeneity of solutions. To address this, in this work, we develop a hybrid solution that leverages both HPC and Cloud-based facilities. Specifically, we utilise the high performance computing system SPARTAN at the University of Melbourne (https://dashboard.hpc.unimelb.edu.au). This provides a 4000-core server cluster with high-speed (10Gb) interconnection between all compute nodes. This system also offers a MatLab (https://au.mathworks.com) environment for researchers at the University of Melbourne.

In addition, we utilise the National eResearch Collaboration Tools and Resources Project Research Cloud (NeCTAR - https://nectar.org.au). NeCTAR is a federally funded project that offers an OpenStack-based Cloud infrastructure that is free to all academic researchers across Australia. NeCTAR offers 30,000 physical servers accessible through multiple availability zones across Australia. This is complemented by large-scale data storage solutions offered (again free!) to Australian academics as part of the Research Data Services (RDS - www.rds.edu.au) program. As part of the RDS program, the University of Melbourne currently makes available over 5 Petabytes of data storage to researchers. NeCTAR itself provides an Infrastructure-as-a-Service platform where instances of virtual machines can be created either through the NeCTAR dashboard (accessible through the Internet2 Shibboleth-based Australian Access Federation (AAF - www.aaf.edu.au)) or through associated tooling, e.g. use of libraries such as Boto (https://pypi.python.org/pypi/boto) for instance creation and management and Ansible (https://pypi.python.org/pypi/ansible) for deployment and configuration of software systems. Boto provides a client library for the OpenStack Nova compute component. These solutions allow for automatic deployment and configuration of virtual machines in a highly flexible and scalable manner. For this work, eight virtual machines (VMs) (each with 8GB RAM and 50 GB volumes) were utilised. Due to the licensing restrictions of MatLab, it was not possible to run the existing MatLab algorithms on NeCTAR, hence they had to be ported to another solution: Octave (https://www.gnu.org/software/octave/). Octave is a free software solution featuring a high-level programming language used for numerical computations.

Compared to other existing applications on Cloud or HPC, the hybrid processing system combines two processing platforms and provides an algorithm to schedule tasks to allow flexible use and optimal performance in use of these platforms. The solution offers a simple, but well-defined user manage-

ment component tailored to users. The platform enables users to run their jobs to obtain an optimal performance through a targeted algorithm that schedules tasks based on the status of processing platforms and the size of the job.

Another challenge to be tackled was how to parallelise the algorithms to speed up the processing time of large thermal image data. The parallelisation approach and the use of MPI are described in the following sections.

## III. Image Analysis Algorithm Design

### A. Problems in Analysis Program

The critical procedure for estimating the water stress condition is to calculate the crop water stress index (CWSI) from the thermal image. The CWSI can be quantified in a generic form [21], as below:

$$CWSI = \frac{T_c - T_w}{T_d - T_w}$$

In this formula, $T_d$ and $T_w$ represent the temperatures of dry (non-transpiring) and wet (fully-transpiring) leaves respectively. $T_c$ is related to the actual temperature of the leaves. It can be inferred from the pixel-by-pixel temperature data stored in a thermal image. $T_d$ and $T_w$ can be estimated from the leaf temperature distribution. Therefore, the key issue of calculating CWSI is to build a temperature distribution model of the crop to estimate the $T_d$ and $T_w$. Once the $T_d$ and $T_w$ are determined, the CWSI can be calculated.

The analysis program is based on the assumption that the temperature distribution of crops belonging to one species and a given soil type follows a mixture of two Gaussian distributions [2]. As shown in Figure 1, since the temperature of crops is lower than that of the soil, one of the two distributions (the smaller mean value) represents the temperature distribution of the crops, and the other (with relative higher mean value) is the soil temperature distribution.

If the whole temperature data stored in the thermal image is used to fit the Gaussian Mixture Model (GMM), then the cross section of the two distributions is unavoidable. This is manifested as a kind of noise for a given pixel since it combines crop or soil thermal readings. Ideally, it is desirable to obtain the temperature distribution model for crops separately from the soil distribution. Therefore, the noise part in the GMM should be reduced or eliminated. In fact, the noise comes from edge temperature between the crops and soil in the thermal image. As such, before building the GMM, the edge should be detected and eliminated from the thermal image. Then, as shown in Figure 2, the GMM built with the remaining temperature data can be clearly divided into the crop temperature distribution model and the soil temperature distribution model respectively.

### B. Image Processing Algorithm

The image analysis program (shown in Figure 3) is responsible for converting the thermal image to the crop water stress index map. The original laptop-based program was implemented in MatLab. It supported the following sequence of steps:

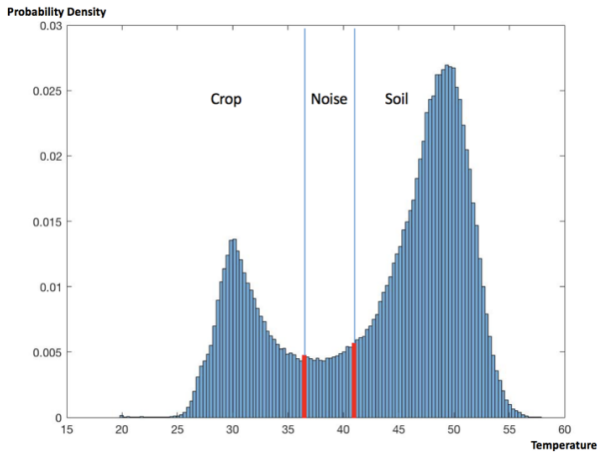



**Fig. 1: Gaussian Mixture Model With Noise**

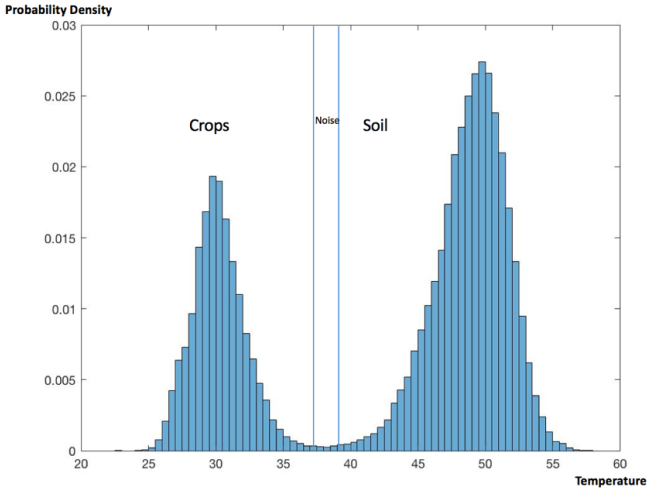



**Fig. 2: Gaussian Mixture Model With Reduction Noise**

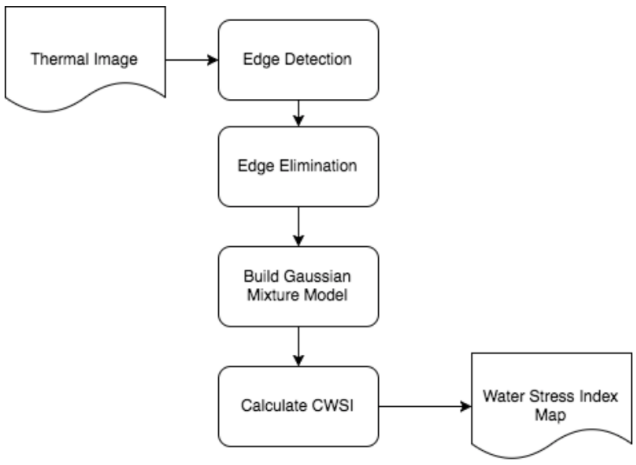



**Fig. 3: Algorithm Flow chart**

**Step 1. Edge detection:** The program first takes the thermal image, which contains the temperature data, as input. It then detects the edge of the image and outputs a binary matrix in which edge pixel values are set to 0 and the others are set to 1.

**Step 2. Edge Elimination:** Because the edge pixel value in the binary matrix is 0, it can be treated as a mask to eliminate the edge information from the original thermal image. In this step, the edge data will be removed by multiplying two matrices, namely the original temperature matrix of the thermal image and the binary matrix obtained from step 1.

**Step 3. Building the GMM:** After eliminating the noise data and the edge temperature data, the remaining temperature data in the thermal image can be used to build the Gaussian Mixture Models. If the image contains multiple crop species, multiple GMMs can be built for each species in the image in order to improve the accuracy of the model, since different species have different temperature distributions.

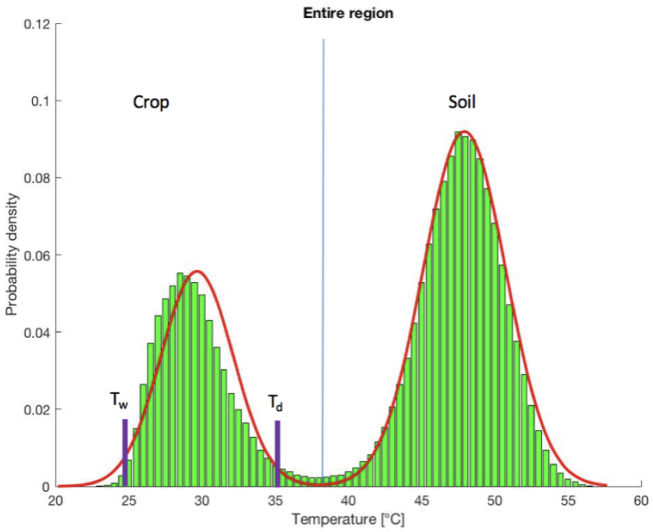



**Fig. 4: Utilisation of the GMM**

**Step 4. Calculating CWSI:** Once step 3 is completed (as shown in Figure 4), the crop's temperature distribution model can be obtained from the Gaussian Mixture Model by selecting the distribution with the lower mean value from the mixture model. The $T_d$ and $T_w$ can then be determined based on the assumption that 95% of the crops are in the normal condition, which means that the temperature for the non-transpiring leaf is the highest temperature value among the 95% of crops and the temperature for the fully-transpiring leaf is the lowest temperature value amongst them. From this and according to the formula described in section III, the CWSI for the whole image can be estimated. Finally, the crop water stress index can be established.

### *C. Image Processing Parallelisation*

When the analysis program processes large images, the analysis time will dramatically increase due to the substantial increase in the number of matrix operations. This is exacerbated when multiple users submit a great number of images to the system. Therefore, the analysis program utilises large scale heterogeneous computational resources including both HPC and Cloud infrastructures. The key idea is that each processor (core) is responsible for processing a subfield of a given thermal image.

**Step 1 and 2: (Figure 5(a))** For the Edge detection and Edge elimination steps, the input image will be cut into different sub-images, based on the current resource availability, i.e. the number of processors that are currently available. Each processor will take one sub-image and subsequently detect the edges and then eliminate the identified edges from those images.

**Step 3: (Figure 5(b))** Creation of the Gaussian Mixture Model cannot, in general, be parallelised because the construction of a GMM needs to use the whole temperature data to fit the model. However, there is one exception to this: when the image contains many crop species. In this case, multiple Gaussian Mixture Models need to be built for each of these species and these can subsequently be built in parallel on different processors. The utilisation of the processor in this step is low since all processors are not needed to build the GMM.

**Step 4: (Figure 5(c))** In this step, each processor will process one subfield of the input image. In particular, if the sub-image contains many crop species, the processor should use the corresponding Gaussian Mixture Model to calculate the CWSI of the sub-image.

## IV. System Architecture

A simplified view of the system is shown in Figure 6. The platform has been designed for users who have already collected the thermal images of the crops (by UAVs or drones). They upload their data (images) to the system to analyse the crop water stress index. The system allows multiple users to upload numerous data sets and submit multiple jobs at the same time and subsequently utilise the HPC (SPARTAN) and Cloud (NeCTAR) systems.

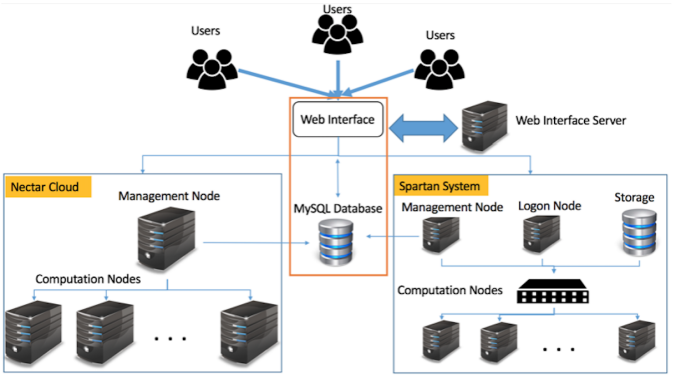
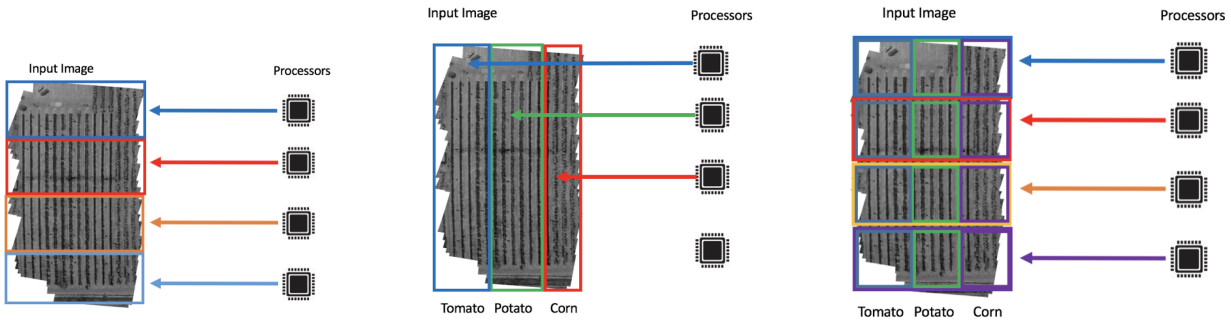



**Fig. 6: Analysis Program System Architecture**

The analysis program is accessible via a web interface. The web interface provides users with a user-friendly means to submit their thermal images and manage the submitted jobs.




**(a) Parallelisation of Edge Detection** **(b) Parallelisation in Building the GMM** **(c) Parallelisation in Calculating the CWSI**

**Fig. 5: CWSI Parallelisation Algorithm**

The web interface supports an authentication mechanism to restrict access to appropriate researchers (or farmers). When the user submits their jobs, they can check the status of the jobs running on the Cloud and/or SPARTAN systems. The decision where to run the jobs depends on the available computational resources. Users also can choose the computation system they wish to use to analyse their images. The system supports various job management capabilities. Firstly, it provides a service for checking the status of jobs. If the job is still processing, the overall status and each processors status can be updated in real time. If the job is finished, the result, the crop water stress index, can be viewed online. Second, the user can also delete the job through the web interface. If the job is still running, this operation will cancel the processing of this job and free the computational resources. If the job is complete, it will delete the result stored in the system. Finally, the web interface also provides a data download service.

A MySQL database has been adopted as the core data management software in the analysis system. It is installed on a server node along with the web interface. It is mainly responsible for storing the status of the job, the status of each processor, the user's information, links to the submitted thermal images and the crop water stress index maps. These data are generated from the web interface to the two management nodes of the NeCTAR Cloud and SPARTAN systems. They can also be queried and visualised through the web interface.

For the NeCTAR Cloud, there are a number of nodes that have been allocated specifically for this project. One node is treated as the management node and the others are treated as computational (data processing) nodes. The management node is used to receive jobs from the web interface and to subsequently schedule these jobs. In addition, it manages the computational resources and monitors and logs their status information, including the status of the job and the load on each computational resource. When the user submits their job, they can choose the number of the cores they need to use for the computation. Once the job is successfully submitted, the management node will prepare a set of computation nodes to analyse the image. These nodes that are responsible for one job, can exchange execution information through Message Passing Interface (MPI) capabilities developed to support the multicore parallelisation.

In terms of the SPARTAN system, this is similar to the NeCTAR Cloud. It contains a management node, a login node and a large number of computational nodes. It also contains a database to support the management of the SPARTAN system. The login node provides the authentication service that ensures that authorised users are able to use the SPARTAN system. The SPARTAN management node has the same functionality as the management node for the NeCTAR Research Cloud.

### *A. NeCTAR Research Cloud Computation Platform*

For the implementation on the NeCTAR Research Cloud, the key components not only contain the implementation of the parallelised image processing program but also a job scheduler that is responsible for scheduling jobs and allocating computational resources. The implementation itself leverages the open source Octave, Message Passing Interface (MPI) libraries and support for resource management (Simple Linux Utility for Resource Management (SLURM - https://slurm.schedmd.com/overview.html)).

It was necessary to provide job management software to manage jobs due to the limited computational resources available. When multiple jobs are submitted to the system, if the computation resources are inadequate for the total requirement of the submitted jobs, these jobs should be executed in a certain order, e.g. the submitted order. We utilise the SLURM scheduler, which is an open source and highly scalable cluster management and job scheduling system. For this purpose, SLURM has a centralised manager to monitor and schedule jobs. In addition, each computational node has a daemon used to receive jobs, execute jobs, return their status and wait for more jobs.

For the implementation of the parallelised analysis program, MPI is used to provide single program multiple data (SPMD) programming capabilities. It supports the use of multicores across different HPC nodes (servers) to parallelise the whole data processing activity. Each node has one copy of the analysis program that is managed by the SLURM daemon. When a job is submitted to the cloud system, the management node will reserve a set of computation nodes according to the requirements of computation resources of the job. Moreover, the daemons on the computation nodes will take the thermal

image of the job as the input and each core will be responsible for the analysis of the subfields in the input image.

### B. SPARTAN Platform

On SPARTAN, the implementation language used is MatLab. MatLab provides a Parallel Computing Toolbox used for supporting parallelisation of a job and solving computational and data-intensive problems using multicore processors. SPMD is one of the solutions provided by Toolbox to speed up processing time, using a pool of MatLab workers to process the body of each statement in parallel. Each worker in the pool contains a unique value of a variable *labindex*. Different workers communicate or transfer data by *labSend* and *labReceive*. Values returned are converted to composite objects on the MatLab client, which contains references to the values stored on the remote MatLab workers. The original image is evenly divided into several parts, and then each worker processes part of the data based on its *labindex*. The resource management solution of SPARTAN is also based on SLURM.

### C. Batch Processing

For NeCTAR, when a user submits a job, a set of virtual machines (VMs) will be created according to the requirement for the computation resources and necessary softwares (Octave and SLURM) will be installed on these VMs to process the job. In order to reduce the time for installing softwares on virtual machines, scripted solutions such as Ansible are used. The execution time in using the Cloud platform includes two parts: creation and configuration of the VMs and the actual data processing of the image data on the Cloud.

The SPARTAN HPC facility is a shared computing system at the University of Melbourne. When a user submits a job, it may take some time to wait for the computation resources to be available - so called queueing time. A range of queues have been set up on SPARTAN to deal with serial jobs, fast jobs or parallel jobs that typically have a longer execution time and demand more computational resources. In the system design the resources on SPARTAN are set at a maximum of 8 cores for any one job, whilst for NeCTAR it an upper limit of 32 cores for a given job can be specified. These were selected on the resources available on the HPC and Cloud systems. It is noted that NeCTAR offers a diverse range of VMs from m1.small: 1 core, 4GB RAM machines with 30GB secondary disk up to m1.xxlarge: 16 cores, 64GB RAM machines with 480GB secondary storage.

When a user submits a batch of jobs without specifying the platform, the web interface will determine where to schedule the jobs in order to reduce the overall processing time. In general, when the job is small, it is faster to process it on SPARTAN, since the queueing time on the *fast* queue is much less (typically around 10s) than the time for creating VMs on NeCTAR (typically 180s). Furthermore, the efficiency of MatLab is much higher than that of Octave. However, when the job is large, it is typically more desirable to process it entirely on NeCTAR, since it can utilise more cores across different nodes to speed up the overall processing time. The simplified job scheduling algorithm is shown below.

**Data:** Submitted Jobs
**Result:** Scheduling Jobs to reduce the processing time
Initialization: Put jobs into Queue Q;
**while** $Q$ *is not empty* **do**
  $j \leftarrow Dequeue(Q)$ ;
  **if** $size(j) \leq 500MB$ **then**
    schedule j to SPARTAN with [4-8] cores;
  **else**
    schedule j to NeCTAR with 32 cores;
  **end**
**end**
**Algorithm 1:** Simplified Job Scheduling Algorithm

## V. CASE STUDY

To understand how this system can be used to analyse water stress across a range of scenarios - in the sense that image is not simply processed in a fixed platform but users can choose the platform, the processing mode and have different data sizes - we present how the platform provides the status of available resources for users so that they can choose the processing modes based on the current system status.

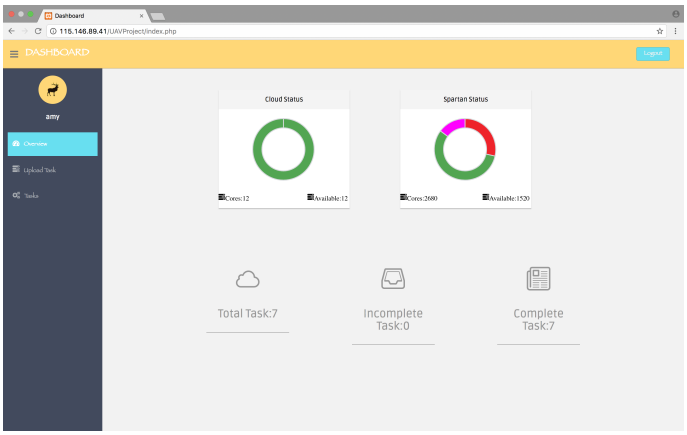



**Fig. 7: Overview of System Status**

To begin with, users need to log in to their account based on their assigned username and password. If they do not yet have an account, they can choose to register one. Once they have logged in to the system, they can check the status of the two processing platforms and the status of their own tasks. Figure 7 shows the typical status of the available resources in the two platforms (NeCTAR and SPARTAN) including the total number of cores and the available number of cores. The number of submitted jobs will also be listed, e.g. how many tasks this user has submitted, how many jobs are still processing and how many have successfully completed.

Based on the status of the processing platforms, users can decide which platform they wish to use to analyse their crop data as well as how many resources they wish to use. In addition to the thermal image data, a CSV file containing the boundary information of the area covered in the image should be uploaded as well as the thresholds for the minimum temperature and maximum temperature. Figure 8 illustrates how a user can process their images on NeCTAR. For

NeCTAR users, the number of nodes (VMs) and the number of cores (vCPUs) need to be provided, whilst for SPARTAN users, only the number of cores is required. It is common knowledge that with more cores being used, the processing time will speed up, however, in the case of NeCTAR delays may be caused by the time to create instances of the VM used for data processing, whilst for SPARTAN, delays may be caused by the queueing and overall load on the (shared) system.

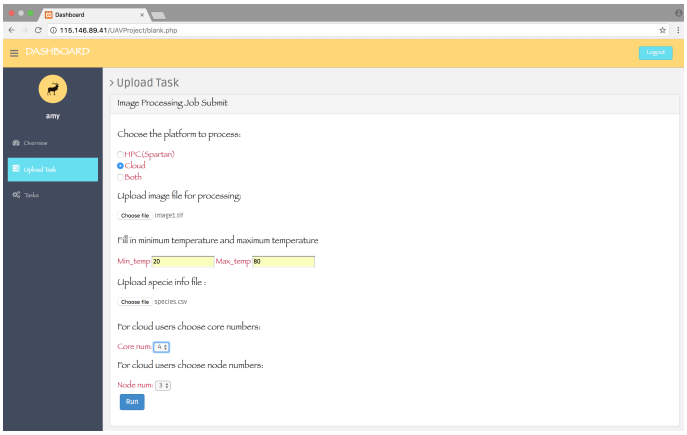



**Fig. 8: NeCTAR And SPARTAN User Interface**

Once a task has been successfully submitted, the status of resources and user's task summary will be updated. A task management part is also used to help users manage their uploaded tasks, e.g. view processor status and task status, delete tasks, view and download results. Figure 9 shows the management interface of a typical job. For each submitted job, there are three types of processing status for a given job: queuing, processing or completed. Users can choose to suspend or cancel a job. Once the job has been completed, users can see the result or choose to download the result to their own computer.

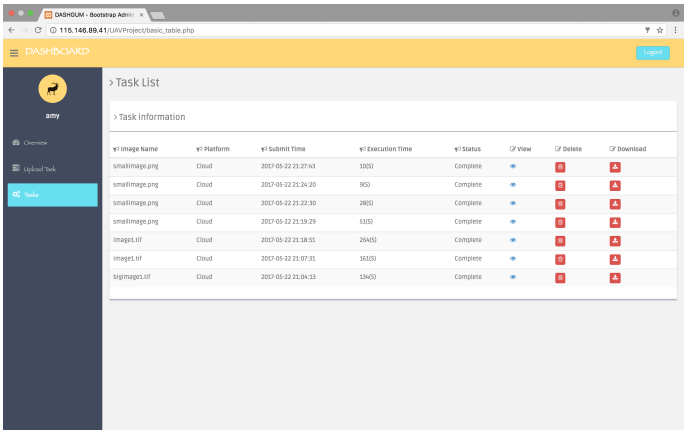



**Fig. 9: Job Management Part**

One key part of the system is that users can check the status of each processor, indicating the stage that each processor has reached. Figure 10 shows that, whilst there are 12 processors in total only 1 processor is currently building the GMM model of this image and the others are just waiting. The status of each processor is updated in real time.

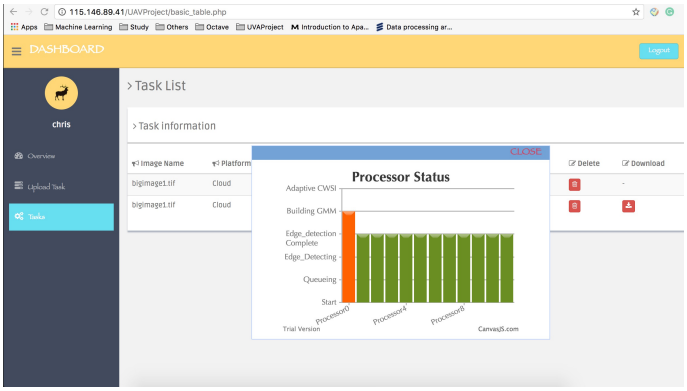



**Fig. 10: Processor Status**

Figure 11(a) illustrates one sample of a thermal image. Figure 11(b,c) shows the analysed result of this image. It is noticeable that some areas in this image are red which means that these areas are water-stressed, while those blue areas indicate that areas are in a normal condition.

## VI. Performance Comparison

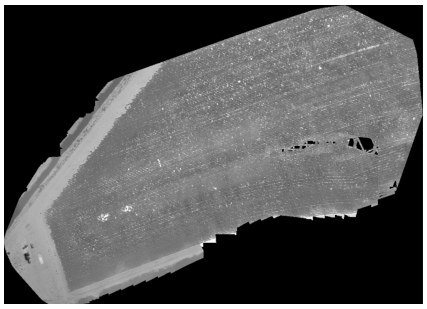
In this section, the execution time in use of the two different platforms is considered using three different sizes images. The sizes of data used in these three test samples are listed in Table 1. This includes the size of the image file and the associated matrix size. For each sample, we calculate the processing time on the two platforms with different numbers of cores and then make comparisons. In particular, the queueing time on SPARTAN and the time for creating VMs on NeCTAR are considered as well. We set the maximum core number for one job on SPARTAN to 8, while on NeCTAR, each VM has 4 cores and 8 instances were used in these experiments (hence jobs can use 32 cores). It is important that the system has been designed to scale however and further resources can be allocated/used.

| Sample | File Size(MB) | Matrix Size |
|---|---|---|
| sample1 | 34.4 | 4190*5597 |
| sample2 | 453.6 | 20255*11194 |
| sample2 | 1392.64 | 24306*27985 |

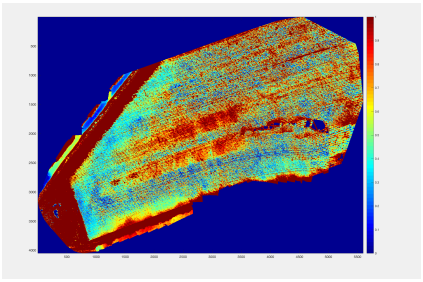
TABLE I: Size Information of Sample

The first sample used an input image of 34.4MB, with a matrix of 4190*5597. Figure 12 shows the performance of the first sample. As the number of cores increases, the processing time (naturally!) reduces. Without considering the queueing time on SPARTAN or the time for creating VMs on NeCTAR, before any parallelisation has been made, the job takes approximately 140s (SPARTAN) and 270s (NeCTAR) to get the analysis results using a single core. However, when using 32 cores on NeCTAR and 8 Cores on SPARTAN, it takes almost 50s on both platforms. It is noted that the analysis time of the job includes the time for creating VMs on NeCTAR and this is far more time-consuming.

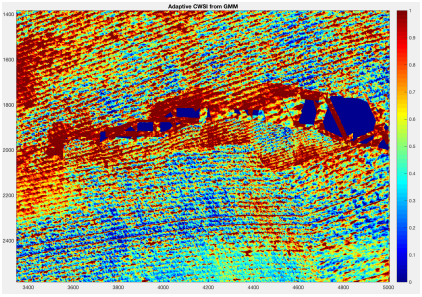
The second sample used was much larger with a thermal



(a) Standard Thermal Image



(b) Processed CWSI for Standard Thermal Image



(c) Higher Resolution CWSI for Subsection of Field

**Fig. 11: Image Analysis Examples**

image of 453.6MB and a matrix of 20255*11194. Figure 13 gives the execution time of sample 2. The performance of sample 2 shows a similar trend as sample 1. The more cores utilised, the less time required for the analysis. It takes approximately 3,320s (NeCTAR) and 2,150s (SPARTAN) respectively using one core to complete the data processing. In addition, when using 32 cores on NeCTAR, NeCTAR is slightly faster than SPARTAN without considering the time for creating VMs. In this case, however, the time for creating VMs is less time-consuming compared to the overall analysis time.

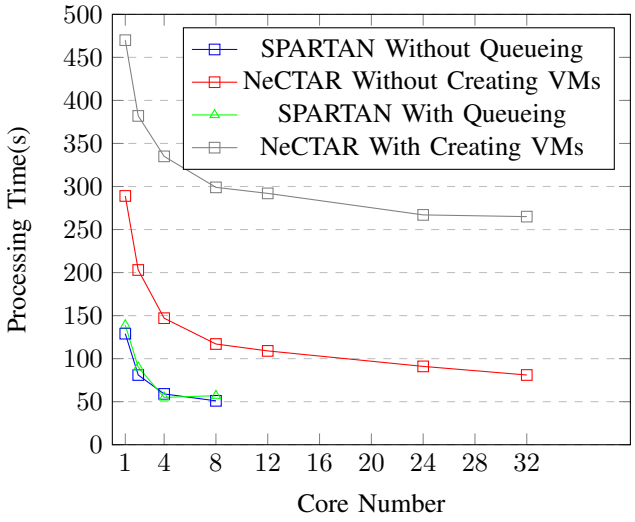



**Fig. 12: Performance of Sample 1**

The final sample used is a thermal image of 1392.64 MB with a matrix of 24306*27985. Figure 14 shows that for this sample, the fastest execution time on the two platforms is almost 813s (NeCTAR) and 1,429s (SPARTAN) respectively without considering the queueing time and the time for creating VMs. In this situation, when using the maximum number of cores, NeCTAR is much faster than SPARTAN. In this case, the time for creating VMs has very limited influence on the total analysis time.

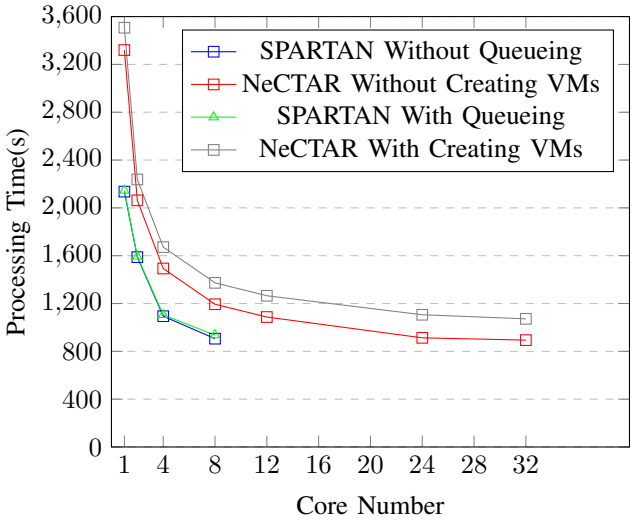



**Fig. 13: Performance of Sample 2**

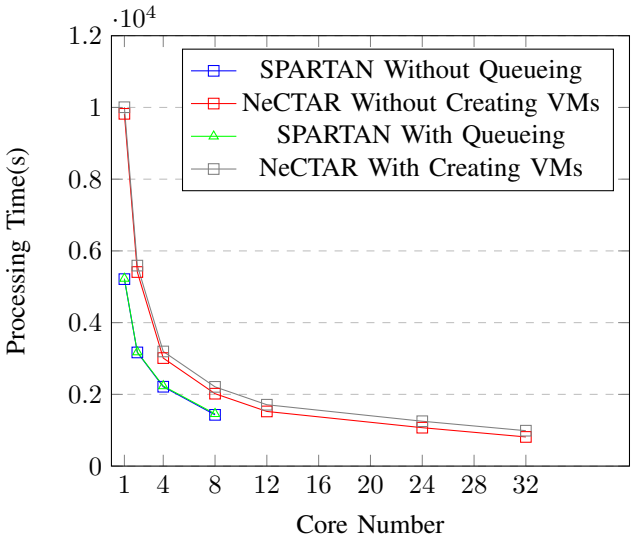



**Fig. 14: Performance of Sample 3**

From these results, it is easy to see that the total processing time on SPARTAN is faster than that on the NeCTAR Research Cloud when the image data sizes are small since the time for creating VMs on NeCTAR is dominant. However, when the data increases in size since NeCTAR can provide more cores than SPARTAN, the degree of parallelism of NeCTAR

is much higher than that of SPARTAN and the analysis time of NeCTAR eventually makes it worthwhile moving to the Cloud. In general, when using the same number of cores, the processing time on SPARTAN is always less than that of NeCTAR. This is also impacted by the language (Octave) used to implement the processing algorithm on the Cloud and the commercial software (MatLab) used on SPARTAN with its improved optimisation for matrix calculations.

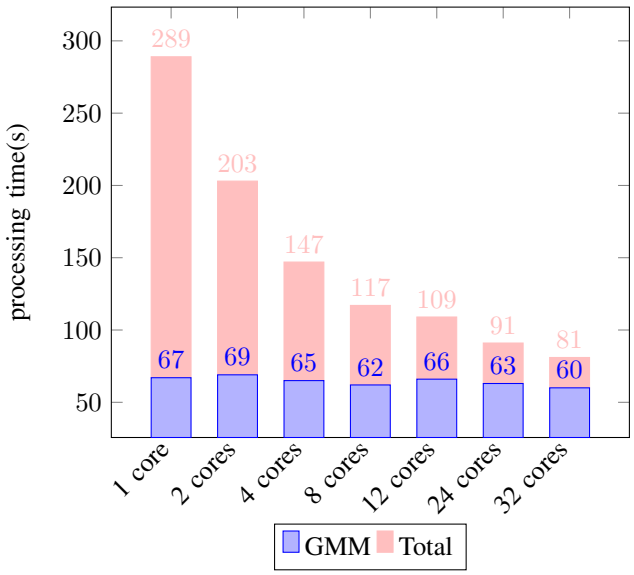



**Fig. 15: Processing time of different part**

It is worth noting that there are two main bottlenecks in the system that exists. One is the time for communication between different nodes or different cores. The other is in creating the Gaussian Mixture Model which cannot be parallelised. Figure 15 uses one sample as an example showing the time of this processing part in isolation. It is noticeable that even if the number of cores increases to 32, it still takes about 1 minute to build the model of the whole image. This is because there is no parallelisation in the GMM part. Thus, during the processing of this part of the algorithm, only one processor is needed while the others are waiting for next stage.

## VII. CONCLUSIONS AND FUTURE WORK

In this paper, we have introduced an infrastructure to support agricultural image water stress analysis. To scale the offering to support large agricultural farms dealing with highly disaggregated data sets, the system needs access to large scale infrastructure - both data and computation. As described, the thermal images themselves are processed in three steps. First, edge detection processes are run over the whole image. Then for each crop species in the image, a Gaussian Mixture Model is built based on specific areas of the image. Finally, a water stress index is calculated according to the mean value from the Gaussian model.

The whole processing algorithm has been parallelised to improve the performance in processing time across two heterogeneous platforms (NeCTAR and SPARTAN). We have benchmarked the algorithms on heterogeneous e-Infrastructure across these HPC and Could platforms. As more cores are used, the processing time is reduced. However, we identify bottlenecks that prevent significant enhancements offered by parallelisation. Firstly the time consumption between different nodes and cores and secondly in building the Gaussian Mixture Model. One possible solution to tackle this is by re-sampling which allows building the Gaussian Mixture Model of one particular species, i.e. only part of the image are required for building the model instead of the whole image.

This work also developed a web interface for users who want to analyse their thermal images to obtain water stress levels over larger areas. This interface helps users access remote processing platforms and monitor available resources as well as manage their own tasks (submitting, deleting and tracking the status of jobs), as well as checking the status of processors and jobs to make suitable resource selection choices.

There are a lot of improvements that can be made to enhance the performance of the whole system. First, the GMM part could be sped up by re-sampling. Second, more analysis over thermal imagery could be made, not only for water stress, but also for disease detection, greenhouse monitoring amongst other applications. The web interface could also be made more interactive, providing a more user-friendly interface. Factoring in data transport times is a further area of exploration of increasing importance in use of the Cloud for big data processing jobs.

A final work that is also being considered is in integrating the system to support real-time data collection. Thus as the farmers collect data with drones the data could be processed in real-time and targeted thermal imagery of those areas that have water stress issues can be identified in more detail.

## ACKNOWLEDGMENTS

The authors would like to thank the NeCTAR and Research Data Services (RDS - www.rds.edu.au) projects for the resources that have been made (freely!) available to support this work. Thanks are also given to the HPC administrators at the University of Melbourne for their continued support. Final thanks are also given to the scientific collaborators involved in this work - most notable Dr Dongryeol Ryu and Dr Suyoung Park.

## REFERENCES

[1] Mo, X, Liu, S, Lin, Z, Xu, Y, Xiang, Y and McVicar T.R. 2005, 'Prediction of crop yield, water consumption and water use efficiency with a SVAT-crop growth model using remotely sensed data on the North China Plain', *Ecological Modelling*, Vol. 183, issues 2-3, pp. 301-322.

[2] Wang, X, Yang, W, Wheaton, A, Cooley, N & Moran, B 2010, 'Automated canopy temperature estimation via infrared thermography: A first step towards automated plant water stress monitoring', *Computers and Electronics in Agriculture*, 73(1), pp. 74-83.

[3] Guilioni, L, Jones, H.G., Leinonen, I. and Lhomme, J.P. 2008, 'On the relationships between stomatal resistance and leaf temperatures in thermography', *Agricultural and Forest Meteorology*, 148(11), pp. 1908-1912.

[4] Gontia, N.K. and Tiwari, K.N. 2008, 'Development of crop water stress index of wheat crop for scheduling irrigation using infrared thermometry', *Agricultural water management*, 95(10), pp. 1144-1152.

[5] Meinke, H and Stone R.C. 1997, 'On tactical crop management using seasonal climate forecasts and simulation modelling: a case study for wheat.', *Scientia Agricola*, vol.54, pp. 121-129.

[6] Jackson, R.D. 1983, 'Assessing moisture stress in wheat with hand-held radiometers', *International Society for Optics and Photonics*, In 26th Annual Technical Symposium (pp. 138-142).

[7] Berni, J.A., Zarco-Tejada, P.J., Surez, L and Fereres, E 2009, 'Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle', *IEEE Transactions on Geoscience and Remote Sensing*, 47(3), pp. 722-738.

[8] Malveaux, C, Hall, S.G. and Price, R 2014, 'Using Drones in Agriculture: Unmanned Aerial Systems for Agricultural Remote Sensing Applications', *American Society of Agricultural and Biological Engineers*, In 2014 Montreal, Quebec Canada July 13July 16, 2014 (pp. 1).

[9] Baher, Z.F., Mirza, M, Ghorbanli, M and Bagher R.M. 2002, 'The influence of water stress on plant height, herbal and essential oil yield and composition in Satureja hortensis L', *Flavour and Fragrance Journal*, 17(4), pp. 275-277.

[10] The State of Victoria, 1996, Agriculture Victoria, 15th June 2017, <http://agriculture.vic.gov.au/agriculture/farm-management/soil-and-water/irrigation/about-irrigation>

[11] Pinter Jr, P.J., Hatfield, J.L., Schepers, J.S., Barnes, E.M., Moran, M.S., Daughtry, C.S. and Upchurch, D.R. 2003, 'Remote sensing for crop management', *Photogrammetric Engineering and Remote Sensing*, 69(6), pp. 647-664.

[12] Lian Pin Koh and Serge A. Wich 2012, 'Dawn of Drone Ecology: Low-Cost Autonomous Aerial Vehicles for Conservation', *Tropical Conservation Science*, 5(2), pp. 121-132.

[13] D. Stajnko, M. Lakota, M. Hoevar 2004,'Estimation of number and diameter of apple fruits in an orchard during the growing season by thermal imaging',*Computers and Electronics in Agriculture*, 42(1), pp.31-42

[14] Oerke, E.-C., Steiner, U., Dehne, H.-W. and Lindenthal, M. (2006) 'Thermal Imaging of Cucumber Leaves Affected by Downy Mildew and Environmental Conditions ', *Experimental Botany*, 57, 2121-2132.

[15] Lindenthal, M., Steiner, U., Dehne, H.-W. and Oerke, E.-C. (2005) 'Effect of Downy Mildew Development on Trans- piration of Cucumber Leaves Visualized by Digital Infrared Thermography ',*Phytopathology*, 95, 233-240.

[16] Oerke, E.-C., Lindenthal, M., Frhling, P. and Steiner, U. (2005) 'Digital Infrared Thermography for the Assessment of Leaf Pathogens',*The 5th European Conference on Precision Agriculture, Uppsala, 9-12 June 2005*, 91-98.

[17] Martin, E. (2009) 'Methods of Determining When to Irrigate '. *Cooperative Extension. College of Agriculture and Life Sciences, The University of Arizona, Tucson*. http://extension.arizona.edu/sites/extension.arizona.edu/files/pubs/az1220.pdf

[18] Hori, M., Kawashima, E. and Yamazaki, T., 2010.'Application of cloud computing to agriculture and prospects in other fields'.*Fujitsu Sci. Tech. J*, 46(4), pp.446-454.

[19] Chen, Z., Chen, N., Yang, C. and Di, L., 2012.'Cloud computing enabled web processing service for earth observation data processing'. *IEEE journal of selected topics in applied earth observations and remote sensing*, 5(6), pp.1637-1649.

[20] Wang, P., Wang, J., Chen, Y. and Ni, G., 2013.'Rapid processing of remote sensing images based on cloud computing'. *Future Generation Computer Systems*, 29(8), pp.1963-1968.

[21] Jones, H.G., Stoll, M, Santos, T, Sousa, C.D., Chaves, M.M. and Grant, O.M. 2002, 'Use of infrared thermography for monitoring stomatal closure in the field: application to grapevine', *Journal of Experimental Botany*, 53(378), pp. 2249-2260.