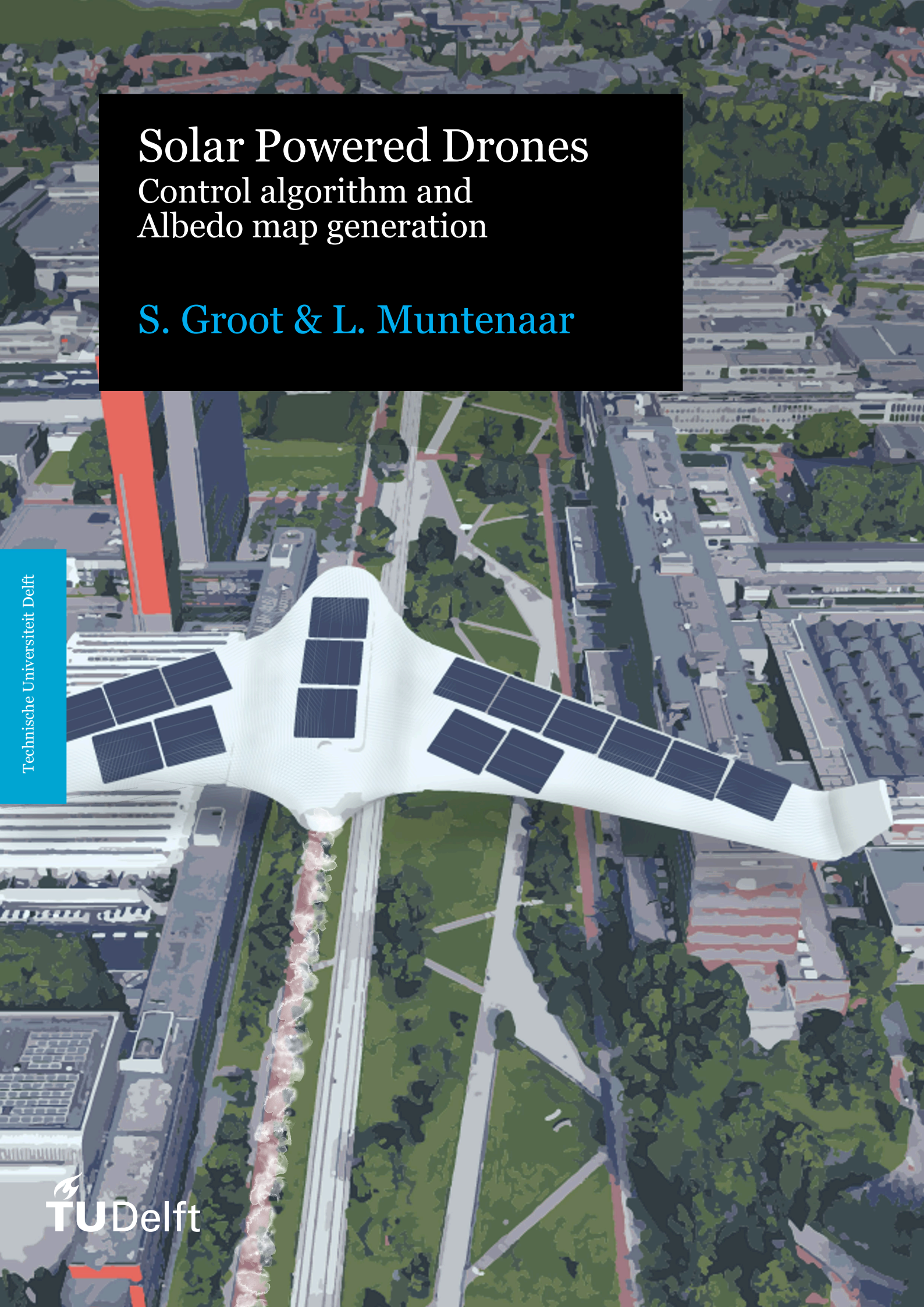


Solar Powered Drones

Control algorithm and Albedo map generation

S. Groot & L. Muntenaar

Technische Universiteit Delft

TU Delft

Solar Powered Drones

Control algorithm and Albedo map generation

by

S. Groot & L. Muntenaar

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended on Tuesday June 30, 2020 at 10:30 AM.

Student numbers: 4694368 (Groot), 4554213 (Muntenaar)
Project duration: April 20, 2020 – July 3, 2020
Thesis committee:

Prof. dr. Ir. J.J.A. Baselmans,	TU Delft, Jury Chair
Dr. P. Manganiello,	TU Delft, Supervisor
Dr. Ir. G.R. Chandra Mouli,	TU Delft, External Assessor
Dr. M. Muttillo,	TU Delft, Jury Member

TU Delft

Abstract & Preface

With the strict Paris climate goals, all eyes are set on the development of sustainable solutions that can replace or improve commercial solutions. This thesis revolves around the design and implementation of a subsystem for the creation of a solar powered drone. The goal of the solar powered drone is increasing the flight range of a commercial drone with the addition of photovoltaic (PV) panels. The subsystem designed and implemented in this thesis is split up into two separate systems namely, the control part and the image processing part. The control part revolves around the design, implementation and validation of Maximum Power Point Tracking (MPPT) for the PV panels. The image processing part revolves around the design, implementation and validation of an albedo processing pipeline. For the MPPT, a variable step size perturb and observe algorithm has been implemented with a worse case tracking delay of 40 milliseconds during a change of irradiation from $900W/m^2$ to $100W/m^2$ and and more than 99% power efficiency in steady state conditions. For the albedo pipeline, multiple RAW images are loaded one after another. Vignette correction is applied making the pixel values at the edges and the center of the image compatible with each other. The user can annotate a known reference target that is used to calibrate the albedo generation. Finally, the images are stitched together using Open Drone Map to create a complete albedo map. The results indicate accurate visible band albedo generation, however additional sensors would be needed to obtain wide band albedo generation.

This thesis is written in context of the Bachelor Graduation Project. We would like to express our gratitude towards our supervisor Patrizio Manganiello for always being available for questions and for the support he gave during the project. Next we want to express our gratitude towards Andres Calcabrini and Mirco Muttillo for delivering their feedback and support during the project. Finally, we want to thank our colleagues: Jasmijn Koning and Rik van der Hoorn of PV generation and Jetse Spijkstra and Martin Geertjes of Power electronics.

S. Groot & L. Muntenaar
Delft, June 2020

Contents

1 Introduction 1
1.1 Mppt Analysis . . . 1
1.2 Albedo Analysis . . . 2
1.3 Thesis synopsis . . . 2

2 Design criteria for the MPPT and albedo generation 3
2.1 Requirements of total system . . . 3
2.1.1 Assumptions . . . 3
2.1.2 Mandatory functional requirements . . . 3
2.1.3 Mandatory non-functional requirements . . . 3
2.1.4 Trade off requirements . . . 4
2.2 MPPT and Albedo generation specific requirements . . . 4
2.2.1 Mandatory Functional requirements . . . 4
2.2.2 Mandatory Non functional requirements . . . 4
2.2.3 Trade off requirements . . . 4

3 Analysis of Existing techniques 5
3.1 Introduction . . . 5
3.2 Albedo map creation introduction . . . 5
3.2.1 RAW vs processed images . . . 5
3.2.2 Vignette correction . . . 5
3.2.3 geometric distortion . . . 7
3.2.4 Orthomosaic images stitching . . . 7
3.2.5 RGB to albedo conversion . . . 8
3.3 Mppt Introduction and specifications . . . 8
3.3.1 Specifications . . . 8
3.3.2 Tracking of MPP . . . 9
3.3.3 Effect of weather on the maximum power point . . . 12
3.3.4 Local peaks problem . . . 12
3.3.5 Effect of solar cell configuration or power converter . . . 13
3.3.6 Implementation of MPPT . . . 13

4 Design 14
4.1 Introduction . . . 14
4.2 MPPT Design . . . 14
4.2.1 Algorithm selection . . . 14
4.2.2 Breakdown of algorithm . . . 15
4.2.3 Variable step analysis . . . 15
4.2.4 Final design . . . 15
4.3 Albedo Map Post Processing . . . 16
4.3.1 Vignette correction . . . 16
4.3.2 Post processing . . . 17

5 Implementation 18
5.1 Introduction . . . 18
5.2 MPPT algorithm . . . 18
5.2.1 Base model of P/O algorithm . . . 18
5.2.2 Test setup . . . 19
5.2.3 Integration of variable step . . . 19
5.2.4 Optimization of MPPT performance . . . 20
5.2.5 Physical implementation of MPPT . . . 20

5.3 Albedo post processing . . . 21
5.3.1 Loading and demosaicing Raw images . . . 21
5.3.2 Vignetting correction . . . 21
5.3.3 Reference target annotation and correction . . . 22
5.3.4 Orthomosaic map generation . . . 22
5.3.5 Physical implementation of the albedo map generation . . . 22

6 Validation and Discussion 23
6.1 Introduction . . . 23
6.2 MPPT . . . 23
6.2.1 Results of the final design maximum power point tracker . . . 23
6.2.2 Comparison of variable step and base algorithm . . . 24
6.2.3 Influence of different solar irradiances on duty cycle . . . 24
6.2.4 Influence of system voltage on duty cycle . . . 25
6.3 Albedo . . . 26
6.3.1 Vignette . . . 26
6.3.2 Albedo map generation . . . 28
6.3.3 Visible band albedo compared to wide band albedo . . . 29

7 Conclusion and Future work 30
7.1 Recommendations . . . 30

Bibliography 31

A Albedo generation step by step 33

B Validation figures 35
B.0.1 Final algorithm . . . 35
B.0.2 Comparison of variable step and base algorithm . . . 36
B.0.3 Final results . . . 37
B.0.4 Influence of different solar irradiances on duty cycle . . . 38
B.0.5 Influence of system voltage on duty cycle . . . 39

C Code 40
C.1 Gitlab MPPT model . . . 40
C.2 Albedo calculations to compare visible and wide band albedo . . . 41
C.3 Vignette modeling . . . 42
C.4 Albedo generation . . . 44

1

Introduction

This thesis is only concerned with of one of three subsystems of the solar powered drone. The final results and the total design/simulation of the drone can be found in the general drone documentation. The division of subgroups for the solar powered drone project is as follows, one group is concerned with the PV generator, one group is concerned with the power converter and one group is concerned with the maximum power point tracker and the processing of images to create an albedo map. This thesis is focused on the work of the last group, the Maximum Power Point Tracking (MPPT) and the albedo map generation. This chapter introduces the important concepts used in this thesis.

1.1. Mppt Analysis

The output power of the solar panel may vary over time due to changes in solar irradiation. Therefore, a control system that controls the power output is necessary. The optimal operation of the solar panel is ensured by implementing the MPPT.

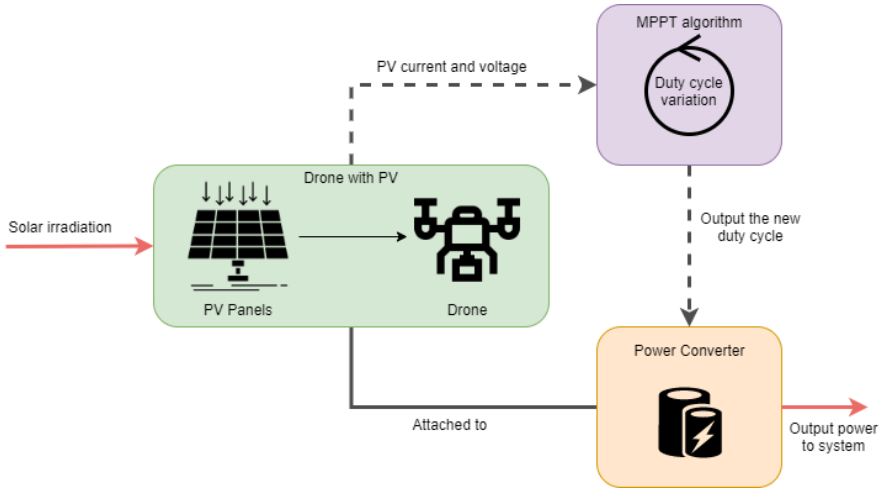


Figure 1.1: Overview where the MPPT controller is placed in the system

Figure 1.1 shows an overview of where the MPPT algorithm is situated in the drone system. The MPPT algorithm controls the power from the solar panel by controlling the duty cycle of the power converter based on the input characteristics of the photovoltaic (PV) panel. The power converter can then efficiently process this power and feed it into the system (motor, camera, controller, etc).

1.2. Albedo Analysis

Next to creating a solar powered drone or also called unmanned aerial vehicle (UAV), the project also has the secondary goal of generating albedo maps. The UAV is equipped with a camera that is facing towards the ground which will take pictures of the surface. These pictures need to be processed into albedo. The albedo of a surface can be seen as the fraction of solar irradiation that the surface reflects.

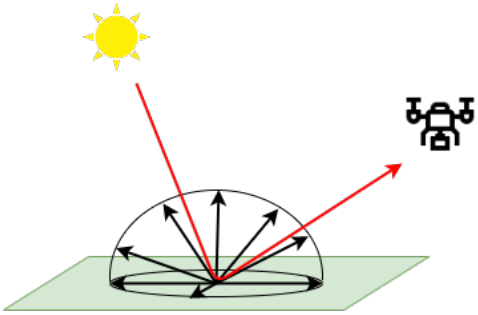


Figure 1.2: Lambertian surface Albedo example

Figure 1.2 shows an example of such a surface albedo. Here, the red arrow represents the sunlight which is reflected back from the surface. The black arrows represent the reflective pattern of the surface. In our case, we take the surface to be Lambertian which means that the ”brightness” of the surface is observed as the same in every direction. Using this and taking pictures with a known camera, the ratio between the red and black arrows can be approximated. Or in other words, the albedo can be approximated.

1.3. Thesis synopsis

This thesis is built up as follows, chapter 2 starts out with explaining the boundaries of the project and the requirements of the subsystems. Chapter 3 then starts with the description of existing techniques with regards to Albedo generation and maximum power point tracking. This is also named the State of the Art (SOTA). Chapter 4 describes the design of the mppt and the albedo from the requirements and the state of the art. Chapter 5 then elaborates on the Implementation of the designed subsystems. Next the sub systems need to be validated, this is explained in chapter 6. Finally, a conclusion and some further recommendations are given in chapter 7.

2

Design criteria for the MPPT and albedo generation

For the design of the MPPT and the albedo generation, the system needs to adhere to certain requirements. For this, a differentiation is made between system requirements and sub-group specific requirements. This chapter elaborates first on the system requirements and afterwards the MPPT and the albedo map generation requirements are specified.

2.1. Requirements of total system

The goal of this project is to implement solar panels on a UAV to increase its range, where the minimum flight range should cover the area of the TU Delft. This is in order to enable an onboard camera to capture an an albedo map of the area.

2.1.1. Assumptions

We assume the following conditions are met to operate our UAV.

- There is a large enough grass field nearby the location of operation to take off and land a fixed wing UAV.
- The operator flying the UAV is at least certified for ROC and the necessary permits for flying in the area are obtained.

2.1.2. Mandatory functional requirements

These are the requirements of the functions the system needs to adhere to.

1. The UAV should be able to capture an area the size of the TU Delft campus in a single flight in a maximum time of 60 minutes.
2. The UAV should be able to fly 5 times during one day, 2 times before noon, 1 time at noon and 2 times after noon, between flights the battery should be replaced or recharged.
3. The UAV's minimum flight altitude is 100m to improve area coverage for mapping and maximum flight altitude is 120m due to Dutch regulations.
4. The UAV should be able to fly with and without solar panels.
5. The components of the UAV should be able to be attached into the UAV.
6. The weight and size of the components should not prevent the UAV's ability to fly.
7. The camera should be able to create images suited for albedo mapping with a GSD (Ground Spatial Distance) of at least 20 cm at a flight height of 120 meters.
8. The UAV should be controllable using an autopilot for efficient albedo mapping.
9. The UAV should be able to take off and land in a controllable manner.

2.1.3. Mandatory non-functional requirements

These are the requirements which specify how the system should work.

1. The UAV should be less than 4kg to fall within the legal classification of a small UAV.
2. The UAV should be able to operate at temperatures ranging from -10 to + 40°C.
3. The UAV should be able to fly in dry and calm weather.
4. Components should be commercial available.
5. Prices of chosen components should be market conform.
6. The safety aspects of the UAV should be able to meet the dutch safety regulations for privately owned drones and UAV's
7. Maintainability:The average lifespan of the UAV should be at least 2 years when used for the predestined purpose.

2.1.4. Trade off requirements

1. The weight of the UAV should be as low as possible to increase flight time.
2. The flight path should be optimized to increase area efficiency.
3. The effective wing area should be as high as possible to increase power generated by the PV generator.
4. The drone should be fast enough to cover the required surface but it shouldn't effect image quality.

2.2. MPPT and Albedo generation specific requirements

For the Maximum Power Point Tracking (MPPT) and the image processing pipeline, the following specific criteria are necessary for the function of the system.

2.2.1. Mandatory Functional requirements

The following requirements are requirements that the system function should meet.

1. The MPPT algorithm should continuously track the mpp to keep the solar panels at their optimum voltage level.
2. The MPPT algorithm should track the global Maximum Power Point (MPP) even if multiple peaks are present.
3. The control algorithm should not affect the battery charge and discharge.
4. The control algorithm should be robust enough to withstand rapid changes in solar irradiation.
5. The efficiency of the MPPT should be at least 97%.
6. The image processing should convert raw color images into albedo images.
7. The Albedo image pipeline should include stitching of the images into one single image.
8. The Albedo image pipeline should be at least 5% accurate.

2.2.2. Mandatory Non functional requirements

The following requirements are requirements that describe how the system works.

1. The selection of the MPPT algorithm should not depend on the solar cell configuration.
2. The albedo map resolution should be sharper than 20cm per pixel.
3. Albedo map creation is done after the flight has taken place.

2.2.3. Trade off requirements

1. The tracking speed of the MPPT algorithm should be as fast as possible.
2. The albedo map should be as physically accurate as possible.
3. The albedo map creation could provide a visual gui to select and process the images.
4. The control of the MPPT should be as little dependent as possible on the power electronics architecture.

3

Analysis of Existing techniques

3.1. Introduction

This chapter explains the existing techniques and approaches that could be implemented for the corresponding subsections. As the concepts for albedo generation and mpp tracking have been around for some time, it is interesting to look into which different approaches could work for the UAV.

3.2. Albedo map creation introduction

The final goal of this project is to extend the range of the drone in order to capture an albedo map of the ground.

An albedo map represents what percentage of incoming solar radiation is reflected back by a surface. Bright white ice would have an albedo of about 0.84 and a dark forest has an albedo of 0.14. [24]

In [30] the generation of an albedo of the greenland ice sheet is described. The general workflow described in this paper was as follows:

1. Convert the RAW image to TIFF
2. Correct the vignette of the camera
3. Correct the geometric distortion
4. Take the mean of the RGB pixel values
5. Include a reference target with a known albedo within the image
6. Calibrate the mean pixel values using this reference target and an upward facing pyranometer to get an illumination corrected image
7. Calibrate this image using the measurments of a down and upward facing pyranometer carried by the drone

3.2.1. RAW vs processed images

RAW images contain all the information that the camera sensor captures without any processing. The pixel values in a raw image linearly encode how much incoming light is captured as suggested in [30]. Processed JPG images are altered by the camera software which will result in a nonlinear relationship between brightness and the pixel values.

For albedo map generation, brightness values need to be linearly dependent on the pixel values. However, [20] shows that it is possible to linearize a JPG image without significant loss in detail when the relationship is properly known.

3.2.2. Vignette correction

When taking a picture, the pixels at the center of the image are brighter than the pixels at the edges of the photo, this effect is called vignetting.

In [18] 4 different causes of vignetting are discussed.

- Mechanical vignetting, this is due to physical obstructions. These obstructions could be for instance a result of a lens hood.
- Optical vignetting refers to the blockage of off-axis light within the barrel of the camera.
- Natural vignetting is the fall-off of light because the image is projected on the flat surface of the sensor as described by the cos^4 law.
- Pixel vignetting describes the effect where the sensor pixels are less sensitive to light rays at an angle than light rays from head on.

According to [31], aperture and field of view are the main factors determining the vignette. In this paper the vignetting is determined by averaging 100 images to a reference image. Then the vignette can be compensated with this reference image divided by the center pixel value.

Instead of storing a complete reference image of the vignette, it is also possible to use a function to model the vignette based on the distance from the center and/or the horizontal and vertical position. In [12] a second degree polynomial function 3.1 was used and provided reasonable accuracy. Here the flat field correction is modeled based on the distance from the image center, r, and the pixel positions on the x and y axis.

$$FF = ar^2 + br + c + dx + ey \tag{3.1}$$

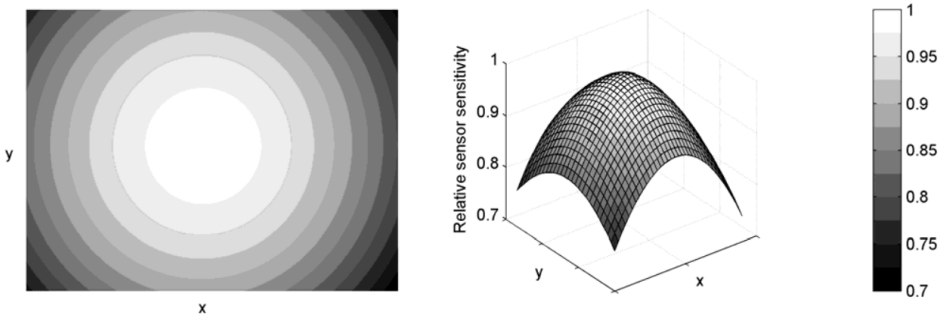


Figure 3.1: Left a quantified version of the effect of vignette and right an approximation with a polynomial function [12]

In [20] different polynomial orders (from 2 to 7) were tested to approximate the vignette. A second degree polynomial function was found to be able to model the vignette of a modern RGB digital camera.

In [18] a vignette correction method was proposed for cases where the vignette is not radially symmetric. It was concluded that the vignetting characteristics of a specific camera should be taken into account when choosing a vignetting function.

3.2.3. geometric distortion

Geometric distortion is the difference between the image that you capture and the actual image in the coordinate system that you want [26]. One factor that influences this is the radial distortion of the lens. In figure 3.2 the distortion of a lens at a focal length of 16mm is shown. Distortions that are a result of the geometry of the camera sensor itself are called internal distortions.

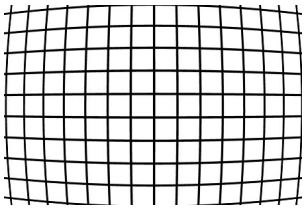


Figure 3.2: Distortion of a lens at 16mm as measured by DXO mark [7]

External distortions are due to external factors. This could be the height and alignment of the camera or the form of the object to be captured.

The orthomosaic image stitching already includes the correction of geometric distortions.

3.2.4. Orthomosaic images stitching

In order to combine the separate images into one view, they have to be stitched together.

Open Drone Map [25] is an open source image processing stack for creating 3D models and Orthomosaic maps from drone photography. The open source nature of this toolkit makes it ideal for research into how an orthomosaic is constructed. Figure 3.3A displays an overview of the scene used to test the process.

During loading of the dataset the images are indexed with the coordinate information from the image metadata. If possible, Ground Control Points (GCP) are read. These contain coordinate positions of locations within the images and are used to scale and place the final result into real world coordinates.

Once the data is loaded, the camera positions and orientations of the images are determined using OpenSFM. The blue rectangles in figure 3.3B depict the reconstructed camera positions above the point cloud of the test scene.

After the camera positions are determined, a dense point cloud of the scene can be reconstructed as displayed in figure 3.3B. In ODM, this is either done using Shading-Aware Multi-View Stereo (SMVS) [19] or using Multi View Environment (MVE) [10]. These algorithms match points on the different photo's and, with the known camera positions and view directions, triangulate the locations of these points in 3D space.

From this dense point cloud, a 3D model is created using a Poisson reconstruction algorithm [17]. First the normal directions of the points are determined and a vector field is constructed. An implicit surface function is approximated that tries to meet this vector field as good as possible. Then a 3D model is made using a cube marching algorithm. The images are projected onto this 3D model creating a fully textured environment as shown in Figure 3.3C.

The final step is to create the orthophoto. This is done by rendering the textured 3D model from above without any perspective. Figure 3.3D shows an example of a final orthomosaic image. As compared with the overview in figure 3.3A, this orthophoto is generated from multiple images and doesn't contain any perspective.

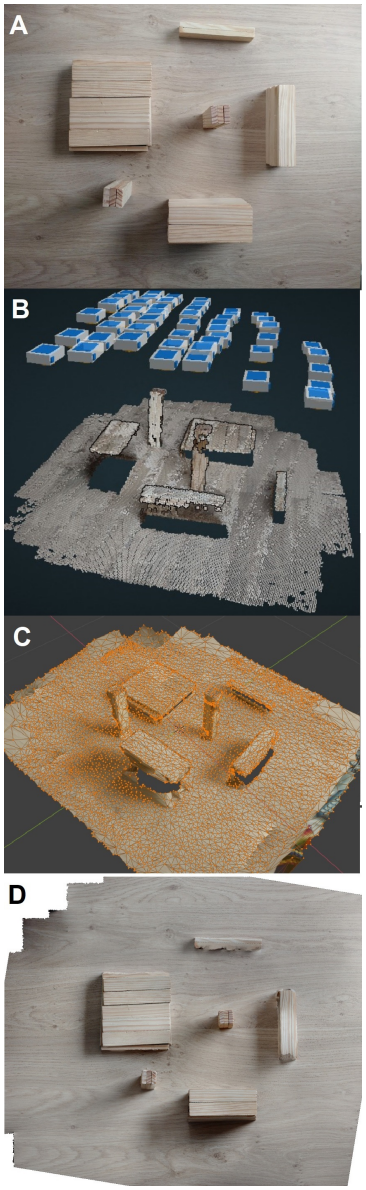


Figure 3.3: Different stages of creating a orthomosaic image.

3.2.5. RGB to albedo conversion

As mentioned earlier, the albedo of a surface is the percentage of light reflected from that surface. This could be expressed as equation 3.2 where δ is the surface albedo, $S(\lambda)$ the downwelling solar irradiance and $\rho(\lambda)$ the spherical reflectance of the material [32].

$$\delta = \frac{\int_{\lambda 1}^{\lambda 2} \rho(\lambda) S(\lambda) d\lambda}{\int_{\lambda 1}^{\lambda 2} S(\lambda) d\lambda} \tag{3.2}$$

In [30] a white Teflon reference target was included in every image. The image was then corrected for illumination using the ratio between the image pixels and the reference target. For cases where the reference target couldn't be included in the image the drone carried an upward facing pyranometer capturing the light conditions.

In their situation the surface they captured was ice and snow. It was noted that the reflectance of snow and ice was fairly invariant in the infrared region. It was therefore possible to only use the camera measurements of the visible wavelengths between 350 and 695 nm to determine the total surface albedo.

In [4] processed jpg images where used instead of raw images. A spectrometer was used to calibrate the images to reflectance values using a regression fit of equation 3.3.

$$y = a[ln(x+1)]^b \tag{3.3}$$

Here x represents the digital numbers of the image and y is the reflectance of ground targets.

After the visible band albedo (here taken to be wavelengths between 380 nm to 760 nm) was determined, it was converted to shortwave albedo values (wavelengths between 250 nm to 2500 nm) by multiplying by a conversion factor. This conversion factor was determined for both vegetation and non vegetation. This was done because of the significant difference between these two categories.

It was noted that adding a lightweight near infrared sensor to the drone could increase the accuracy of the results. The current spectrometer measurement for vegetation was based on measurements of grass, however a large portion of the vegetation consists of trees. Furthermore adding more subcategories could increase the accuracy of converting from visible to shortwave albedo.

3.3. Mppt Introduction and specifications

For the maximum power point tracking, the PV specifications need to be explained first. Figure 3.4 shows the average PV characteristics of a solar array. The marked point shows the global maximum, which is the point at which the solar array works most efficiently. The goal of the MPPT is to move the power operating point of the solar panels to this Maximum Power Point (MPP), thus ensuring that the PV panels are working as efficiently as possible. An efficient tracker algorithm thus aims to find the global maximum of the p-v characteristic as fast and reliably as possible.

3.3.1. Specifications

Existing approaches of power point tracking are introduced in this section as well as implementations. The main specifications that define a good power point tracker for our case are the following, which can be related to the requirements specified in Chapter 2.

- Fast tracking
- Robust
- Low complexity of simulation, Low complexity
- Not dependent on solar array configuration or power converter
- Dynamically adaptable

This limits the amount of existing techniques that can be used. For our algorithm, the following 5 existing techniques are investigated; Hill climbing, Ripple correlation control, Fuzzy logic, incremental conductance and current sweep.

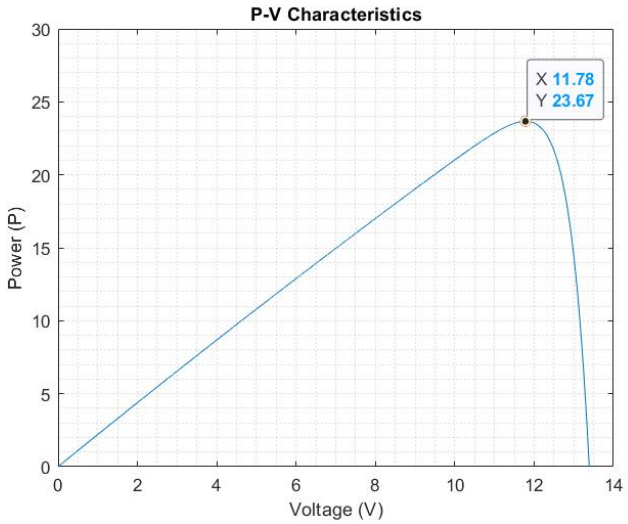


Figure 3.4: PV characteristics

3.3.2. Tracking of MPP

The tracking of the MPP to obtain the point indicated in figure 3.4 can be done using different methods. Paper [8] [33] introduce several of these different methods.

Hill Climbing

The first and most intuitive way of tracking the MPP is the Hill-Climbing or Perturb and observe (P/O) method. The hill-climbing or P/O algorithm involves the perturbation of the PV operating voltage to indicate a positive or negative change in the power [8][33]. Looking at Figure 3.4, it becomes visible that a perturbation is voltage results in a perturbation in power.

Table 3.1: Tracking of MPP with P/O algorithm

perturbation in voltage	Place on curve	Power Perturbation	Powerpoint w.r.t. the MPPT
Positive	Left of MPP	Positive	Powerpoint moves towards MPP
Positive	Right of MPP	Negative	Powerpoint moves away from MPP
Negative	Left of MPP	Negative	Powerpoint moves away from MPP
Negative	Right of MPP	Positive	Powerpoint moves towards MPPMPP

In table 3.1 the resulting ”output” power perturbation due to voltage perturbation is indicated. With this, the maximum power point can be tracked. If a positive voltage perturbation results in the operation point moving towards the maximum power point, the system will know that the current point is on the left of the MPP so the next perturbation will also need to be a positive change in voltage. This process continues until the MPP is reached and then the algorithm will oscillate around the MPP. This oscillating around the MPP results in the algorithm never truly reaching the MPP, causing the algorithm to not be as accurate.

Incremental conductance

Another way of tracking the MPP is using a method which is called Incremental conductance(ICC). Paper [8] describes the incremental conductance algorithm. The ICC algorithm is based on the fact that the slope of the PV power curve is zero at the MPP, positive on the left of the MPP and negative on the right side of the MPP.

$$\begin{cases} dP/dV = 0 & \text{at MPP} \\ dP/dV > 0 & \text{left of MPP} \\ dP/dV < 0 & \text{right of MPP} \end{cases} \tag{3.4}$$

This can be re-written since it is known that $dP/dV = d(IV)/dV \simeq I + V\frac{\Delta I}{\Delta V}$

$$\begin{cases} \Delta I/\Delta V = -I/V & \text{at MPP} \\ \Delta I/\Delta V > -I/V & \text{left of MPP} \\ \Delta I/\Delta V < -I/V & \text{right of MPP} \end{cases} \tag{3.5}$$

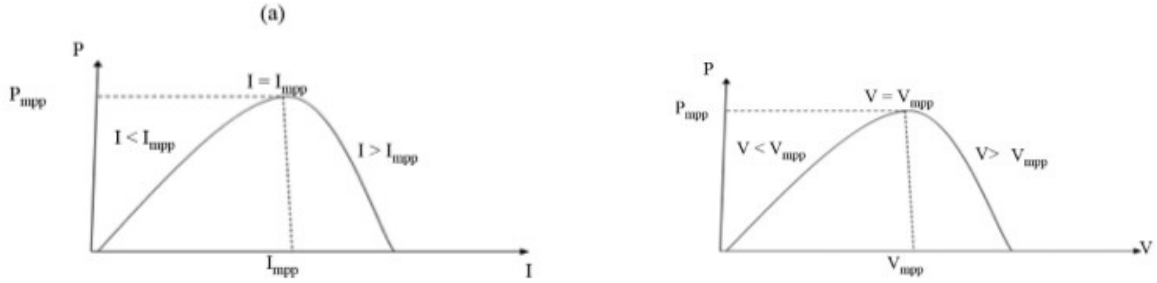
The MPP can then be tracked by comparing the instantaneous conductance (I/V) to the incremental conductance (ΔI/ΔV) and adapting the duty cycle accordingly.

One of the advantages of ICC is that with this algorithm, the actual MPP can be tracked. Next to this, ICC performs well under varying weather conditions. The algorithm is not dependent on the PV array or the power electronics architecture.

The algorithm does, however, also have the problem of step size. It will never completely reach the MPP but rather oscillate around that point. Also, according to [33], the implementation of ICC is complex and there is no parameter tuning for this algorithm.

Ripple correlation control

The previous methods were mostly digital ways of implementing an MPP control. An analogue method of tracking the MPP is Ripple Correlation Control (RCC) [9]. When the PV array is connected to a power converter, the switching action of the power converter induces a voltage and current ripple on the PV array. Thus inducing a power ripple on the PV array power. RCC uses this ripple to perform MPPT.[8] This is done by correlating the time derivative of the time-varying PV power dp/dt to the time derivative of the time-varying PV current or voltage (di/dt or dv/dt respectively)[33] .



(a) Power vs current characteristics [33]

(b) Power vs voltage characteristics [33]

Figure 3.5: Power depending on current or voltage

Figures 3.5a, 3.5bshows the general behaviour of the pv array. Looking from a RCC perspective, the following two cases can occur:

$$\frac{dv}{dt} > 0 \text{ or } \frac{di}{dt} > 0 \text{ and } \frac{dp}{dt} > 0 \text{ means that } V < V_{mpp} \text{ or } I < I_{mpp}$$

$$\frac{dv}{dt} > 0 \text{ or } \frac{di}{dt} > 0 \text{ and } \frac{dp}{dt} < 0 \text{ means that } V > V_{mpp} \text{ or } I > I_{mpp}$$

With RCC, the power ripple needs to be forced down to zero. Looking closely at these two equations, it becomes visible that the product $\frac{dv}{dt}\frac{dp}{dt}$ or $\frac{di}{dt}\frac{dp}{dt}$ is positive when the current power point is on the left of the MPP and negative when on the right of the MPP. This fact can thus be used to change the duty cycle of the power converter accordingly.

$$d = -k \int \frac{dv}{dt}\frac{dp}{dt} dt$$

Fuzzy logic control

Another method which potentially could be used is Fuzzy logic control. Fuzzy logic control consists of 3 stages: fuzzification, rule based table lookup and defuzzification [8]. In the first stage, numerical values are put into different member categories. Figure 3.6 shows the different categories:

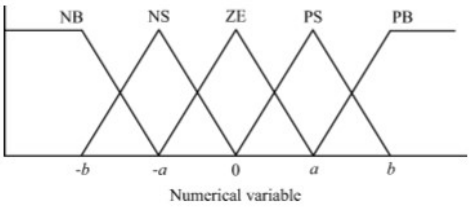


Figure 3.6: Fuzzy logic decision boundaries

Where the values have the following meaning:
- NB: Negative big
- NS: Negative small
- ZE: Zero
- PS: Positive small
- PB: Positive Big

The values of a and b can be different depending on the implementation. More levels can be added if more precision is needed. The levels can also be moved over the x-axis as a certain input needs to be prioritized. The inputs for the Fuzzification are the error difference ΔE and the error E. The computation of the error and the error difference can be computed freely but common approaches are:

$$E(n) = \frac{P(n) - P(n-1)}{V(n) - V(n-1)} \tag{3.6}$$

$$E(n) = \frac{I}{V} + \frac{dI}{dV} \tag{3.7}$$

Then using the error equation:

$$\Delta E = E(n) - E(n-1) \tag{3.8}$$

These values for ΔE and E can be transformed into a single linguistic value with the use of the following table:

TABLE II
FUZZY RULE BASE TABLE AS SHOWN IN [50]

E \ ΔE	NB	NS	ZE	PS	PB
NB	ZE	ZE	NB	NB	NB
NS	ZE	ZE	NS	NS	NS
ZE	NS	ZE	ZE	ZE	PS
PS	PS	PS	PS	ZE	ZE
PB	PB	PB	PB	ZE	ZE

Figure 3.7: Fuzzy logic rules [8]

The resulting value can be implemented as the change in duty cycle ΔD. The accuracy of the table in figure 3.7 is dependent on the experience and knowledge of the developer and is based on the power converter being used.

Current sweep

For the current sweep method, a sweep waveform is used for the PV array current. This will result in the I-V characteristics of the PV array updated at a constant time interval [8][33]. The MPP can then be computed from the characteristic curve at the same intervals. The function chosen for the sweep waveform is directly proportional to its derivative as: [33]

$$i(t) = k_1 \frac{di}{dt} \tag{3.9}$$

The solution of this differential equation can be calculated as follows:

$$i(t) = k_2 e^{\frac{t}{k_1}} \tag{3.10}$$

Here , k_2 is taken as the I_{mpp} at MPP. At MPP, the $\frac{dp(t)}{dt} = 0$. Using these equations, the following can be said:

$$\frac{dp(t)}{dt} = \frac{d(v(t)i(t))}{dt} = i(t)\frac{dv(t)}{dt} + v(t)\frac{di(t)}{dt} = 0 \tag{3.11}$$

$$\frac{dp(t)}{dt} = (k_1\frac{dv(t)}{dt} + v(t))\frac{di(t)}{dt} = 0 \tag{3.12}$$

Where $i(t)$ can be found from equation 3.9 and V_{mppt} from 3.12.

3.3.3. Effect of weather on the maximum power point

The weather changes the behaviour of the PV panels. One of the possible scenarios is a change in irradiance due to, for example a cloud shading the UAV or a sudden change in weather conditions. This changes the overall power output of the PV cells and it may lead to instability when tracking the MPP. [27].

With the hill-climbing algorithm, this could be a problem. Figure 3.8 shows the working of hill- climbing under changing atmospheric conditions.

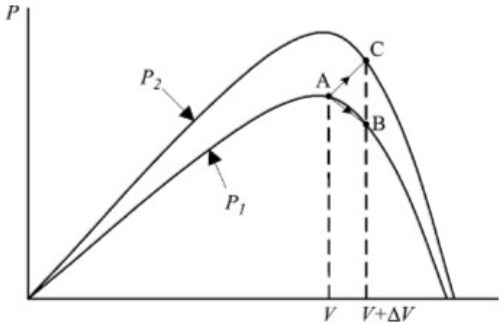


Figure 3.8: Divergence of MPP [8]

Here, P1 and P2 represent the curves for 2 different sets of atmospheric conditions. When the current operating point is point A, and the MPP is tracking curve P1, it becomes visible that in the next perturbation step it will see point B. It realises that the power at point B is lower so the perturbation sign is changed. But if the irradiance suddenly changes and point C is seen in the next perturbation step, it will think that it is still not at the MPP and thus the perturbation sign is not changed. Resulting in a loss of power.

This problem is less prevalent when using Incremental Conductance as proposed in [14]. Rather than hill climbing, the array terminal voltage is always adjusted according to its value relative to the MPP voltage.

For Ripple correlation control this problem is not a problem at all. The dynamic version of RCC is introduced in [9]. As in RCC there is no perturbation period but rather a ”dynamic perturbation” from the ripple, the tracking of the MPP is so fast that it takes into account the varying atmospheric conditions.

3.3.4. Local peaks problem

When multiple solar cells are combined to form a module, there are also by-pass diodes inserted. This results in multiple peaks in the P-V characteristic curve. [27] Introduces the concept of multiple peaks through partial shading. Figure 3.9 shows the P-V and I-V curve due to partial shading of the PV cells. The current that the shaded modules can generate is less than the current than un-shaded modules can generate. Two different conditions can arise: In the first case, string voltage is reduced since bypassed modules do not produce power and operate at bypass diode voltage. This is the low-voltage peak in the figure. In the second case, all modules conduct, thus voltage is higher, but current is reduced to the current of the shaded modules. This is the high-voltage peak in the figure.

One easy way of solving this, is introduced in [29]. Here a normal P/O algorithm is modified into a two stage algorithm. In the first part, the MPP is tracked using a large perturbation size in the P/O algorithm. As a result of this voltage sweep an approximate location (within a certain voltage range) of the MPP is detected. For Figure 3.9 this range can thus be around 17V. In the second stage, a small perturbation step is used which will oscillate within this approximate range of voltage. This approximation of the location of the global MPP by adapting the voltage range is then repeated every so often.

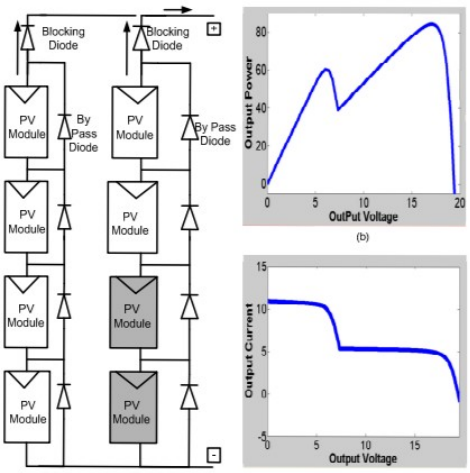


Figure 3.9: PV characteristics under partial shading conditions [29]

3.3.5. Effect of solar cell configuration or power converter

The paper [27] introduces the impact that the solar cell configuration might have on the power output thus the MPP tracking. This can also be seen from figure 3.9. The way the pv cells are integrated can affect their performance under partial shading as well as their variation in the power output.

Looking at the above algorithms, it becomes visible that the P/O algorithm and ICC only rely on the PV voltage and current. They can be easily adapted for any solar cell configuration or any power converter by only fine-tuning the perturbation period and perturbation step.

For the Ripple correlation control, this is different. As this is a mostly analogue implementation which uses the ripple induced from the power converter, the algorithm is dependent on the power converter. Making it less dynamical. [8][33].

Fuzzy logic control can depend on the solar cell configuration as the categorical values are implemented based on a certain solar cell configuration. It is thus not adaptable for any solar module.

3.3.6. Implementation of MPPT

For the implementation of the MPPT algorithm, three cases can be distinguished. The implementation can be either analogue, digital or a mix of both. When looking at P/O or ICC it becomes apparent that these are mostly digital implementations of the MPPT. This makes them easy to simulate and dynamical.

For RCC, the implementation is mostly analogue as presented in [9]. This makes it complex to simulate and not easily integrated. Current sweep has not been successfully implemented on a PV cell array, meaning that no possible documentation of how to implement this algorithm or debug this algorithm is available. Making it difficult to implement.

4 Design

4.1. Introduction

In this chapter the design process and the choices made based on the existing techniques and the requirements for the corresponding subsystems are investigated. First the MPPT design cycle is elaborated on and a final design is presented. Then the Albedo design cycle is presented and a final design is shown.

4.2. MPPT Design

For the MPPT design, the following choices need to be made. An algorithm needs to be chosen based on the literature study and the requirements. Then the optimization of the algorithm with all its parameters need to be determined. This section describes the MPPT design choices that were made.

4.2.1. Algorithm selection

From the literature study, the following table can be created based on the requirements for the design and implementation of the control system.

	Fast	*Accurate*	*Ease of simulation*	*Independent*	*Complex*	*Intuitive*
Hill Climbing	Moderate	Moderate	High	High	Low	High
Incremental Conductance	Moderate	High	High	High	Low	Moderate
Ripple correlation control	High	High	Low	Low	High	Low
Fuzzy logic	Moderate	Depends	High	Depends	Low	Moderate
Current Sweep	Moderate	Moderate	Moderate	Moderate	Low	High

From this table it becomes apparent that the algorithms most in line with the requirements described in chapter 2 are Hill climbing and Incremental conductance. For the implementation, the hill climbing algorithm was chosen as this algorithm is intuitive, making it easily understandable for the entire team and thus it enables easy integration of the entire system. Next to this, the P/O algorithm is dynamic and does not depend on the architecture of the power converter or solar panel configuration.

Although the Incremental conductance algorithm shows an overall better performance then the P/O algorithms in some cases [23], their implementation in simulink is quite similar so it was arguably smarter to start with the more intuitive approach and then, if the result does not meet our standards, move to ICC or adapt the P/O algorithm.

The base P/O algorithm is not the fastest or most accurate implementation of MPP tracking. The anticipated result will thus be a base MPPT algorithm which is dynamic and optimal but might not be fast.

4.2.2. Breakdown of algorithm

Figure 4.1 shows the base perturb and observe algorithm. The PV panel voltage and current, V_{PV} and I_{PV} respectively, are sensed and the PV power is calculated. Then the power perturbation (ΔP) and the voltage perturbation (ΔV) are obtained by looking at the current and the last known power and voltage.

Then based on the table 3.1, the new perturbation direction of the duty cycle is calculated, added to the last known duty cycle fed to the power converter. This is then converted into a pwm signal and changes the switching behaviour of the power converter. The output voltage of the power converter is thus controlled by the equation:

$$D = 1 - \frac{V_{in}}{V_{out}} \tag{4.1}$$

Where D is the duty cycle, V_{in} is the PV output voltage and V_{out} is the power converter output voltage. Vin is thus changed to track the MPP through a variation of the duty cycle.

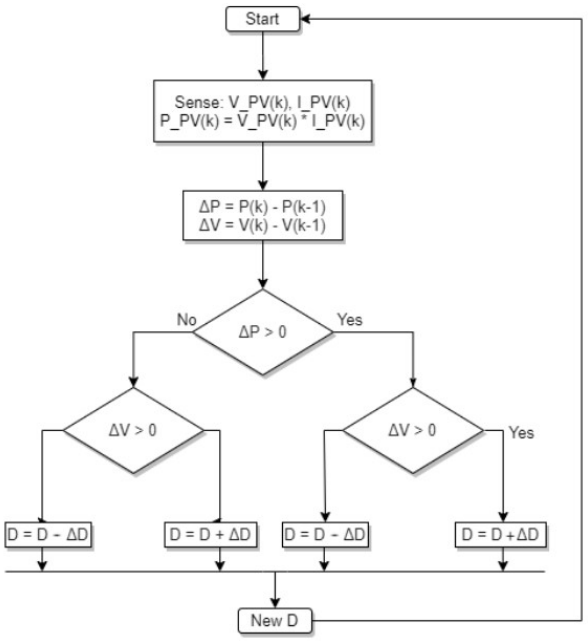


Figure 4.1: Base algorithm P/O

4.2.3. Variable step analysis

One problem with the P/O algorithm is the fact that is not fast enough and not robust enough under varying conditions. To account for all these factors, a variable step adaptation to the base P/O algorithm can be made.

For this variable step, the algorithm proposed in [16] can be used. Looking at the general behaviour of the voltage and power vs time (Figure 3.4), it can be noted that when the voltage and power reach steady state, the variation between them becomes small. In other words:

$$\left|\frac{\Delta P}{\Delta V}\right| < e \tag{4.2}$$

Where e is maximum allowable error, which is determined empirically during the test phase.

From this formula and the respective graphs, a divide can thus be made. If $\frac{\Delta P}{\Delta V} < e$ then a small perturbation step is used as the algorithm has (almost) reached steady state. This reduces the steady state oscillation. When $\frac{\Delta P}{\Delta V} > e$ the perturbation step will be large as the algorithm is not close to steady state.

4.2.4. Final design

For the final design, the base algorithm is enhanced with a variable step algorithm. Which, when combined, make up the final algorithm which is the algorithm displayed in figure 4.2.

Here, after the power and voltage perturbations are calculated, the size of the duty step is determined based on formula 4.2. For the specific solar panels in the system the ”big” perturbation step was chosen as $\Delta D_{big} = 0.02$ and the ”small” perturbation step was chosen as $\Delta D_{small} = 0.01$. The algorithm now displays faster convergence to a steady state with less oscillation around the steady state. A bigger version of Figure 4.2 can be found in Appendix B.1.

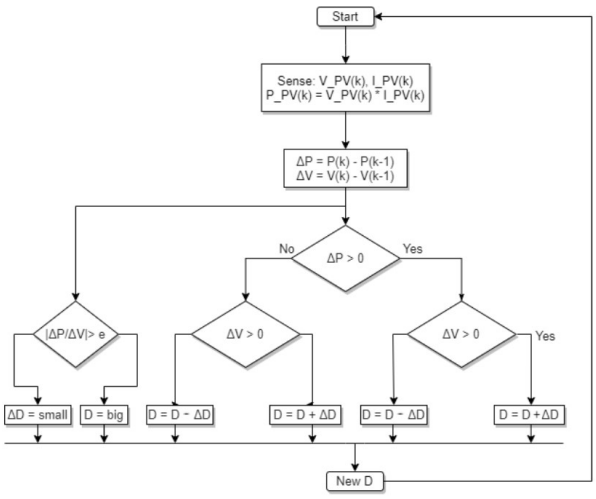


Figure 4.2: Variable step size algorithm P/O

4.3. Albedo Map Post Processing

Figure 4.3 provides a high level overview of the complete post processing pipeline. First the RAW image is loaded and demosaiced to provide the digital numbers (DN) the program can work with. Then the vignette effect is corrected and the user is presented with an image to annotate where the reference target is. The color channels of the digital numbers are averaged and the image is calibrated using the reference target and it's known reflectance. Finally these individual albedo images are fused together using Open Drone Map, providing the complete albedo map.

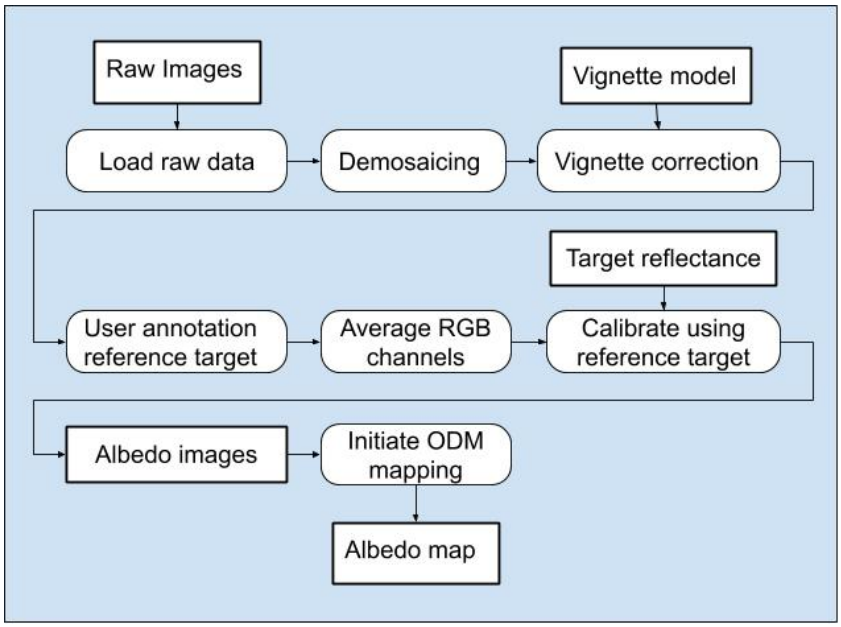


Figure 4.3: Image processing pipeline

4.3.1. Vignette correction

Based on the literature research in Chapter 3 a polynomial function was used to approximate the effect that vignetting has on the image. It was observed from the sample images that the vignette effect from the camera of the Nokia 6.1 used to capture our Raw images was radially symmetrical. Our polynomial vignette model was therefore chosen to be of the form:

$$c_0 + \left(\sum_{n=1}^{N} c_n r^n\right) + c_{n+1} y + c_{n+2} x \tag{4.3}$$

Here r is the radial distance from the center and y and x are the vertical and horizontal offset from the center. By dividing the image by the vignetting model the image is corrected. After this is done, the brightness of pixels in the image will be independent of the location within the image.

Based on our testing, a polynomial of 4 was taken. However this is completely dependent on the camera and a different value might be required for other cameras. This will be elaborated further on in the validation chapter.

4.3.2. Post processing

The RAW camera digital numbers (DN) can be expressed to be linear with the integral of the reflectivity of the material times the solar irradiation that hit that surface.

$$DN = c \int_{\lambda} \rho(\lambda) S(\lambda) d\lambda \tag{4.4}$$

DN represents the Digital Numbers the camera produces, c is the factor representing camera sensitivity in the visible band, ρ is the reflectively of the surface, S is the incoming solar radiation on the surface, the range of wavelengths λ is determined by the response of the camera.

A reference target with a known spectrum-independent reflectively ρ_{rt} is used. This reference target acts as a proxy for the current light condition.

$$DN_{rt} = c\rho_{rt} \int_{\lambda} S(\lambda) d\lambda \tag{4.5}$$

One example of such a reference target is a Teflon target [15] which is also mentioned to be used in [30]. In our situation, only limited materials were available. Therefore, a white A4 paper was used as a reference target.

Knowing the reference target's visible band reflectance allows for the relative incoming solar irradiance to be approximated. The digital numbers in a user's defined reference target region are averaged and divided by the user's provided reflectance to obtain a reference value.

The red green and blue channels are averaged and then all pixels in the image are divided by the reference value to obtain the visible band albedo:

$$\frac{DN}{DN_{rt}/\rho_{rt}} \approx \frac{\int_{\lambda} \rho(\lambda) S(\lambda) d\lambda}{\int_{\lambda} S(\lambda) d\lambda} = \alpha_{vis} \tag{4.6}$$

The current process makes a couple of assumptions.

The first assumption is that the response to the red, green and blue channel is equal. Ideally the spectral response of the sensor is known. This way the contributions of the red, green and blue could be compensated accordingly.

Furthermore the view direction has not been taken into account and it is assumed that the surface is an lambertian emitter. Estimating an accurate bidirectional reflectance distribution function for every surface would have been out of the scope of this project.

Finally our process produces a visible band albedo value. This is due to the limitations of the available camera equipment.

5 Implementation

5.1. Introduction

This chapter elaborates on the implementation of the MPPT and the albedo generation according to the design generated in chapter 4. Simulink is used for the modelling of the MPPT. For the albedo map generation, the processing pipeline is created in python.

5.2. MPPT algorithm

For the MPPT algorithm, the implementation of the base algorithm in Simulink is investigated. Then the variable step algorithm is added, the general test setup is evaluated and finally, considerations and adaptations to the general model are given.

5.2.1. Base model of P/O algorithm

Figure 5.1 shows the base level implementation of the hill climbing algorithm in Simulink. On the left, the PV array parameters enter in a bus signal. For the MPPT, only the PV current and the PV voltage are of importance so these are selected with a bus selector. The PV current and voltage are then low pass filtered and analogue to digital sampled.

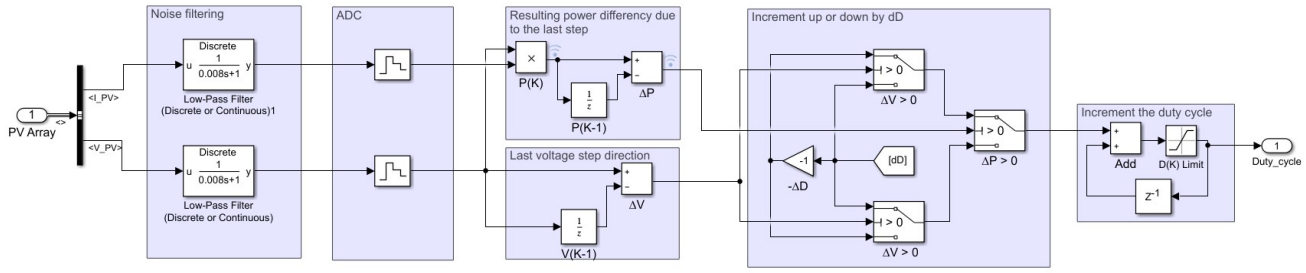


Figure 5.1: Base model implementation of P/O

Next, the perturbation in voltage (ΔV) as well as the perturbation in power (ΔP) are calculated. For this, the last known voltage and power are obtained with the $1/Z$ unit delay. Then, on the second to last block in fig 5.1, the duty cycle perturbation is calculated. In the base model case, the dD is set to 0.01. So the difference in duty cycle per step is ± 0.01.

Then in the last block, the perturbation in duty cycle is added by the last known duty cycle and the new duty cycle is outputted to the Power converter.

Pre-processing of signals

Due to the switching nature of the power converter, the current and the voltage are not smooth curves. Rather, their behaviour exhibits a ripple. To filter out this ripple, the signals are low-pass filtered. For this low-pass filter, a cut-off frequency of 10x the perturbation frequency is used as this ensures systematic changes propagate to the final sampled value but higher frequency noise is filtered out.

To test our implementation of the MPPT algorithm, an analog to digital converter is used to sample the voltage and current signal at the perturbation period.

5.2.2. Test setup

To test the base model, the pv array with its PV current and voltage needed to be simulated as well as the corresponding power converter. Figure 5.2 shows the test setup for the base model. On the left, the Simulink inherited PV array is used to mimic the behaviour of the PV cells on the UAV. The following block represents the P/O algorithm to be tested. It obtains the PV current and voltage from the PV block and outputs the duty cycle to the power converter. The P/O algorithm is followed by the power converter system block. This block contains a boost converter that mimics the behaviour of the implemented boost converter. Then a controlled voltage source and a load are attached to mimic the UAV battery and load.

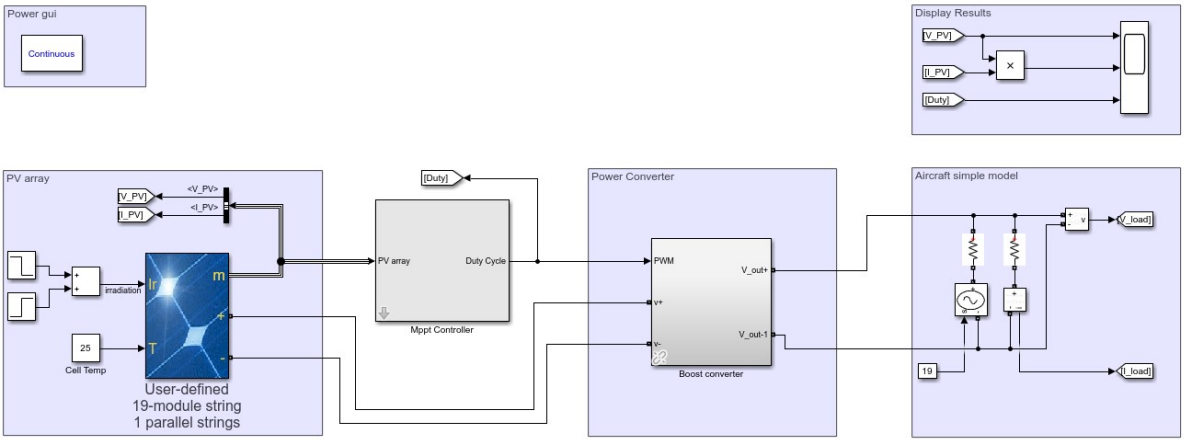


Figure 5.2: Test setup for the base model

The test setup works with 3 different time bases. The time base of the simulation of the entire model (model), the time step with which the voltage and power perturbation are calculated and the switching frequency of the power converter.

	Model	MPPT	Power converter
Time base	Continuous	Δt = 0.01s	F_s = 150kHz

The perturbation period of the MPPT algorithm should be long enough so that the PV and power converter have reached steady state after the previous perturbation. Otherwise the voltage and current readings might be different from the actual values at the corresponding duty cycle. Therefore the perturbation period (Δt_{MPPT}) has to be larger than the settling time ($\Delta t_{settling}$) which was determined by the power electronics team to be 5.0 ms.

$$\Delta t_{MPPT} > \Delta t_{settling} \Longrightarrow \Delta t_{MPPT} > 5.0ms \tag{5.1}$$

To ensure that the system adheres to this, a perturbation period of 10ms is chosen.

5.2.3. Integration of variable step

The current tracking is slow. This process can be sped up by implementing a variable step according to the final design displayed in figure 4.2. The block on the right bottom displays the variable step system.

Following equation 4.2, the error rate is determined. Empirically, the optimum error to differentiate between the dD_{big} and dD_{small} was chosen as 2.5.

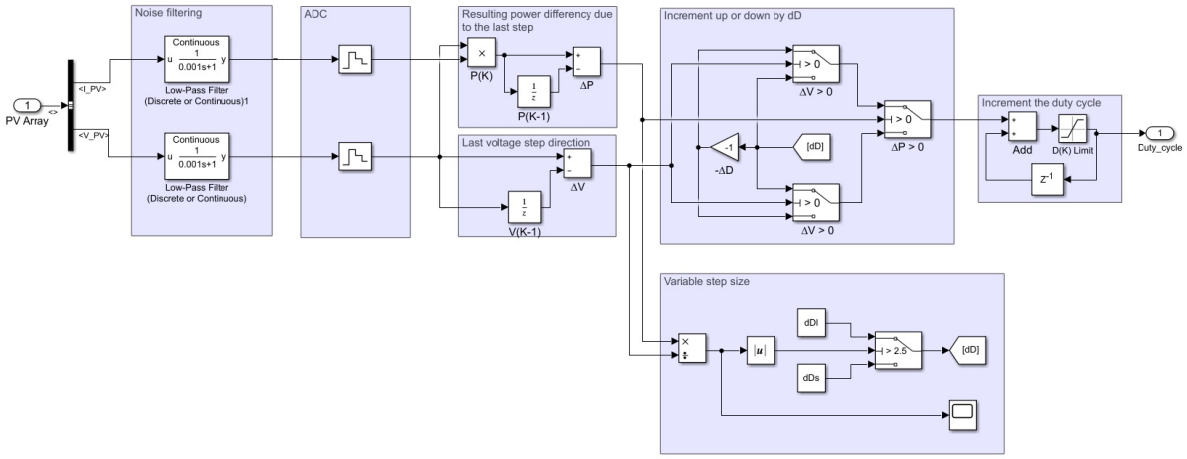


Figure 5.3: Variable step base implementation

If the absolute rate of change between the power and voltage is bigger than 2.5 then the duty cycle step becomes $dD_{big} = 0.02$. Otherwise the duty cycle step stays $dD_{small} = 0.01$.

5.2.4. Optimization of MPPT performance

For the design to work as optimal as possible, the parameters of the current implementation can be tweaked to increase the power efficiency. First, the system needs to be optimized for the duty cycle step. Looking at the equation 4.1, it can be deducted that a difference in duty cycle directly relates to a difference in the ratio $\frac{V_{in}}{V_{out}}$. If the V_{out} is assumed to be relatively constant at 18V, this means that the optimal V_{in} can be tracked with a resolution of $18 * \Delta D$.

From simulation, the optimal V_{in} is around 11.5V. If ΔD is then chosen at 0.01 around the MPP, the maximum error around the MPP is ±0.18V which relates to a power ripple of 0.24W. This results in an accuracy of 0.12W around the MPP or a power efficiency of 99.8% when the solar irradiance is $800 W/m^2$ and the cell temperature is 25 degrees Which adheres to our requirements.

Different variable step algorithms

Secondly, for the optimization of the variable step algorithm, multiple options were considered and implemented. These options include:

- A proportional variable step algorithm which assumes the optimal voltage is around 11V and determines the duty cycle perturbation based on the voltage offset.
- A variable step algorithm which keeps track of the steady state and will enable a sleep state when steady state is reached as discussed in [28].

The result, however, was that the basic two step approach performed similar to more complex implementations.

5.2.5. Physical implementation of MPPT

For the physical implementation of the MPPT, a microcontroller was chosen. For our MPPT, the Adafruit ItsyBitsy 32u4 [1] containing an ATmega32u4 microcontroller [5] was chosen. This board is fast enough to provide the PWM signals to control the power converter and contains analog to digital converters to read the voltage and current sensor outputs.

Table 5.1: ItsyBitsy 32u4

Operating voltage	5 V
Clock rate	16 MHz
Analog inputs	6
Additional PWM outputs	4

5.3. Albedo post processing

Based on the proposed design pipeline explained in Chapter 4, the actual pipeline can be implemented. For the implementation of the post-processing pipeline, Python is used. Next to this, Numpy, Matplotlib, Sklearn, Rawpy and ODM are used for further implementation of the processing. The implementation of the albedo map generation involves the loading and demosaicing of the raw images taken by the camera, the vignette correction, the annotation of a reference target and eventually, the orthomosaic map generation.

5.3.1. Loading and demosaicing Raw images

Rawpy was used for loading and demosaicing the raw images since it accepted a broad range of input images and provided manual control over the post-processing.

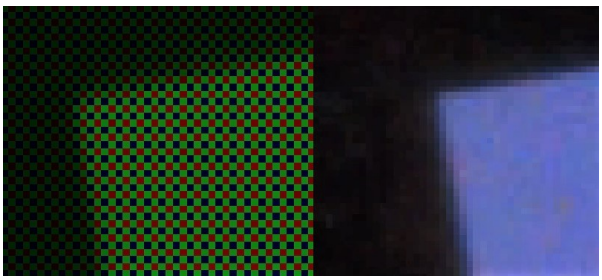


Figure 5.4: Left: Bayer pattern of the raw image data, Right: Demosaiced using AHD algorithm

Modern CMOS camera sensors consist of a single sensor with red, green and blue passing filters in front of individual pixels layed out in a bayer pattern, Figure 5.4. For the demosaicing of this bayer pattern, the Rawpy's Adaptive Homogeneity-Directed (AHD) [13] implementation is used. This provides a higher resolution when compared to down-sampling of the bayer pattern as performed in [20] while also preventing discoloration as a result of the interpolation of the missing color values.

Two different images are generated after post-processing is performed by Rawpy in our pipeline.

The first image is only used to display the image to the user during annotation. This image is processed using the default post-processing settings of Rawpy. This provides better visibility of the content of the image as compared to an image with a linear gamma curve where the digital numbers linearly correspond to the brightness.

The second image is used for the actual albedo processing. Therefore a linear relationship between the measured intensity and the digital numbers, without any gamma correction is desired. Furthermore auto-brightness has been disabled and the intermediate values that are used during processing have a 16 bit resolution.

5.3.2. Vignetting correction

The vignetting correction consists of two processes. Firstly the model coefficients have to be acquired based on a noisy reference image of the vignette. This has to be done only once. However, it is worth noting that the camera zoom and aperture effect the vignetting, so these camera parameters should be the same for all images. Once the model coefficients are determined, the program may recreate the vignetting model at any time to correct an image.

Obtaining the model coefficients

For every pixel in the vignetting image the corresponding x, y, r, r^2, r^3 and r^4 features are calculated. The center of the image is taken as the origin for x and y. These values are then normalized and the 2 dimensional matrices are reshaped to 1 dimensional vectors. These vectors are concatenated to provide a matrix with every row containing the features for the corresponding pixel.

1st pixel	1	r_0^4	r_0^3	r_0^2	r_0	y_0	x_0
2nd pixel	1	r_1^4	r_1^3	r_1^2	r_1	y_1	x_1
3th pixel	1	r_2^4	r_2^3	r_2^2	r_2	y_2	x_2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 5.5: Features for the linear regression

The reference image, containing the noisy vignette, is reshaped to a column vector. This will provide the samples that our model should try to approximate with a linear combination of the features in Figure 5.5. To find these coefficients the linear regression function from Sklearn is used. These 7 coefficients are now sufficient to model the vignette.

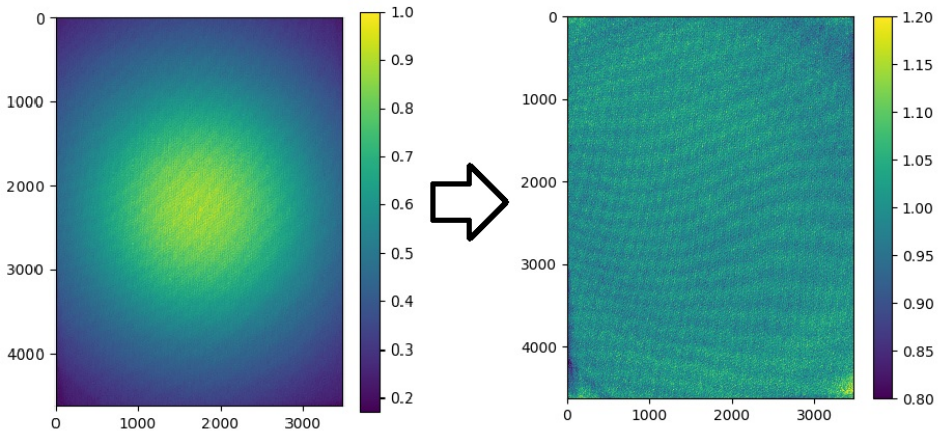


Figure 5.6: Result of dividing the vignette image by the vignette model

Correcting the vignette

Once vignette has to be corrected in a different image, the vignette model is generated from the model coefficients and the image pixels are divided by the vignette model pixels.

5.3.3. Reference target annotation and correction

For the reference target annotation, Matplotlib is used to plot the regular image. The user is then able to drag across the image to select an area. Once the user is in contempt with their selection and closes the image, the red green and blue channels of the linear image are averaged. The values in the area of the user selection are also averaged and the image is calibrated with this value and the corresponding target reflectance.

Now the calibrated albedo image is saved in its designated folder for later use in the orthomosaic map generation.

5.3.4. Orthomosaic map generation

ODM was used to implement the Orthomosaic image stitching. WebODM provides a web interface for loading the images and viewing the final result.

To use WebODM, a local server was set up in a docker container hosting the service on port 8000. After which, WebODM can be accessed by navigating to *dockerip*/8000 in a browser where the *dockerip* is the ip address of the docker machine.

ODM has many different parameters to configure. For this project it is important that ODM doesn't alter the values of the albedo images. Furthermore, the vignette has already been corrected and the brightness between the images should be directly comparable due to the calibration. Therefore the setting 'texturing-skip-global-seam-leveling: true' is set.

5.3.5. Physical implementation of the albedo map generation

The camera mounted on the UAV, as described in the general drone documentation, is used for the physical implementation of the albedo map generation. This camera is a Sony Alpha 6000. This camera is capable of capturing RAW images at the required resolution and speed. The company Agrowing sells special lenses for this camera providing multi-spectral imaging capabilities [2]. However, since we are not in possession of this camera nor the multi-spectral lenses we will not consider this during the validation.

6

Validation and Discussion

6.1. Introduction

After the design and implementation of the subsystems, it is important to look into what the results are and how valid or accurate the results are. This chapter touches on the results of the subsystems and explains the different assumptions that were made and how they influence the overall validity of the system.

6.2. MPPT

This section elaborates on the obtained results. It explains the efficiency of the algorithm and the factors influencing the obtained results. All figures can also be viewed in a larger size in Appendix B.

6.2.1. Results of the final design maximum power point tracker

Figure 6.1 shows the output of the model when implemented in the test setup. Here, the solar irradiance is visible in the bottom left corner and is altered from 800 W/m^2 to 200 W/m^2 at t = 0.5.

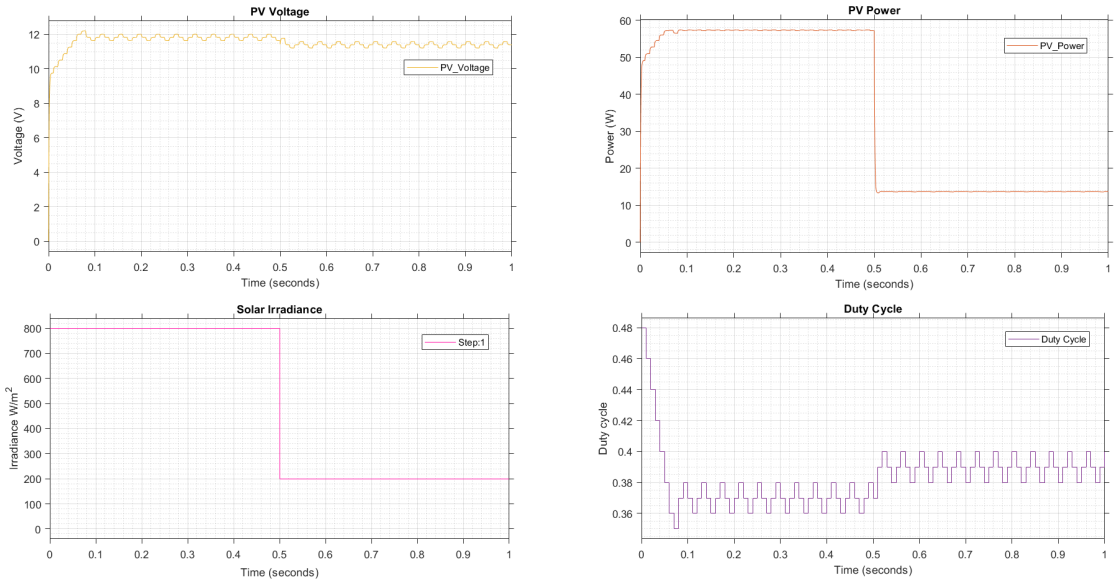


Figure 6.1: Performance of MPPT algorithm

The PV power is tracked and adapted accordingly, ensuring that our control algorithm works. In this figure, the implementation of the variable step also becomes visible. If the solar irradiance changes at t=0 and t = 0.5, the difference in $\Delta P/\Delta V$ becomes large and a big duty cycle step will be taken.

Tracking speed and accuracy

The tracking speed is defined as the time it takes the algorithm to converge the system to steady state power and voltage. Table 6.1 shows the performance of the algorithm in the best and worst case scenario.

Table 6.1: Tracking performance of MPPT

	System voltage	Solar irradiance	Tracking performance
Worst case scenario	21 V	900 $\longrightarrow$ 100 W/m^2	0.04 sec
Best case scenario	18 V	600 $\longrightarrow$ 400 W/m^2	0.02 sec

For the accuracy of the algorithm, a comparison between the optimal power curve generated by the solar panels in simulink and the optimal power adjusted by the duty cycle can be made. Table 6.2 shows a set of these comparisons. From this, it becomes visible that the power efficiency of the MPPT is about 99.29%. The power efficiency is calculated with the mid, min and max value of the simulation power over 3*the expected power.

Table 6.2: MPPT efficiency

Solar irradiance	Expected PV power	Simulation PV power	Power efficiency
800 W/m^2	57.87W	57.41W	99.3%
500 W/m^2	35.66W	35.36W	99.16%
100 W/m^2	6.72W	6.68W	99.40%

6.2.2. Comparison of variable step and base algorithm

As explained in chapter 4, the regular P/O algorithm is rather slow and a variable step extension can be implemented to enable faster tracking and faster convergence to the MPP. Figure 6.2 shows a comparison of the standard P/O algorithm and the variable step algorithm. From this, it becomes clear that the variable step algorithm is significantly faster when the solar irradiation changes.

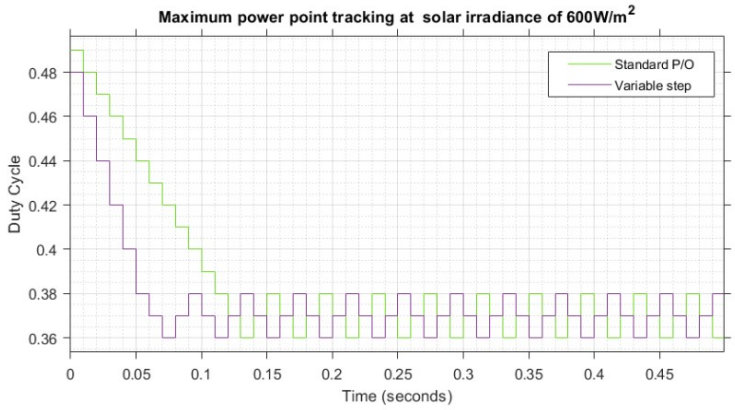


Figure 6.2: Comparison of base algorithm and variable step starting from initial conditions (duty cycle = 0.5)

In Figure 6.2 the variable step algorithm reaches the optimal duty cycle in 0.07 seconds whereas the standard P/O algorithm reaches it in 0.13 seconds. Meaning that it is 0.06 seconds faster or almost twice as fast.

6.2.3. Influence of different solar irradiances on duty cycle

The goal of the MPPT is to control the PV generator operating point. Figure 6.3 shows the performance of the MPPT algorithms when the solar irradiance changes during the simulation. In the figure, the variable step as well as the standard P/O are tested under these different solar irradiances. The bottom left curve in Figure 6.3 shows the change made in irradiation and on the bottom right the corresponding duty cycle change. Looking at the top right and top left curves, it becomes visible that the PV voltage and PV power are positively correlated with the solar irradiance.

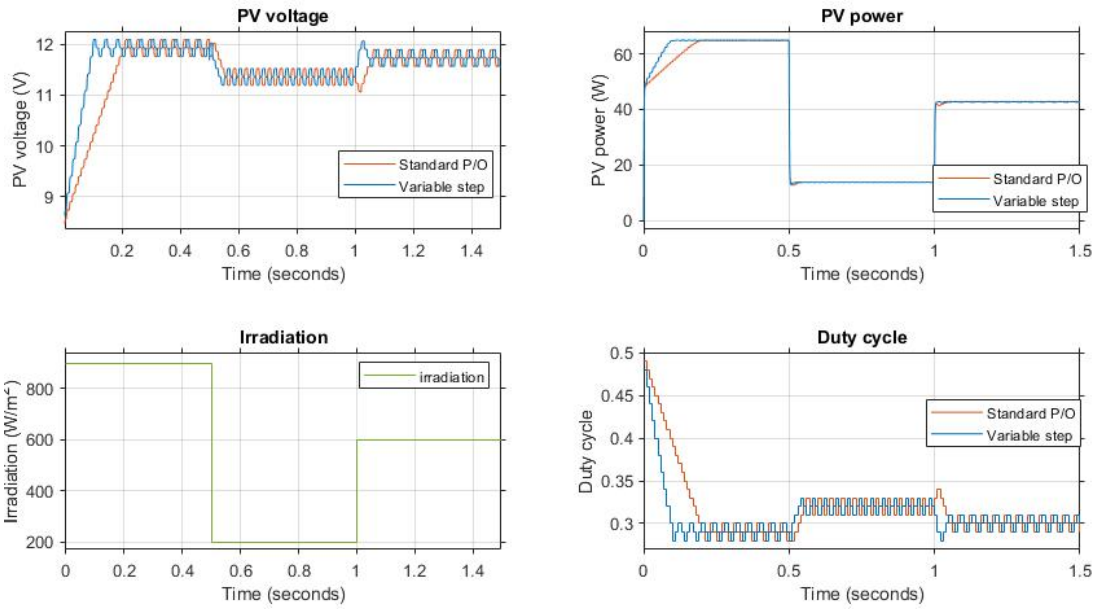


Figure 6.3: Performance of standard P/O and variable step under different solar irradiances

Then looking at equation 4.1, it becomes clear that a higher V_{in} at an almost constant V_{out} will result in a smaller duty cycle. This is also reflected in Figure 6.3. From this figure, the effect discussed in 3.3.3 also becomes visible when the solar irradiance changes suddenly.

6.2.4. Influence of system voltage on duty cycle

During flight, the battery will discharge slowly which could cause the PV voltage to fluctuate more. In order to investigate this, the battery voltage in the test-setup load (5.2) was changed during the simulation. Figure 6.4 shows the influence of changing the voltage system on the duty cycle whilst keeping irradiation the same. Although, in practice this change of system voltage is gradually, Figure 6.4 shows the impact different voltages have on the duty cycle.

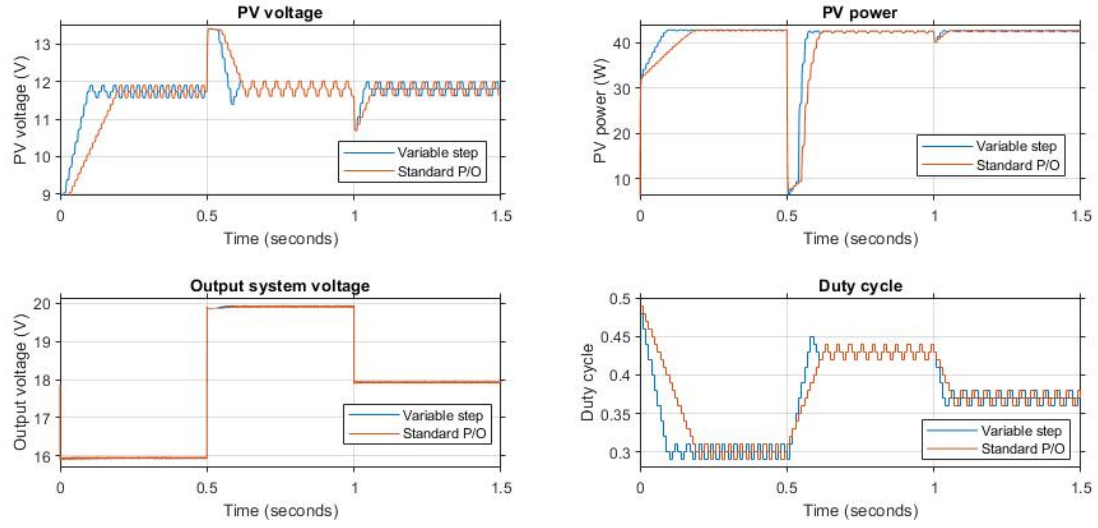


Figure 6.4: Algorithm at different system voltages

From this figure it becomes evident that the system voltage does influence the duty cycle. A higher system voltage means a higher duty cycle to extract the same PV power. This is expected as from equation 4.1 it can be seen that if Vout goes up, the duty cycle goes up for the same PV voltage.

6.3. Albedo

This section elaborates on the obtained results for the albedo map generation. First the results for the vignette correction is elaborated on, then the actual albedo results are presented. Lastly a comparison between the visible band albedo and the broadband albedo is made.

6.3.1. Vignette

To gather test data, multiple images of a solid white painting canvas were taken. All images were shot from a different angle to try to minimize error of potential directionality of the light condition. In total, 10 RAW images were captured as the data sample set.

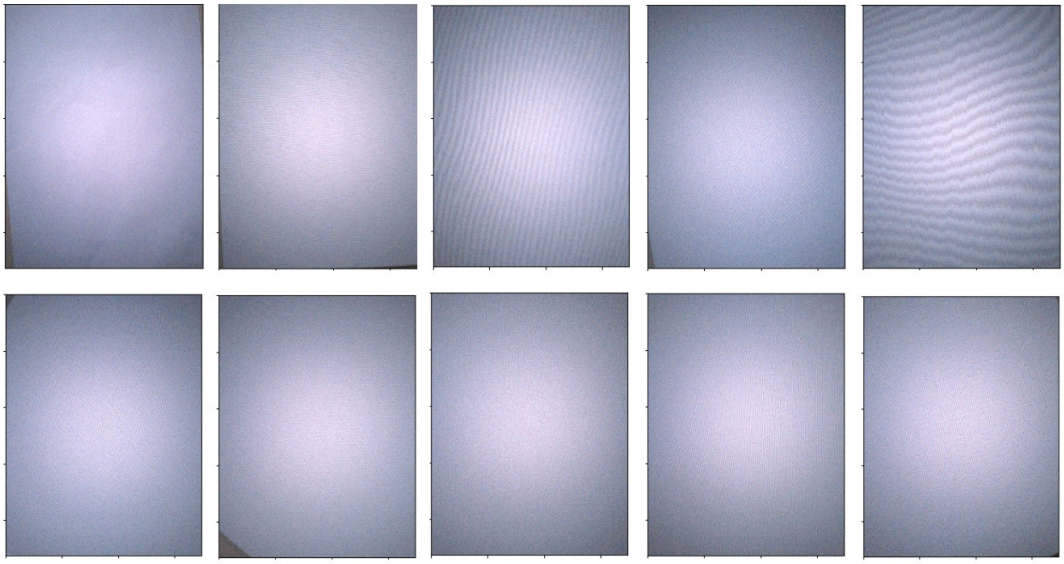


Figure 6.5: The 10 images used to determine and test the vignette

The albedo pipeline was used to load and process these RAW images. The center of the frame was calibrated to a brightness of 0.7 so that the different images would be comparable and to make sure no clipping would occur. These processed images where saved in a 8bit resolution where the digital numbers of the pixels corresponded linearly to the RAW intensities.

These different images were averaged together providing the vignette image as seen in figure 6.6. Using these images as the input "noisy" vignette, 6 different polynomial models of equation 4.4 with $N = \{1, 2, 3, 4, 5, 6\}$ are created as described in the implementation chapter.

In table 6.3 the coefficients of the different polynomial models are listed. For the coefficients for y and x the values -2.15e-02 and -9.72e-07 were determined in every model.

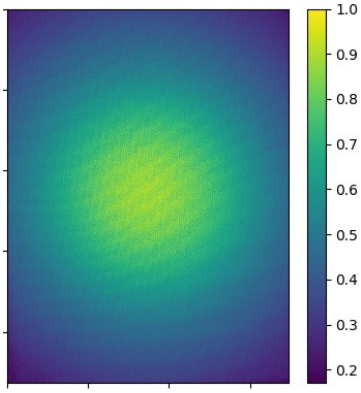


Figure 6.6: Calibrated and averaged vignette samples

Table 6.3: Coefficients found for the different polynomial vignette models

	1	r^6	r^5	r^4	r^3	r^2	r
$N = 1$	1.09e+00	-	-	-	-	-	-8.88e-01
$N = 2$	1.14e+00	-	-	-	-	2.04e-01	-1.10e+00
$N = 3$	1.07e+00	-	-	-	7.97e-01	-1.01e+00	-5.58e-01
$N = 4$	1.00e+00	-	-	-2.60e+00	6.07e+00	-4.62e+00	3.80e-01
$N = 5$	9.96e-01	-	4.76e-01	-3.80e+00	7.19e+00	-5.08e+00	4.60e-01
$N = 6$	1.03e+00	1.16e+01	-3.47e+01	3.73e+01	-1.61e+01	1.44e+00	-3.51e-01

Vignette correction result

In figure 6.8a the averaged vignette image of 6.6 is compensated by the different vignette models. It can be observed that the compensated images for a polynomial of 1, 2 and 3 still presented undesirable rings where, on average, the image was still darker or brighter than it was supposed to be. Starting from a polynomial of 4 the images start to lose this ringing structure and only the noise and imperfections in the test images are left.

This can be verified by examining the root mean squared (rms) error between the vignette image and the different models. In figure 6.7 the rms error values are displayed. Based on this finding a polynomial of 4 chosen for the vignette correction in our implementation.

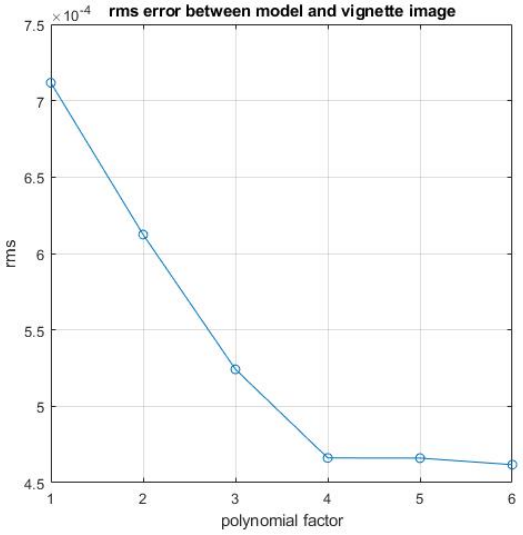
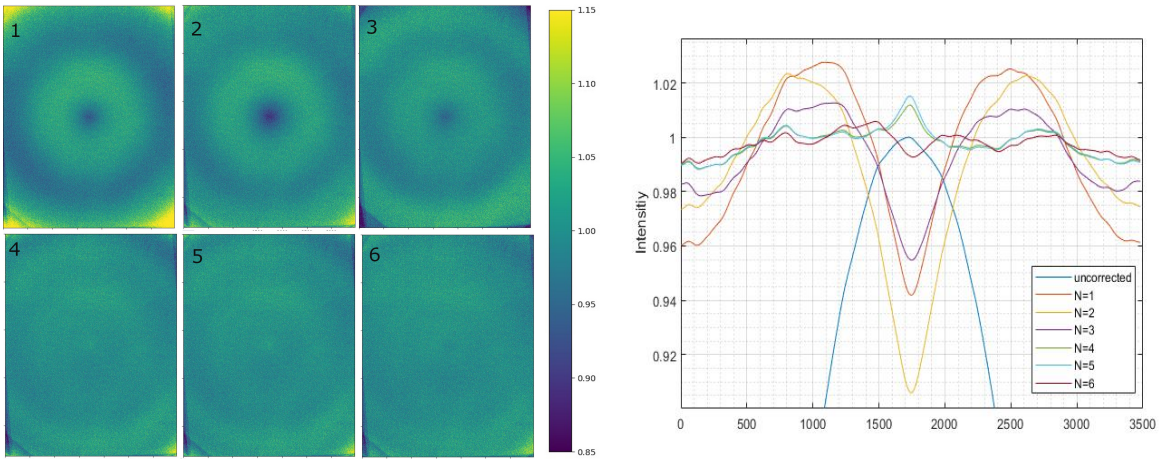


Figure 6.7: RMS error for different polynomials

In figure 6.8b intensity values along the x axis in the center of the images are shown. These values are smoothed using a gaussian blur in order to remove most of the effect of noise and only leave the systematic error. It can be observed that, starting from N=4, the maximum error is around 1%.



(a) Corrected vignette image

(b) Intensity on the center x axis

Figure 6.8: Average vignette images from figure 6.6 corrected using the different vignette models

6.3.2. Albedo map generation

To test the complete albedo mapping pipeline, 19 RAW images were taken from 4 different locations. In figure 6.9 these images are displayed. Ideally these images would have been facing down from different locations to provide more than 65% overlap, however, these images were only taken from 4 different positions due to practical constraints.

Figure 6.9: Albedo and image stitching test images

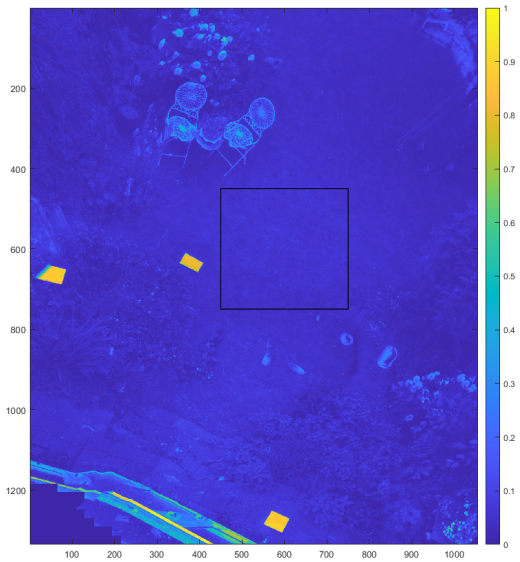
These images were taken during overcast light conditions to reduce uneven light conditions as a result of shadows, and to reduce the effect of specular reflections since a lambertian reflection of the surfaces is assumed. However, this also means that most of the illumination comes from diffuse sky radiation and will have a higher intensity in the blue spectrum than green and red.

This can be observed in the histogram of the reference target in figure A.2b in the appendix. As the visible band reflectance value of the a4 paper reference target, a value of 0.871 was used as measured in [6].

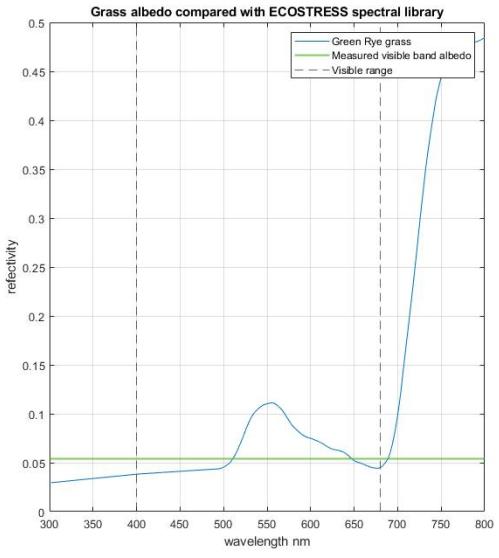
In figure 6.10a the result of the albedo mapping process is shown.

It can be observed that some deformation is present around the edges of the image. This is because the distance estimation requires multiple view directions. Furthermore, in the current pipeline the albedo images that are already processed are stitched together. However, these images have less contrast in darker areas making it harder to generate the point cloud in these regions.

The average of the calculated albedo from the grass within the black rectangle is 0.054. In Figure 6.10b this albedo value is compared to the reflectivity of rye grass from the ECOSTRESS library [3, 22] in the visible band spectrum. When assuming diffuse irradiation, a visible band albedo of 0.058 is calculated. These values seem to be close however more known test subjects would have been preferable. Furthermore, as will be discussed in the next section, there is a difference between visible and broadband albedo.



(a) Final albedo map



(b) Measured visible band albedo compared with the spectral reflectivity of rye grass from the Spectral Library [3, 22]

Figure 6.10: Albedo map generation results of the images from figure 6.9

6.3.3. Visible band albedo compared to wide band albedo

One important factor of the current albedo map generation is that it only uses the visible band wavelengths that a regular camera can capture. However, the solar irradiance spectrum extends beyond the visible wavelengths. Furthermore, the reflectivity of most materials is different depending on the wavelength.

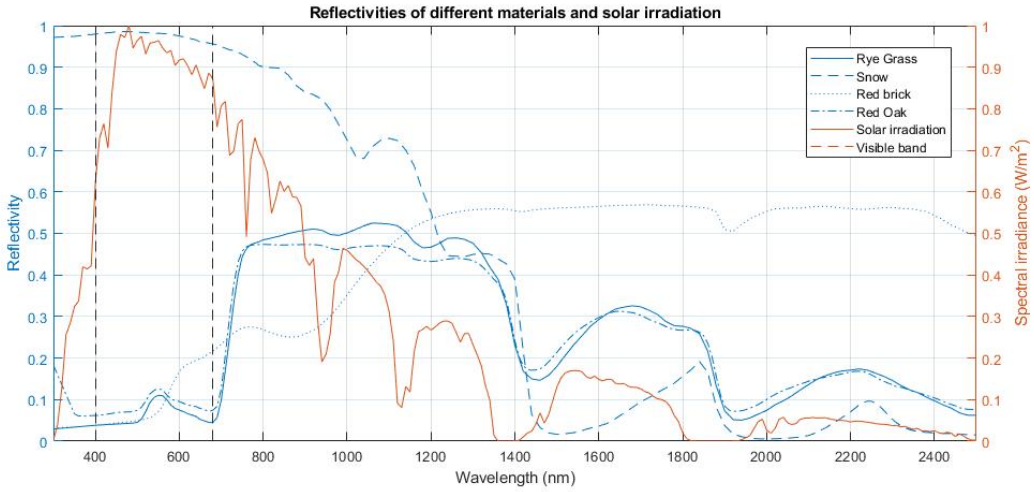


Figure 6.11: Comparison between different material reflectivities [3, 22] and AM1.5g solar irradiation [11, 21] at different wavelengths

When integrating the the AM1.5g solar irradiation [3, 11] from 400nm to 680nm and comparing it to the irradiation between 300nm and 2500nm it becomes clear the the visible band only contributes to approximately 42% of the total spectrum.

As displayed in figure 6.11 the spectral reflectivity of surfaces could differ significantly in the near infrared region as compared to within the visible band region. For instance, vegetation such as the grass and Oak have a lower reflectivity in the visible band while the reflectivity of snow is higher in the visible band.

In table 6.4 both the visible and broadband albedo are depicted. It becomes clear that the visible and broadband albedo, although related, can differ from each other. Furthermore, the ratio between visible and broadband albedo is material dependent.

Table 6.4: Broadband and visible band albedo values calculated using [21] and [3, 22]

Material	Broadband albedo	Visible band albedo
Rye grass	0.246	0.063
Snow	0.800	0.977
Bare red brick	0.251	0.103
Red Oak	0.249	0.086

7 Conclusion and Future work

The goal of the project was the design of a highly efficient maximum power point tracker and an accurate albedo processing pipeline. In this thesis, a successful MPPT has been implemented. With a resulting MPPT efficiency in the operating solar irradiance spectrum of about 99.3%, the result satisfies our preset requirements of at least 97%. Furthermore a processing pipeline has been created that converts RAW images into an albedo map by compensating the sensor vignette and calibration using a reference target. The final verification of measuring the albedo of Rye grass suggests that this process produces accurate visible band albedo results.

Although the current results are promising, additional improvements could still be made with regards to the current system. For the MPPT this relates to the tracking speed or accuracy. For the Albedo map generation this relates to considering wavelengths in the near infrared spectrum.

7.1. Recommendations

For the MPPT the following recommendations can be made.

- Adding support to track the global maximum power point with the presence of multiple local peaks could be implemented. This could improve the overall performance when the PV generator is partially shaded.
- Adding more variable steps or dynamically choosing the step size based on the ratio between $\delta P/\delta V$. This could increase the conversion speed of the current MPPT implementation.
- A sleep state can be implemented when the maximum power point is reached [28]. The current implementation oscillates around the maximum power point. Detecting when the MPPT is reached and staying at the mean value could increase the efficiency and allow the algorithm to detect what direction it should move when the irradiance does change.

The main recommendations of albedo map generation would be taking the near infrared region into account. Different approaches of broadening the detection spectrum could be considered.

- Adding a conversion factor based on the surface material. As discussed by [4] it could be possible to add a conversion factor for different material types like vegetation and non-vegetation.
- Using broadband pyranometers to measure the general downwelling irradiance and reflected irradiation from the ground. This could provide an accurate conversion factor to calibrate the camera without the need of reference targets. However, This does not take the difference in reflection spectra for different materials into account. When mapping a homogeneous surface like the ice sheet in [30] This wouldn't be a problem. However, if the surface is a mix of snow and vegetation, the vegetation would receive an albedo value that is too low as most of the irradiation reflected by the vegetation would be in the invisible near infrared region.
- Adding a NIR camera or multispectral camera to measure the near infrared wavelengths. This could provide accurate measurements of the additional spectral regions with the required resolution. Implementing a second camera on the system would require the separate images to be aligned and fused together in order to create the single albedo map.

Bibliography

[1] Adafruit. Introducing itsybitsy 32u4. URL https://learn.adafruit.com/introducting-itsy-bitsy-32u4.

[2] agrowing. Agro 60x00 dual based sensor. URL https://agrowing.com/products/alpha-6x00-dual/.

[3] Alice M Baldridge, SJ Hook, CI Grove, and G Rivera. The aster spectral library version 2.0. *Remote Sensing of Environment*, 113(4):711–715, 2009.

[4] Chang Cao, Xuhui Lee, Joseph Muhlhausen, Laurent Bonneau, and Jiaping Xu. Measuring landscape albedo using unmanned aerial vehicles. *Remote Sensing*, 10(11):1812, 2018.

[5] USB Controller. Atmega16u4/atmega32u4.

[6] KA Dornelles, Maurício Roriz, et al. A method to identify the solar absorptance of opaque surfaces with a low-cost spectrometer. In *Conference on Passive and Low Energy Architecture*, volume 23, page 2006, 2006.

[7] DXOmark. Sony e 16-50mm f/3.5-5.6. URL https://www.dxomark.com/Lenses/Sony/Sony-E16-50mm-F35-56-mounted-on-Sony-A6000---Measurements__942.

[8] T. Esram and P. L. Chapman. Comparison of photovoltaic array maximum power point tracking techniques. *IEEE Transactions on Energy Conversion*, 22(2):439–449, 2007.

[9] T. Esram, J. W. Kimball, P. T. Krein, P. L. Chapman, and P. Midya. Dynamic maximum power point tracking of photovoltaic arrays using ripple correlation control. *IEEE Transactions on Power Electronics*, 21(5):1282–1291, 2006.

[10] Simon Fuhrmann, Fabian Langguth, and Michael Goesele. Mve-a multi-view reconstruction environment. In *GCH*, pages 11–18. Citeseer, 2014.

[11] Christian Gueymard et al. *SMARTS2: a simple model of the atmospheric radiative transfer of sunshine: algorithms and performance assessment*. Florida Solar Energy Center Cocoa, FL, 1995.

[12] Teemu Hakala, Juha Suomalainen, and Jouni I Peltoniemi. Acquisition of bidirectional reflectance factor dataset using a micro unmanned aerial vehicle and a consumer camera. *Remote Sensing*, 2(3):819–832, 2010.

[13] Keigo Hirakawa and Thomas W Parks. Adaptive homogeneity-directed demosaicing algorithm. *IEEE Transactions on Image Processing*, 14(3):360–369, 2005.

[14] K. H. Hussein, I. Muta, T. Hoshino, and M. Osakada. Maximum photovoltaic power tracking: an algorithm for rapidly changing atmospheric conditions. *IEE Proceedings - Generation, Transmission and Distribution*, 142(1):59–64, 1995.

[15] Martin Janecek. Reflectivity spectra for commonly used reflectors. *IEEE Transactions on Nuclear Science*, 59(3):490–497, 2012.

[16] R. John, S. S. Mohammed, and R. Zachariah. In *2017 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)*, pages 1–6, 2017.

[17] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[18] Andrzej Kordecki, Henryk Palus, and Artur Bal. Practical vignetting correction method for digital camera with measurement of surface luminance distribution. *Signal, Image and Video Processing*, 10(8):1417–1424, 2016.

[19] Fabian Langguth, Kalyan Sunkavalli, Sunil Hadap, and Michael Goesele. Shading-aware multi-view stereo. In *European Conference on Computer Vision*, pages 469–485. Springer, 2016.

[20] V. Lebourgeois, A. Bégué, S. Labbé, B. Mallavan, and B. Roux L. Prévot. Commercial digital cameras be used as multispectral sensors? a crop monitoring test. *Sensors*, 8(11):7300–7322, 2008.

[21] Keith McIntosh, Malcolm Abbott, and Ben Sudbury. spectrum library, Jan 2014. URL pvlighthouse.com.au/resources/optics/spectrumlibrary/spectrumlibrary.aspx.

[22] Susan K Meerdink, Simon J Hook, Dar A Roberts, and Elsa A Abbott. The ecostress spectral library version 1.0. *Remote Sensing of Environment*, 230:111196, 2019.

[23] Shazly Mohammed and Montaser Abd El Sattar. A comparative study of p&o and inc maximum power point tracking techniques for grid-connected pv systems. *SN Applied Sciences*, 1:174, 02 2019. doi: 10.1007/s42452-018-0134-4.

[24] NASA. Measuring earth's albedo. URL https://earthobservatory.nasa.gov/images/84499/measuring-earths-albedo.

[25] OpenDroneMap. Drone mapping software. URL https://www.opendronemap.org/.

[26] Amit Patel and Devangi Thakkar. Geometric distortion and correction methods for finding key points:a survey. *International Journal for Scientific Research & Development|*, 4:311–314, 04 2016.

[27] H. Patel and V. Agarwal. Matlab-based modeling to study the effects of partial shading on pv array characteristics. *IEEE Transactions on Energy Conversion*, 23(1):302–310, 2008.

[28] Francisco Paz and Martin Ordonez. Zero-oscillation adaptive-step solar maximum power point tracking for rapid irradiance tracking and steady-state losses minimization. In *2013 4th IEEE International Symposium on Power Electronics for Distributed Generation Systems (PEDG)*, pages 1–6. IEEE, 2013.

[29] Hamdy Radwan, Omar Abdel-Rahim, Mahrous Ahmed, Mohamed Orabi, and Ahmed Mahfouz. In *Two Stages Maximum Power Point Tracking Algorithm for PV Systems Operating under Partially Shaded Conditions*, 12 2010.

[30] Jonathan C. Ryan, Alun Hubbard, Jason E. Box, Stephen Brough, Karen Cameron, Joseph M. Cook, Matthew Cooper, Samuel H. Doyle, Arwyn Edwards, Tom Holt, Tristram Irvine-Fynn, Christine Jones, Lincoln H. Pitcher, Asa K. Rennermalm, Laurence C. Smith, Marek Stibal, and Neal Snooke. Derivation of high spatial resolution albedo from uav digital imagery: Application over the greenland ice sheet. *Frontiers in Earth Science*, 5:40, 2017. ISSN 2296-6463. doi: 10.3389/feart.2017.00040. URL https://www.frontiersin.org/article/10.3389/feart.2017.00040.

[31] Chen Siyu and Mu Pingan Dai Shuguang. Correction of ccd camera lens vignetting. *Applied Mechanics and Materials*, pages 373–375, 2013.

[32] Jie Song and W Gao. An improved method to derive surface albedo from narrowband avhrr satellite data: narrowband to broadband conversion. *Journal of Applied Meteorology*, 38(2):239–249, 1999.

[33] B. Subudhi and R. Pradhan. A comparative study on maximum power point tracking techniques for photovoltaic power systems. *IEEE Transactions on Sustainable Energy*, 4(1):89–98, 2013.

A

Albedo generation step by step

In this appendix the intermediate steps of processing a raw image to obtain an albedo image are displayed.

Figure A.1: a) Original raw input image that is to be processed. b) The linear demosaiced image with vignette corrected

In figure A.2 the annotation process is displayed. The red rectangle on top of the white paper is the user selection. It is important that the reference target is not overexposed. Therefore the intensities of the reference are checked to ensure that they are not near, or at the maximum intensity. It can be observed that the blue channel has a higher response than the red and green channel. This can be explained by the lighting conditions the images where shot at. As explained in the validation chapter, the sample images are shot at an overcast day resulting in most of the light contribution coming indirectly from the sky.

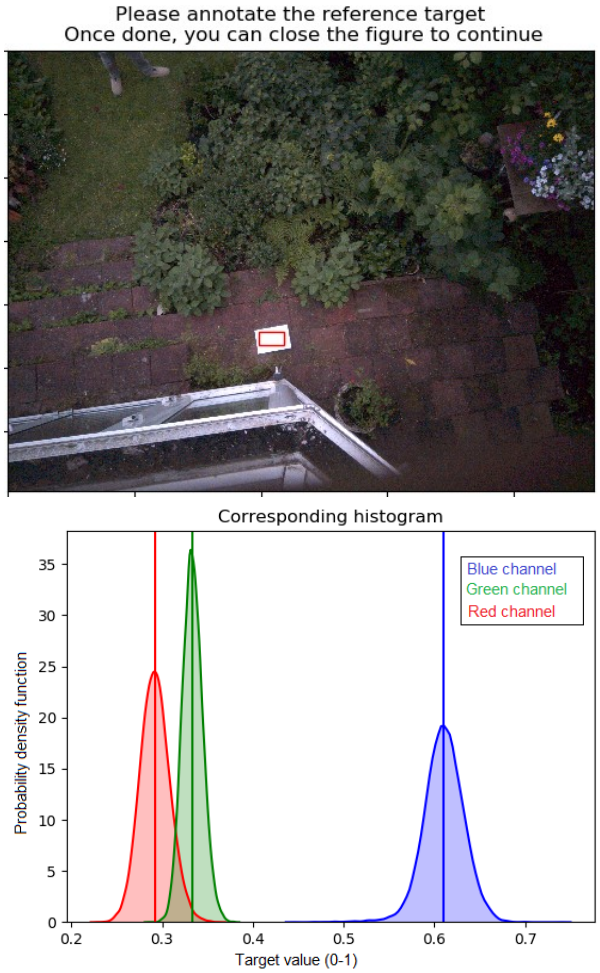


Figure A.2: a) The image for annotation b) The intensities of the different color channels of the reference target

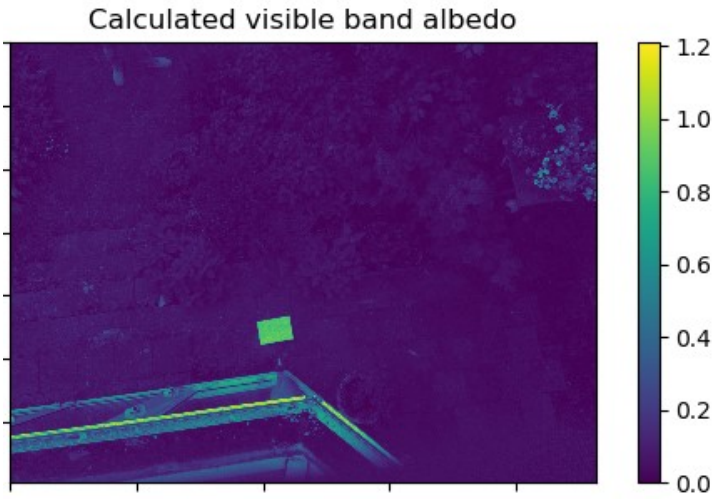


Figure A.3: Calculated visible band albedo

B

Validation figures

B.0.1. Final algorithm

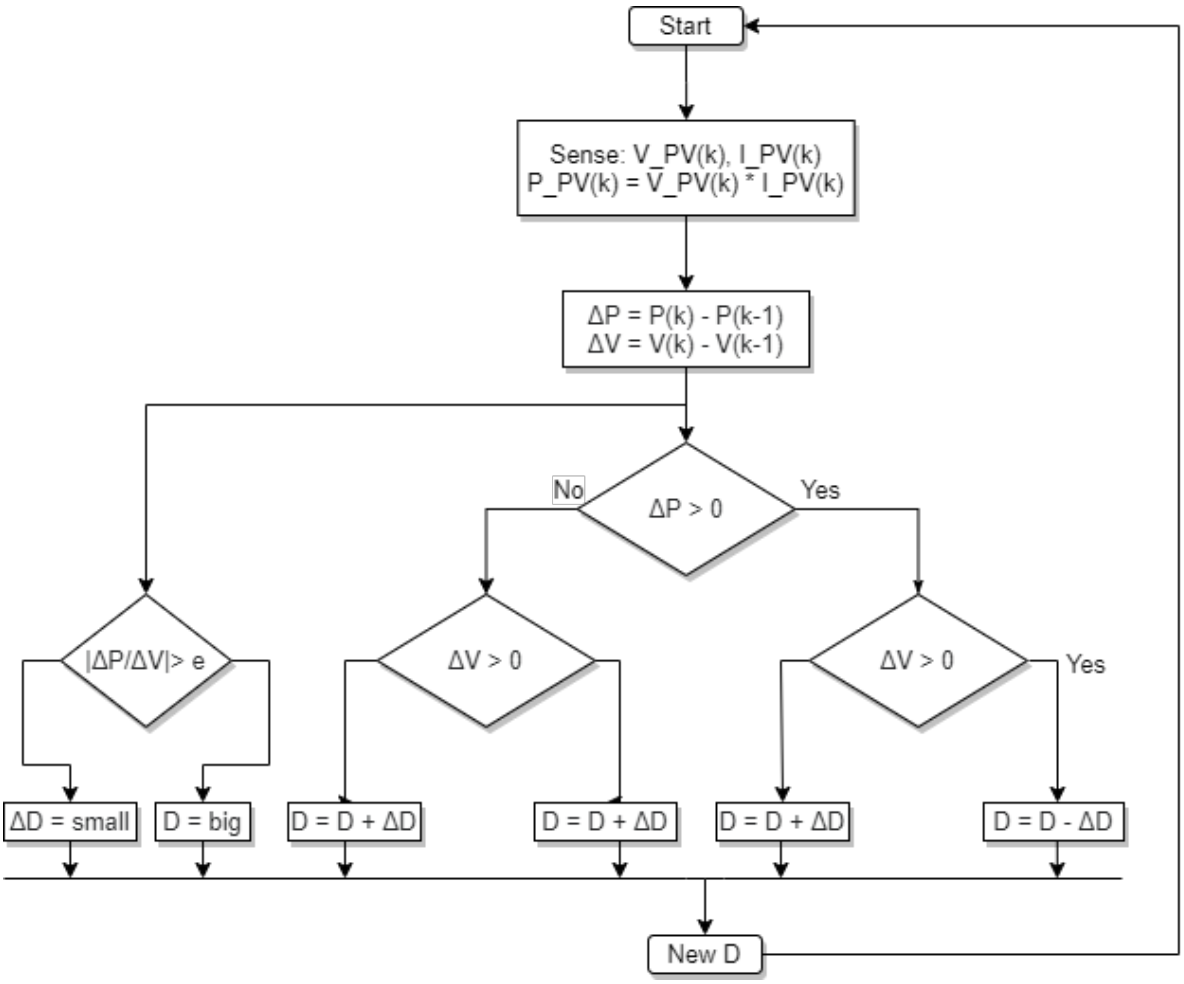


Figure B.1: Final MPPT design

B.0.2. Comparison of variable step and base algorithm

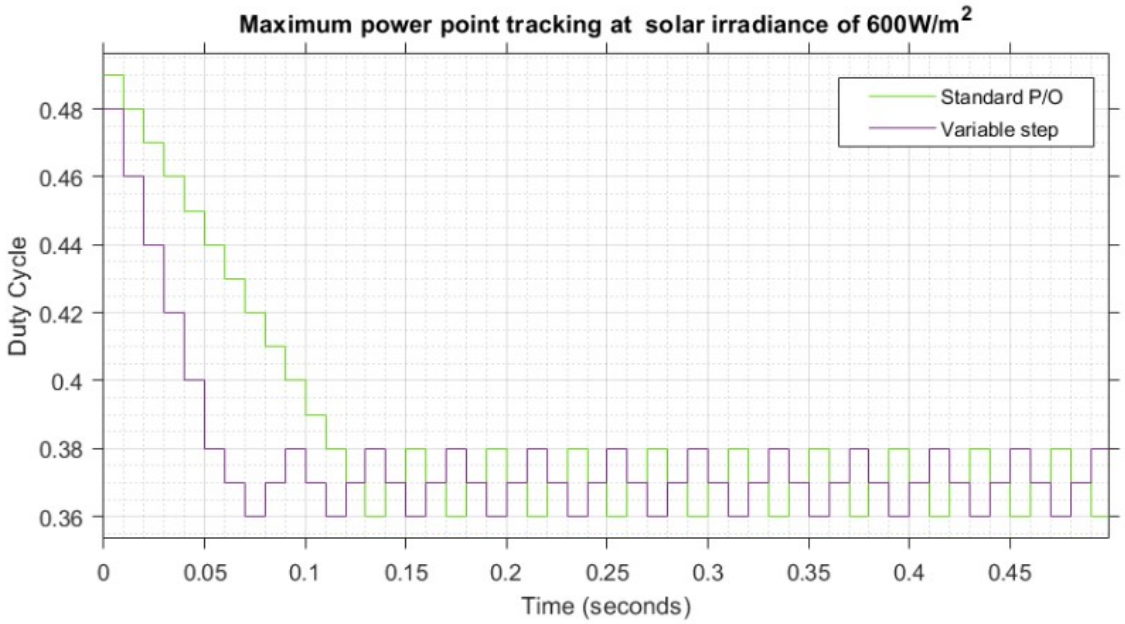


Figure B.2: Comparison of base algorithm and variable step starting from initial conditions (duty cycle = 0.5)

B.0.3. Final results

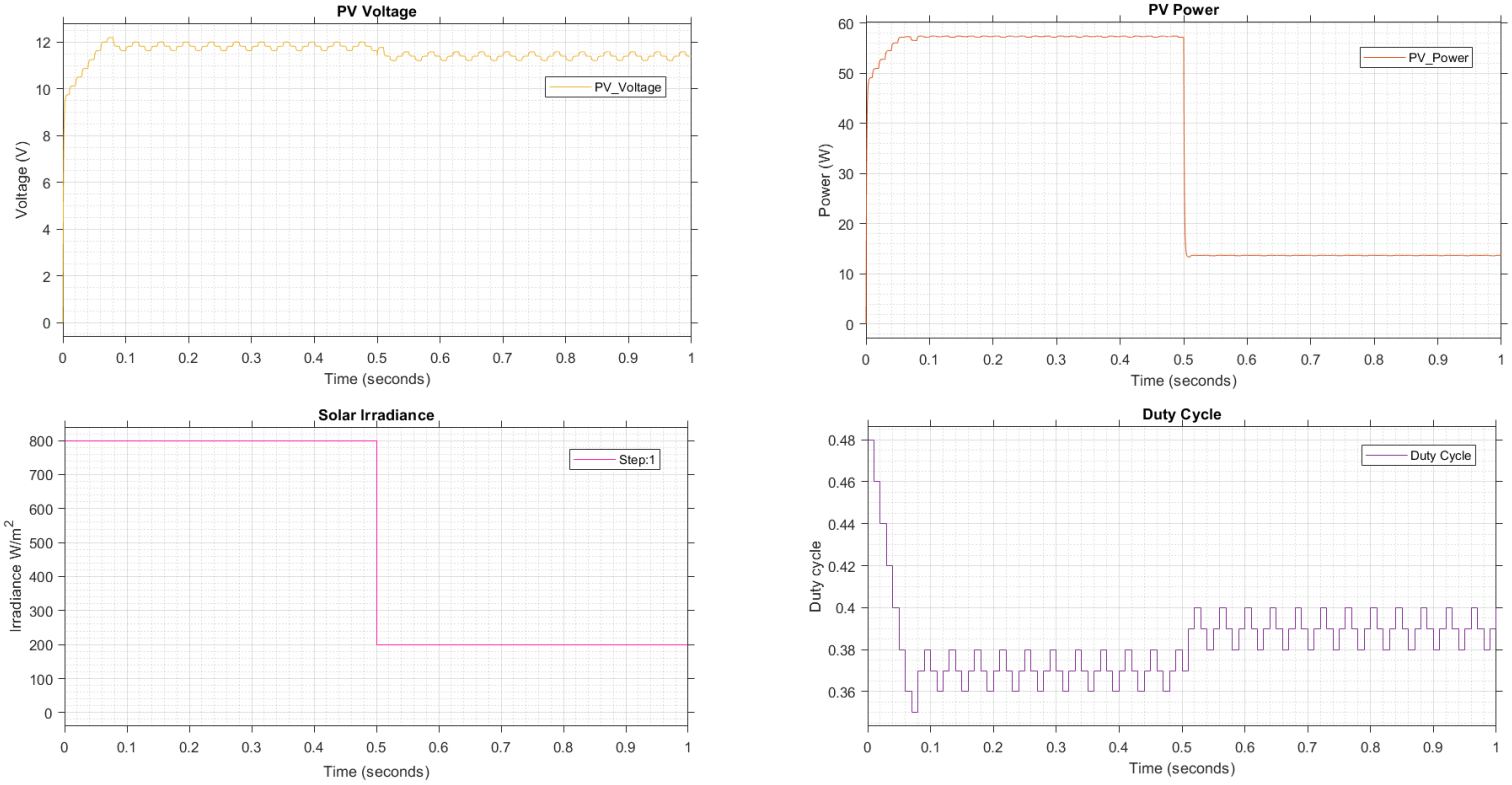


Figure B.3: Performance of MPPT algorithm

B.0.4. Influence of different solar irradiances on duty cycle

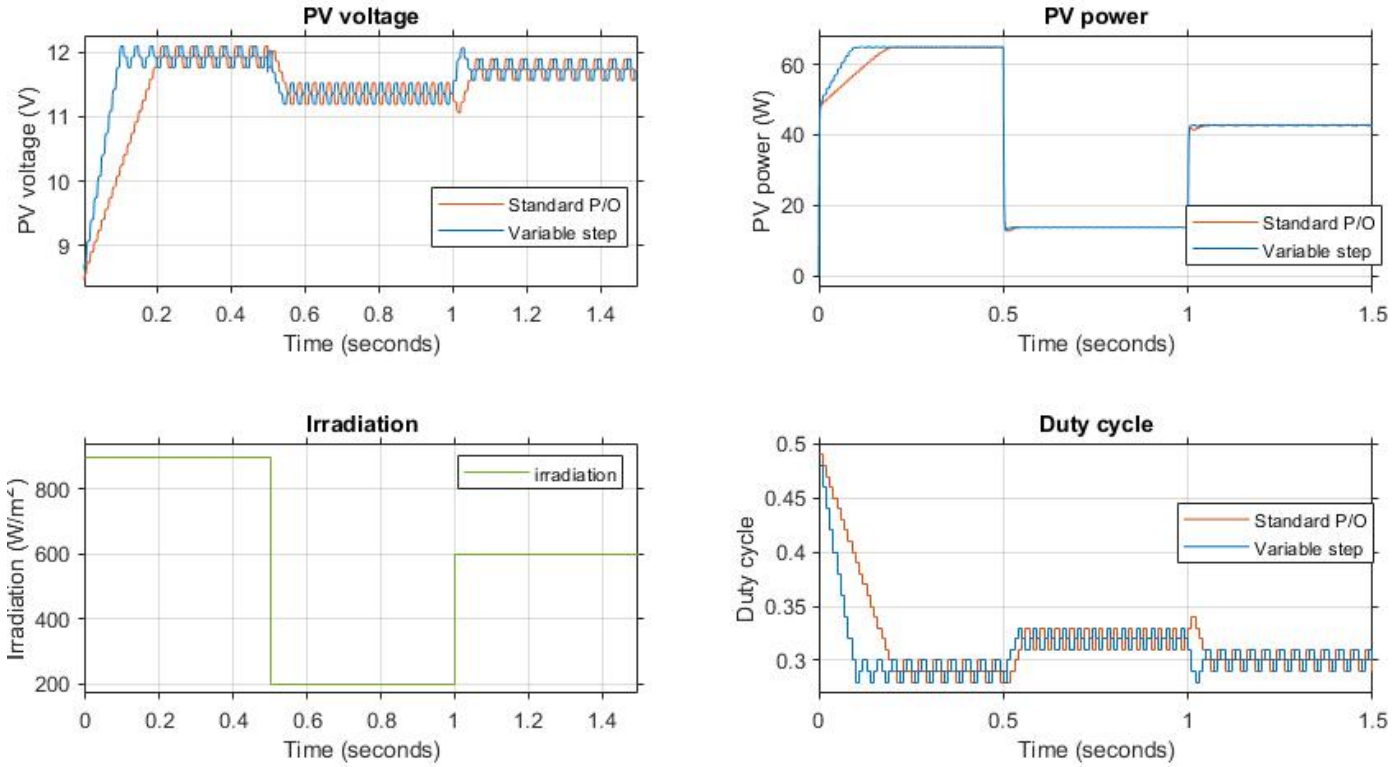


Figure B.4: Performance of standard P/O and variable step under different solar irradiances

B.0.5. Influence of system voltage on duty cycle

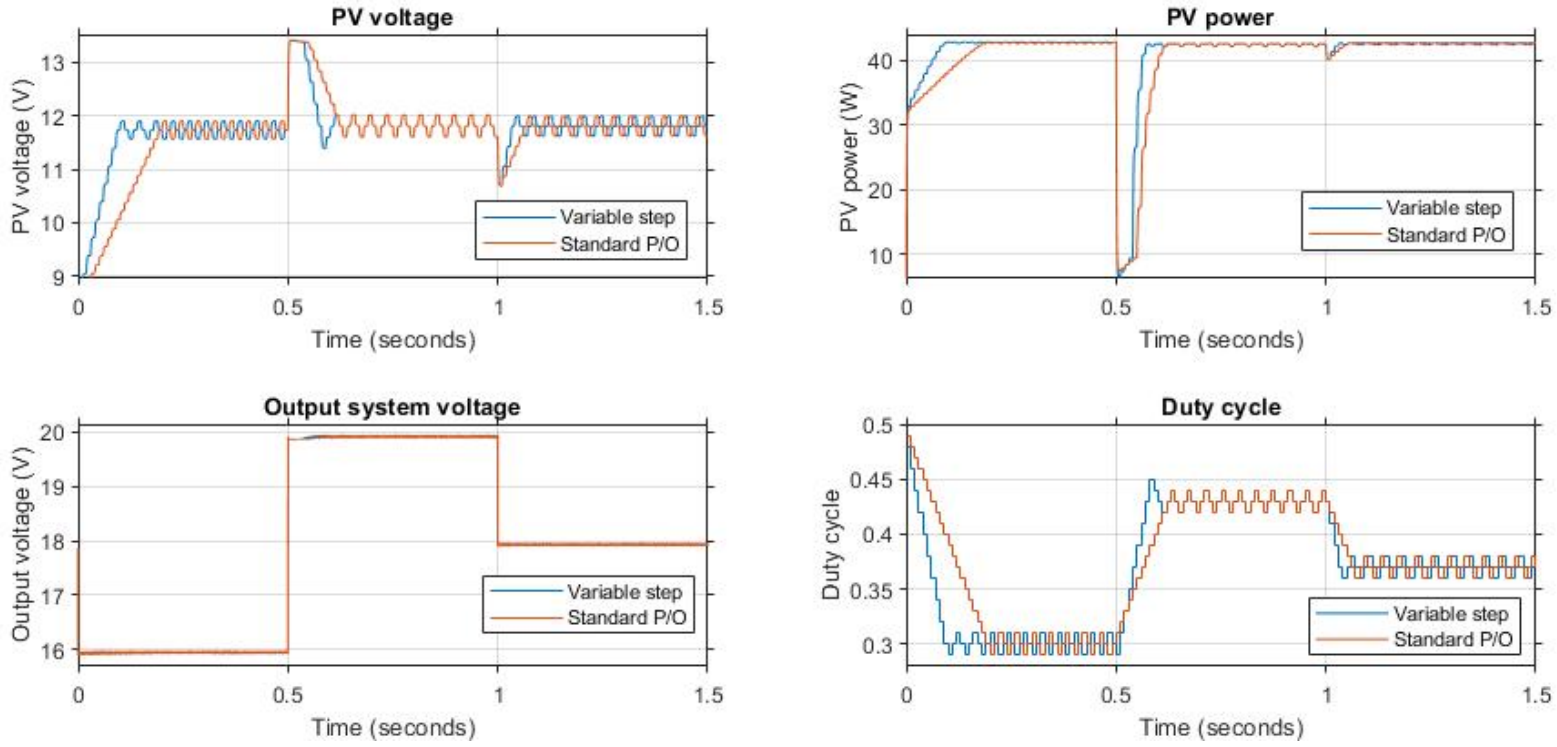


Figure B.5: Algorithm at different system voltages

C

Code

C.1. Gitlab MPPT model

For all the MPPT code, the model and the results can be found on gitlab using the following link or qr code.

https://gitlab.ewi.tudelft.nl/lmuntenaar/solar-powered-drones

C.2. Albedo calculations to compare visible and wide band albedo

```
% Calculate the albedo values based on the .

% Snow
% https://speclib.jpl.nasa.gov/ecospeclibdata/water.snow.mediumgranular.medium.all.medgran_snw_.jhu.becknic.spectrum.txt
% rye grass
% https://speclib.jpl.nasa.gov/ecospeclibdata/vegetation.grass.unknown.unknown.all.grass.jhu.becknic.spectrum.txt
% bare red brick
% https://speclib.jpl.nasa.gov/ecospeclibdata/manmade.generalconstructionmaterial.brick.solid.all.0413uuubrk.jhu.becknic.spectrum.txt
% red oak
% https://speclib.jpl.nasa.gov/ecospeclibdata/vegetation.tree.quercus.rubra.vswir.quru-3-11.ucsb.asd.spectrum.txt

% Solar spectrum - Am1.5g as SolarSpectrumGlobal and Am1.5d as SolarSpectrumDirect
% https://www2.pvlighthouse.com.au/resources/optics/spectrum%20library/spectrum%20library.aspx

% Determin the diffuse solar irradiation as the global - direct irradiation
SolarSpectrumDiff = SolarSpectrumGlobal;
SolarSpectrumDiff(:, 2) = SolarSpectrumGlobal(:, 2) - SolarSpectrumDirect(:, 2);

startVisibleBand = 400;
endVisibleBand = 680;
VisibleSpectrumIndecies = SolarSpectrumGlobal(:,1) >= startVisibleBand & SolarSpectrumGlobal(:,1) <= endVisibleBand;

% Interpolate to get reflectanes at the wavelengths of the solar spectrum values
GrassReflectivity = SolarSpectrumGlobal;
GrassReflectivity(:,2) = interp1(reflectivityGrass(:,1)*1000, reflectivityGrass(:,2)/100, GrassReflectivity(:,1), 'linear', 'extrap');
SnowReflectivity = SolarSpectrumGlobal;
SnowReflectivity(:,2) = interp1(reflectivitySnow(:,1)*1000, reflectivitySnow(:,2)/100, SnowReflectivity(:,1), 'linear', 'extrap');
BrickReflectivity = SolarSpectrumGlobal;
BrickReflectivity(:,2) = interp1(reflectivityBrick(:,1)*1000, reflectivityBrick(:,2)/100, BrickReflectivity(:,1), 'linear', 'extrap');
OakReflectivity = SolarSpectrumGlobal;
OakReflectivity(:,2) = interp1(reflectivityOak(:,1)*1000, reflectivityOak(:,2)/100, OakReflectivity(:,1), 'linear', 'extrap');

% Calculate the albedo as sum( reflectivity * solarirradiation ) / sum( solarirradiation )
albedoGrass = sum(GrassReflectivity(:,2) .* SolarSpectrumGlobal(:,2)) / sum(SolarSpectrumGlobal(:,2));
albedoGrassVis = sum(GrassReflectivity(VisibleSpectrumIndecies,2) .* SolarSpectrumGlobal(VisibleSpectrumIndecies,2)) / sum(
```

```
    SolarSpectrumGlobal(VisibleSpectrumIndecies,2));
albedoSnow = sum(SnowReflectivity(:,2) .* SolarSpectrumGlobal(:,2)) / sum(
    SolarSpectrumGlobal(:,2));
albedoSnowVis = sum(SnowReflectivity(VisibleSpectrumIndecies,2) .*
    SolarSpectrumGlobal(VisibleSpectrumIndecies,2)) / sum(
    SolarSpectrumGlobal(VisibleSpectrumIndecies,2));
albedoBrick = sum(BrickReflectivity(:,2) .* SolarSpectrumGlobal(:,2)) /
   sum(SolarSpectrumGlobal(:,2));
albedoBrickVis = sum(BrickReflectivity(VisibleSpectrumIndecies,2) .*
    SolarSpectrumGlobal(VisibleSpectrumIndecies,2)) / sum(
    SolarSpectrumGlobal(VisibleSpectrumIndecies,2));
albedoOak = sum(OakReflectivity(:,2) .* SolarSpectrumGlobal(:,2)) / sum(
    SolarSpectrumGlobal(:,2));
albedoOakVis = sum(OakReflectivity(VisibleSpectrumIndecies,2) .*
    SolarSpectrumGlobal(VisibleSpectrumIndecies,2)) / sum(
    SolarSpectrumGlobal(VisibleSpectrumIndecies,2));

albedos = [albedoGrass,  albedoGrassVis;
           albedoSnow,   albedoSnowVis;
           albedoBrick,  albedoBrickVis;
           albedoOak,    albedoOakVis]

% Calculate the albedo of grass in the visible band in diffuse light
    conditions
albedoGrassVisDiffuse = sum(GrassReflectivity(VisibleSpectrumIndecies,2)
    .* SolarSpectrumDiff(VisibleSpectrumIndecies,2)) / sum(
    SolarSpectrumDiff(VisibleSpectrumIndecies,2))

% Calculate what percentage of of the solar irradation in the visible band
visibleContribution = sum(SolarSpectrumGlobal(VisibleSpectrumIndecies,2))
    / sum(SolarSpectrumGlobal(:,2))

% Plot the reflectivities
yyaxis left
hold off
plot(GrassReflectivity(:,1), GrassReflectivity(:,2));
hold on
plot(SnowReflectivity(:,1), SnowReflectivity(:,2));
plot(BrickReflectivity(:,1), BrickReflectivity(:,2));
plot(OakReflectivity(:,1), OakReflectivity(:,2));
ylabel("Reflectivity");
yyaxis right
plot(SolarSpectrumGlobal(:,1), SolarSpectrumGlobal(:,2)/max(
    SolarSpectrumGlobal(:,2)));
ylabel("Spectral irradiance (W/m^2)");
xline(startVisibleBand, "--")
xline(endVisibleBand, "--")
grid on
legend(["Rye Grass", "Snow", "Red brick", "Red Oak", "Solar irradiation",
    "Visible band"])
xlim([300 2500]);
xlabel("Wavelength (nm)");
title("Reflectivities of different materials and solar irradiation")
```

C.3. Vignette modeling

```
from os import listdir
```

```
from os.path import isfile, join
import numpy as np
from PIL import Image
from vignettingModeling import model_vignette

# ------------------------------------------------------------------
# This file is used to load the vignette images, average them
# and run the vignette modeling code with it
#
# Sjoerd Groot, Laura Muntenaar
# Thesis Solar Powered Drones - Control algorithm and Albedo generation
# ------------------------------------------------------------------

# File path to the folder with vignette images
inputPath = "VignettingCalibrated/"
input_images = [f for f in listdir(inputPath) if isfile(join(inputPath, f))]
print(input_images)

# Open all images and add them to create an average vignette
average_vignette = None
for image_name in input_images:
    image = Image.open(inputPath + image_name)
    image_arr = np.array(image, dtype=np.float64)

    if average_vignette is None:
        average_vignette = image_arr
    else:
        average_vignette = image_arr + average_vignette

# Normalize the average_vignette
average_vignette = average_vignette / np.max(average_vignette)

# Create the vignette model
model = model_vignette(average_vignette)

print("Model is created:")
print(model)
```

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# ------------------------------------------------------------------
# This file contains functions to find the model coefficients
# and to recreate the vignette model
#
# Sjoerd Groot, Laura Muntenaar
# Thesis Solar Powered Drones - Control algorithm and Albedo generation
# ------------------------------------------------------------------

def model_vignette(vignette_array):
    """This function finds the model coefficients from the vignette image
       array"""
    n, m = vignette_array.shape

```

```
    print("Creating features")
    polynomial_features = create_polynomial_features(n, m)
    data_vec = vignette_array.reshape(n * m, 1)

    # Use linear regression to find the coefficients
    print("Finding model coefficients")
    model = LinearRegression(fit_intercept=False)
    model.fit(polynomial_features, data_vec)
    y_predicted = model.predict(polynomial_features)

    # print performance metrics
    print(mean_squared_error(data_vec, y_predicted))

    return model.coef_, model.fit_intercept


def create_polynomial_features(n, m):
    """Creates the feature vector with [1, r4, r3, r2, r, y, x]"""

    # x, y and r at avery position in the matrix
    matrix_y = np.arange(n).reshape((n, 1)).repeat(m, 1) - n / 2 + 0.5
    matrix_x = np.arange(m).reshape((1, m)).repeat(n, 0) - m / 2 + 0.5
    matrix_r = np.sqrt(matrix_x ** 2 + matrix_y ** 2)  # Euclidean
        distance from the center

    # Make a linear vector of the positional data
    y_vec = matrix_y.reshape(n * m, 1)
    x_vec = matrix_x.reshape(n * m, 1)
    r_vec = matrix_r.reshape(n * m, 1)

    # Normalize the features
    y_vec = y_vec / np.max(y_vec)
    x_vec = x_vec / np.max(y_vec)
    r_vec = r_vec / np.max(r_vec)

    r_square_vec = r_vec ** 2
    r_third_vec = r_vec ** 3
    r_forth_vec = r_vec ** 4
    r_fifth_vec = r_vec ** 5
    r_six_vec = r_vec ** 6

    unity = np.ones((n * m, 1))

    # Concatenate the features
    return np.concatenate((unity, r_forth_vec, r_third_vec, r_square_vec,
        r_vec, y_vec, x_vec), axis=1)


def create_vignetting(model_coefficients, polynomial_features):
    """Recreate the vignette model from the model coefficients and
        polynomial features"""
    return polynomial_features @ model_coefficients.T
```

C.4. Albedo generation

```
from image_loading import createAlbedo
from os import listdir
```

```
from os.path import isfile, join
import numpy as np


# ------------------------------------------------------------------
# This file coverts a folder of images and stores the
# resulting albedo images in a different folder
#
# Sjoerd Groot, Laura Muntenaar
# Thesis Solar Powered Drones - Control algorithm and Albedo generation
# ------------------------------------------------------------------


# Folder with input images
input_path = "sample_raw_images/large_scale_garden"
input_images = [f for f in listdir(input_path) if isfile(join(input_path, f))]

# Folder for output images
output_path = "./result/"

reflectivity = float(input("Target reflectivity"))

vignette_model = np.array([[1, -2.60e+00, 6.07e+00, -4.62e+00, 3.80e-01,
                            2.15e-02, -9.72e-07]])

for i in input_images:
    input_image_path = join(input_path, i)
    output_path = join(output_path, i[:-3]) + "png"
    createAlbedo(input_image_path, output_path, vignette_model=vignette_model, debug=True,
                 reflectivity=reflectivity, showResult=True)
```

```
import rawpy
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import seaborn as sns
import cv2
from vignette.vignettingModeling import create_polynomial_features, create_vignetting


# ------------------------------------------------------------------
# This file contains functions for loading a raw image
# and converting it to an albedo image
#
# Sjoerd Groot, Laura Muntenaar
# Thesis Solar Powered Drones - Control algorithm and Albedo generation
# ------------------------------------------------------------------


class ImageDisplay:
    """
    This class is used to track user selections on an image
```

```
22      Adapted from https://matplotlib.org/3.1.1/users/event_handling.html
23      """
24
25      def __init__(self, imPlotAxis, imageMatrix=None):
26          """
27          imPlotAxis is the axis containing the image
28          imageMatrix may be the linear image matrix, if given
29              the average value of the user selection in image matrix will
                    be calculated
30          """
31          self.imPlot = imPlotAxis
32          self.rect = None
33          self.press = None
34          self.imageMatrix = imageMatrix
35
36      def connect(self):
37          """connect to all the events we need"""
38          self.cidpress = self.imPlot.figure.canvas.mpl_connect('
                button_press_event', self.on_press)
39          self.cidrelease = self.imPlot.figure.canvas.mpl_connect('
                button_release_event', self.on_release)
40          self.cidmotion = self.imPlot.figure.canvas.mpl_connect('
                motion_notify_event', self.on_motion)
41
42      def on_press(self, event):
43          """on button press we will see if the mouse is over us and store
                some data"""
44          if event.inaxes != self.imPlot.axes:
45              return
46
47          contains, attrd = self.imPlot.contains(event)
48          if not contains:
49              return
50
51          self.press = event.xdata, event.ydata
52          self.rect = Rectangle((event.xdata, event.ydata), 40, 30,
                linewidth=1, edgecolor='r', facecolor='none')
53
54          self.imPlot.add_patch(self.rect)
55          self.imPlot.figure.canvas.draw()
56
57      def on_motion(self, event):
58          """Update the user selection on motion"""
59          if self.press is None: return
60          if event.inaxes != self.imPlot.axes: return
61
62          xpress, ypress = self.press
63          dx = event.xdata - xpress
64          dy = event.ydata - ypress
65
66          self.rect.set_width(dx)
67          self.rect.set_height(dy)
68          self.imPlot.figure.canvas.draw()
69
70      def on_release(self, event):
71          """reset the press data on release"""
```

```
72          self.press = None
73          self.imPlot.figure.canvas.draw()
74
75          if self.imageMatrix is not None:
76              x1, y1, x2, y2 = self.getSelection()
77
78              section = self.imageMatrix[y1:y2, x1:x2]
79              m = np.mean(section)
80              print(m)
81
82      def getSelection(self):
83          """Get the current user selection"""
84          if not self.rect:
85              return None, None, None, None
86
87          x1 = self.rect.get_x()
88          y1 = self.rect.get_y()
89          x2 = x1 + self.rect.get_width()
90          y2 = y1 + self.rect.get_height()
91          x1, x2 = min(x1, x2), max(x1, x2)
92          y1, y2 = min(y1, y2), max(y1, y2)
93
94          return int(x1), int(y1), int(x2), int(y2)
95
96      def disconnect(self):
97          """Disconnect all the stored connection ids"""
98          self.imPlot.figure.canvas.mpl_disconnect(self.cidpress)
99          self.imPlot.figure.canvas.mpl_disconnect(self.cidrelease)
100         self.imPlot.figure.canvas.mpl_disconnect(self.cidmotion)
101
102
103 def makeHistogram(rgb_image):
104     """Creates a distribution plot"""
105
106     datacount = rgb_image.shape[0] * rgb_image.shape[1]
107     print(f"{datacount} data points in the reference target")
108     pixels = rgb_image.reshape((datacount, 3))
109
110     if (datacount > 100000):
111         print("limiting histogram visualisation to 100.000 data points")
112         selection = np.random.choice(datacount, 100000)
113         sns.kdeplot(pixels[selection, 0], color='red', shade=True)
114         sns.kdeplot(pixels[selection, 1], color='green', shade=True)
115         sns.kdeplot(pixels[selection, 2], color='blue', shade=True)
116     else:
117         sns.kdeplot(pixels[:, 0], color='red', shade=True)
118         sns.kdeplot(pixels[:, 1], color='green', shade=True)
119         sns.kdeplot(pixels[:, 2], color='blue', shade=True)
120
121
122 def createAlbedo(path, storePath, vignette_model=None, debug=False,
        reflectivity=None, showResult=False):
123     """
124     Creates the albedo of a single image from path and stores the result
          in storePath
125     """
```

```
126
127     # Open the image with rawpy
128     with rawpy.imread(path) as raw:
129         if debug:
130             print(raw.raw_image)
131             x, y = raw.raw_image.shape
132             color_raw = np.zeros((x, y, 3))
133             color_raw[::2, ::2,  1] = raw.raw_image[::2, ::2]/1023
134             color_raw[1::2, 1::2, 1] = raw.raw_image[1::2, 1::2]/1023
135             color_raw[1::2, ::2,  0] = raw.raw_image[1::2, ::2]/1023
136             color_raw[::2,  1::2, 2] = raw.raw_image[::2, 1::2]/1023
137             print(color_raw)
138             plt.imshow(color_raw)
139             plt.show()
140
141         # Demosaicing the image with a standard post processing
142         print("Demosaicing regular image")
143         demosaiced_img = raw.postprocess()
144
145         # Demosaicing the image with linear response and 16 color depth
146         # Gamma is 1,1 to keep a linear relationship
147         print("Demosaicing linear 16bit image")
148         demosaiced_img_linear = raw.postprocess(gamma=(1, 1),
                no_auto_bright=True, output_bps=16) / 2**16
149
150         # Compensate the vignette
151         if vignette_model is not None:
152             n, m, _ = demosaiced_img_linear.shape
153             image_features = create_polynomial_features(n, m)
154             vignette = create_vignetting(vignette_model, image_features).
                    reshape(n, m, 1)
155             demosaiced_img_linear = demosaiced_img_linear / vignette
156
157         if debug:
158             plt.imshow(demosaiced_img_linear)
159
160         # Plot the image and let the user select the reference target
                region
161         print("Plotting image for annotation")
162         plt.figure()
163         axis = plt.gca()
164         imPlot = axis.imshow(demosaiced_img)
165         plt.title("Please annotate the reference target\n Once done, you
                can close the figure to continue")
166
167         # Make an image display object that will track the user input
168         im = ImageDisplay(axis)
169         im.connect()
170
171         if debug:
172             plt.figure()
173             makeHistogram(demosaiced_img_linear)
174
175         # Show the image and stall until the annotation is done
176         plt.show()
177         im.disconnect()
```

```
178
179         # get the selected target pixels
180         x1, y1, x2, y2 = im.getSelection()
181         print(x1, y1, x2, y2)
182         target_pixels_linear = demosaiced_img_linear[y1:y2, x1:x2]
183
184         if debug:
185             # Display user annotation
186             print("Plotting user annotation")
187             target_pixels_gamma = demosaiced_img[y1:y2, x1:x2]
188             plt.imshow(target_pixels_gamma)
189             plt.title("Selected target")
190
191             # Display histogram Target histogram
192             print("Plotting annotation histogram")
193             plt.figure()
194             makeHistogram(target_pixels_linear)
195             plt.title("Corresponding histogram")
196
197         # Calculate mean target brightness values
198         average_target_linear = np.mean(target_pixels_linear, axis=(0, 1))
199         print(f'The average values of the reference target were: r {
                average_target_linear[0]}, g {average_target_linear[1]}, b {
                average_target_linear[2]}')
200         if max(average_target_linear) > 0.98:
201             print("Warning the reference target might be overexposed")
202
203         if debug:
204             plt.axvline(x=average_target_linear[0], c="red")
205             plt.axvline(x=average_target_linear[1], c="green")
206             plt.axvline(x=average_target_linear[2], c="blue")
207             plt.show()
208
209         # Average the red green and blue bands
210         mean_target_linear_value = np.mean(average_target_linear)
211         demosaiced_img_linear = np.mean(demosaiced_img_linear, axis=2)
212
213         # Ask for target reflectivity
214         if reflectivity is None:
215             reflectivity = float(input("What is the target reflectivity?")
                    )
216
217             # Make sure the value is not between 0-100 but between 0-1
218             if reflectivity > 1:
219                 reflectivity /= 100
220
221         # Calibrate the image
222         albedo = reflectivity * demosaiced_img_linear /
                mean_target_linear_value
223
224         if debug or showResult:
225             plt.figure()
226             plt.imshow(albedo)
227             plt.colorbar()
228             plt.title("Calculated visible band albedo")
229             resultDisplay = ImageDisplay(plt.gca(), albedo)
```

```
230             resultDisplay.connect()
231             plt.show()
232             resultDisplay.disconnect()
233 
234         cv2.imwrite(storePath, albedo * 255)
235         return True
```