

Benjamin de Capitani

CamZilla: A GigaPixel Panorama Robot in the Mountains



Semester Thesis SA-2012-HS
July 2012 to September 2012

Tutor: Matthias Keller
Co-Tutor: Jan Beutel
Supervisor: Prof. Lothar Thiele

Abstract

The creation of large scale panorama images with gigapixel resolution is a very time consuming task. Especially when the camera setup is located at a remote site and relies solely on solar panels for power. Any movement and picture capturing consumes energy and therefore one wants to reduce such actions to a minimum. Also, with every capturing the network is loaded with huge amounts of data, in the order of tens of gigabytes. In a case where the view is obstructed by fog, the capturing would consume power and load the network without a satisfactory result. In this thesis, we present an end-to-end solution for capturing gigapixel panorama images autonomously without the need of any human interaction. By first evaluating the sight conditions with the help of a preview image provided by a low resolution webcam, picture capturing in bad conditions is avoided and thus, power and network bandwidth are used more efficiently.

Acknowledgments

First, I would like to thank Prof. Dr. Lothar Thiele for giving me the opportunity to write this semester thesis in his research group.

Also, I would like to thank my advisor Matthias Keller for his support throughout the whole project. His broad knowledge and dedication were very helpful to realize this work.

Special thanks goes to all the TIK staff for their open and very friendly nature.

And last but not least, I would like to thank all the students who were present in the ETZ G69 room during my work for their moral support.

Contents

1	Introduction	7
1.1	Motivation	7
1.2	Environment	7
1.2.1	PermaSense	7
1.2.2	PermaSense Technology	8
1.2.3	Code Environment	8
1.2.4	Gigapixel Images	8
1.2.5	Gigapixel Images within PermaSense	8
1.3	Goal of this Thesis	8
1.3.1	Sight Evaluation	9
1.3.2	Control Loop	9
2	Related Work	11
3	Materials and Methods	13
3.1	CamZilla Robot	13
3.2	Remote Control	14
4	End-To-End Solution	15
4.1	Sight Evaluation	15
4.1.1	Entropy	15
4.1.2	Sharpness	16
4.1.3	Average Intensity of Image	16
4.2	Aligned Image Series	16
4.3	Panorama Stitching	17
4.4	Enhancement	17
4.4.1	Contrast stretching	17
4.5	Publishing	18
5	Experiments and Results	19
5.1	Sight Evaluation	19
5.1.1	Calibration	19
5.2	Example Panoramas	21
6	Conclusion and Outlook	23
6.1	Conclusion	23
6.2	Outlook	23
A	Panorama Scripting	25
B	Example Exif	27
C	CamZilla Schedule	29
D	Workflow	31
E	Original Problem	33

List of Figures

1.1	Panorama stitching from masaic of images.	8
1.2	Schematic of the developed control loop.	10
3.1	Schematic architecture of the CamZilla robot head.	13
3.2	Communication flow between the control Server (GSN) and the CamZilla Robot.	14
4.1	Raw image and its corresponding pixel value histogram.	17
4.2	Contrast stretched image and its corresponding pixel value histogram	18
4.3	Raw image (a), simple RGB stretched with color shift (b) and enhanced image (c) from the Dirruhorn deployment of PermaSense.	18
5.1	Luminosity factor of daytime images (blue and red) compared to nighttime images (green).	19
5.2	Histograms of entropy and sharpness measures on good (blue) and bad (red) sight images.	20
5.3	Evaluation Process.	20
5.4	Example images with evaluation results. Values below the threshold are red, values above green.	21
5.5	Example panorama image with a resolution of 0.38 gigapixels.	21
D.1	Process flow of the CamZilla robot.	32

Chapter 1

Introduction

This thesis describes an end-to-end solution for capturing high resolution (gigapixel) images with the help of a Camera mounted on a remotely controllable tripod. The final product is integrated into an existing sensor network for monitoring permafrost in the Swiss Alps (PermaSense). The process should work with as little human interaction as possible. The application for such high resolution images is to monitor changes in landscape visually.

1.1 Motivation

The process for creating gigapixel images within the PermaSense project to this date has been very time consuming. First a preview image was captured to visually assess the sight conditions. Afterwards, one could trigger the panorama job and finally stitch the raw images manually to one gigapixel image. Therefore, the goal is to create an automated end-to-end solution for the whole process. This will allow for gigapixel images to be captured automatically at regular intervals, while keeping human interaction at a minimum.

1.2 Environment

The following sections introduce the environment in which the solution is integrated as well as some terminology.

1.2.1 PermaSense

From permasense.ch¹:

“PermaSense is a consortium of researchers and research projects bringing together different engineering and environmental research disciplines from several Swiss research institutions and companies. We develop, deploy and operate wireless sensing systems customized for long-term autonomous operation in high-mountain environments. Around this central element we develop concepts, methods and tools to investigate and to quantify the connection between climate, cryosphere (permafrost, glaciers, snow) and geomorphodynamics. Both the better understanding and the reliable observation of phenomena such as slope instability are of practical relevance and motivate close collaboration with public authorities. The long-term collaboration in this consortium develops solid interdisciplinary know-how, experience and networks in the participating institutions as well as in their national and international partners.”

¹30.09.2012, <http://www.permasense.ch/home/about.html>

1.2.2 PermaSense Technology

The architecture of the network consists of remote CoreStations [1] containing a variety of sensors for environment monitoring and a powerful server (GSN [3]) in the backend. The GSN collects all data from the sensors and acts as the central command center for the PermaSense network.

1.2.3 Code Environment

Since this project is part of the PermaSense project, all source code is embedded in the framework provided. Algorithms running on the CoreStation are written in Python. On the server side, the implementation is done in Java. For performance reasons all image analysis and processing was written in C++ with the help of the OpenCV² library.

1.2.4 Gigapixel Images

A digital image containing at least one billion (10^9) pixels is referred to as a gigapixel image. As a comparison, state-of-the-art DSLRs can take pictures containing around 10^7 pixels. One technique for creating such large images is taking an image series (mosaic) with a high resolution DSLR and stitching them together to one large panorama (see Figure 1.1). This is also the method employed in this project. Another method requires scanning a film negative with a high-end scanner with at least 3000 dpi. However this method does not allow for automated capturing, since a physical object (the film negative) has to be obtained from the camera. Apart from artistic applications gigapixel images also provide great opportunities to monitor a view of interest with high detail which is also the application relevant for this thesis.

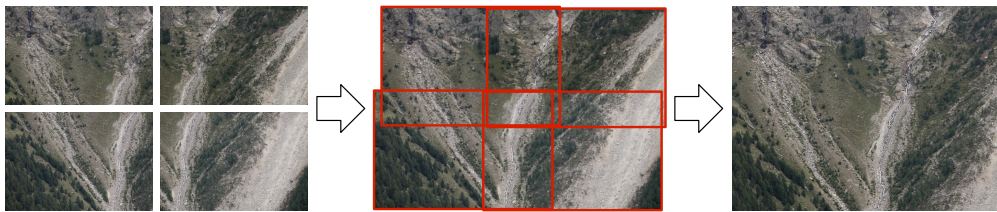


Figure 1.1: Panorama stitching from masaic of images.

1.2.5 Gigapixel Images within PermaSense

The method for creating gigapixel images discussed in this thesis is to capture an aligned array of high resolution (megapixel) images and stitching them together to one large gigapixel image. The mounting system enables an operator (or an algorithm) to control the camera's view on two axes, thus a possibility to capture aligned images over a large field of view is provided. The CamZilla setup incorporates the cameras, actuators and a CoreStation. With the help of gigapixel images, changes in the environment can be monitored with high detail. For the PermaSense project it is of interest to monitor glacier or rock displacements [4][2]. Daily images from the same site can be compared side by side to track changes.

1.3 Goal of this Thesis

The goal of this thesis is to create an end-to-end solution for automated capturing of gigapixel images with the help of the CamZilla robot and integrate the whole process into the existing PermaSense network. The process should run autonomously without the need of human interaction. The task can be split into two parts:

²OpenCV (**Open** Source **C**omputer **V**ision) is a library of programming functions for real time computer vision. <http://opencv.willowgarage.com>

1.3.1 Sight Evaluation

To avoid taking pictures while the view is obstructed by fog, heavy snowfall or similar, the sight conditions are evaluated prior to capturing the image series. This is done by analyzing an image taken by the webcam. The webcam provides an efficient possibility to assess sight conditions. Since it has a wide angle lens, only few evaluation images are sufficient to determine if the view is free from obstructions. Multiple possible measurements were considered and implemented (see Chapter 4.1).

1.3.2 Control Loop

The control loop covers the whole panorama capturing process. A visual workflow is given in Figure 1.2. The control implementation was adapted from a recent project that automatically generates job schedules based on real-time sensor readings [12].

Panorama Image Series

Whenever the sight evaluation was positive (Chapter 1.3.1) the aligned images series will be captured. The basic functionality for creating a mosaic of images is already provided by the framework. The contribution in this thesis is to build a new schedule for such a task automatically and uploading it to the CoreStation.

Stitching Gigapixel Images

For stitching the images to one large panorama, we rely on the existing open source panorama stitcher Hugin³. Hugin provides a wide array of prebuilt functions and can stitch the provided images with no further input than the images themselves. It reads the camera variables directly from the Exif⁴ of the images and thus all properties (distortions from lens, vignetting, color correction, etc.) are extracted. To further improve the result an automated color correction is run. For efficient viewing, a browser plugin is suggested. Some example panoramas were prepared for the HD view SL⁵ plugin. However, this step is not yet integrated into the PermaSense framework since there is no consensus on a designated viewer yet.

Workflow

Figure 1.2 describes the whole workflow as it was developed for this thesis. Since the GSN and the panorama creation algorithm are completely independent, they could be run on two separate servers. By completely separating the two processes, one can assure, that the GSN is not affected by the heavy workload while stitching hundreds of images. The panorama creator is mostly passive, it queries the GSN at regular intervals if a new image series is available for stitching. Since the stitcher does not have a direct connection to the CoreStation, this connection is dashed. The delay after a failed evaluation is typically 1 hour, small enough to allow for a new evaluation later the same day. After a successful panorama series was captured, the delay is usually 1 day. The possibility to shorten or lengthen these delays exists, if desired.

³Hugin: a cross-platform, open source panorama stitcher

⁴Exchangeable image file format is a format for storing metadata containing information about camera, lens type, GPS location e.t.c. in image files.

⁵High resolution image viewer plugin, Microsoft.

<http://research.microsoft.com/en-us/um/redmond/groups/ivm/hdviews/>

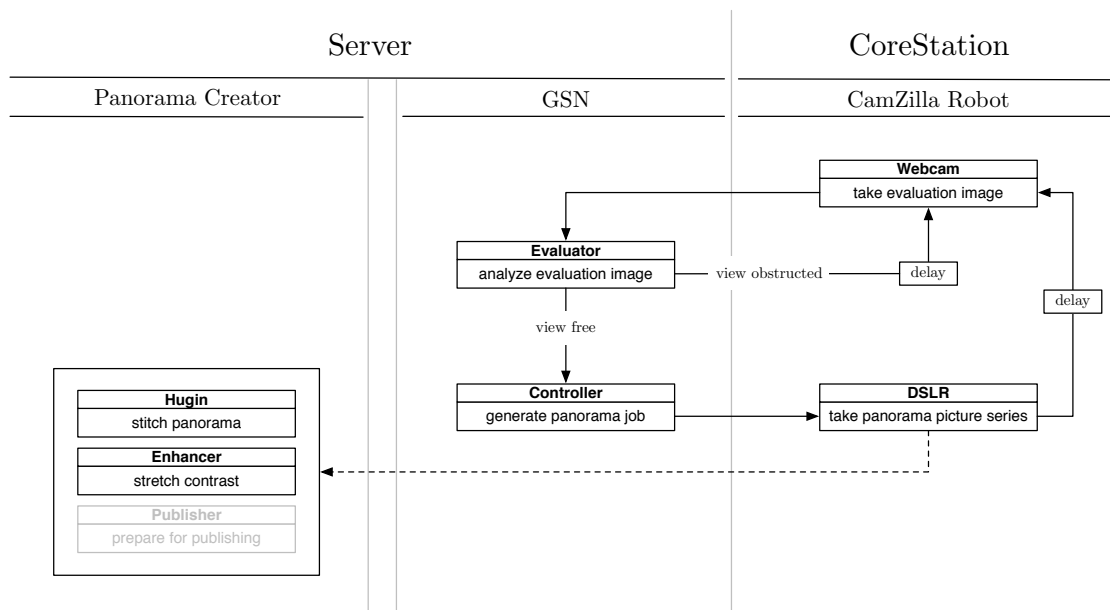


Figure 1.2: Schematic of the developed control loop.

Chapter 2

Related Work

Since this project covers multiple fields there is no single resource available for reference.

Remote cameras are employed in multiple fields of research. These cameras are typically triggered via hand triggers, a timer built into the camera, or radio transmitters. For some applications they are also triggered by an event in their surrounding. For example, loud sounds can be the trigger to capture rocket launches¹ or movement can trigger wildlife monitoring cameras, called game cameras². However, these cameras are typically not connected to a network and therefore the images have to be transmitted manually to a computer. Also, they are typically deployed for a limited time only. A former PermaSense project, MountainView[4], describes a universal DSLR mount with network access, which allows for remote capturing and viewing of high quality images. MountainView provides the basic setup for remotely capturing images in the PermaSense network.

Several existing gigapixel capturing setups exist, most of them are commercial. Gigapan³ provides a complete setup with a camera mount, a stitcher and a browser based viewer. Another complete solution is Panogear⁴. Both setups described above cost approximately 500\$ for the basic unit. However all currently available setups rely on a human being present and initializing the capturing. The sight evaluation in these setups is also the job of the human operator. Since the CamZilla robot is at a remote site, this would not be suitable. For an autonomous system all these tasks are to be automated.

A suggestion for sight evaluation is given in [10]. They state, that only approximately 8% of bad images are falsely classified. The method applied has already been successfully tested in the PermaSense project and was therefor the base for our evaluation algorithm. Other, more complex fog detection algorithms primarily come from the automobile industry [11][9][7]. However, they all rely on a very specific field of view (e.g. part of the image is a road, vanishing point detectable). Since our system should work in an arbitrary environment and not depend on a specific view, these methods are not be feasible.

¹<http://www.sportsshooter.com/news/1429>

²Bushnell: www.bushnell.com, Cuddeback: <http://cuddeback.com>

³<http://www.gigapan.com>

⁴<http://www.kolor.com>

Chapter 3

Materials and Methods

3.1 CamZilla Robot

CamZilla is the name for a node within the PermaSense network consisting of a CoreStation and the aforementioned controllable camera tripod. In addition to the high resolution DSLR (Digital single-lens reflex) camera, it also includes a low resolution, wide angle webcam. The webcam is mounted on top of the DSLR, such that any movement from the tripod results in movements of both cameras (see Figure 3.1). Ideally, the positioning axes intersect at the Panoramic Pivot Point¹ of the DSLR camera, to provide optimally aligned images. However, the axes from the CamZilla robot have not been aligned to this point yet. The resulting panoramas are hardly affected by this, since the offset is small enough.

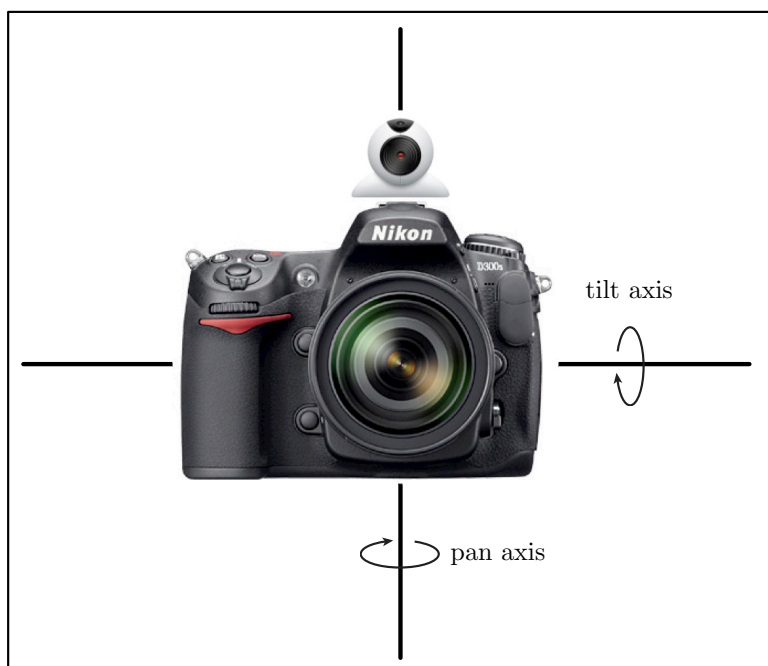


Figure 3.1: Schematic architecture of the CamZilla robot head.

The DSLR Camera in this setup is a Nikon D300s with a 12 megapixel optical sensor. For the image series at maximum zoom the field of view is limited to 4 degrees. The webcam utilized is a Mobotix Allround M24 with a 90 degree field of view. The optical sensor provides 3 megapixels, however, the typical resolution utilized in our project is 768×576 pixels (0.4 megapixel). A raw image from the DSLR contains roughly 7 MB of data, one from the webcam only around 100 KB. Since the webcam's resolution is significantly lower, taking an image costs significantly less

¹Panoramic Pivot Point: The rotational point at which the alignment of all objects seen through the lens stays the same, even as the lens and camera is rotated.[5]

power and network bandwidth. This is especially a crucial factor in a remote location where the only power source is solar panels.

3.2 Remote Control

The CamZilla robot can be controlled through the PermaSense web interface or by uploading a suitable schedule (see section 4.2). All communication to the robot comes from the GSN and is analyzed on the CoreStation by an embedded computer (Gumstix²). The Positional controls are passed through an Arduino³ actuator controller. The capturing task is sent to the DSLR with the help of gPhoto⁴. The webcam image can be accessed via HTTP. The exact architecture is described in more detail in [4].

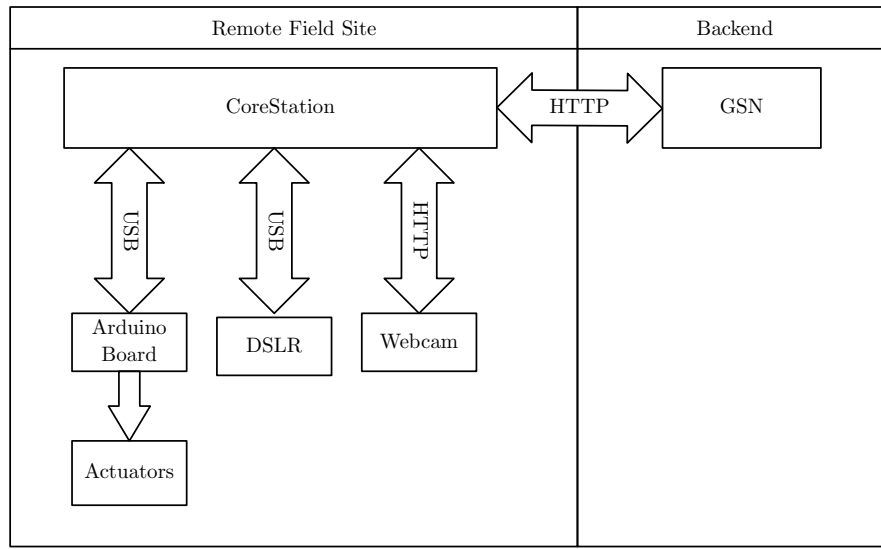


Figure 3.2: Communication flow between the control Server (GSN) and the CamZilla Robot.

The CamZilla parameters can be configured and picture taking tasks can be initiated remotely from the GSN. The most relevant configurable parameters are:

- Starting position x,y.
- Pictures in x and y direction.
- Rotation around pan and tilt axis between each image, this defines the overlap of two adjacent images.
- Image Quality
- Image size
- Aperture
- Shutter speed
- ISO
- Manual focus value or auto focus

Apart from these variables, virtually every camera parameter can be configured. The full set of configurations provide a very flexible system, even capable of shooting images at night with appropriate values.

²Gumstix: A low-power, embedded PC running linux. <http://www.gumstix.com/>

³Arduino Board: An open-source electronics prototyping platform. <http://www.arduino.cc/>

⁴gPhoto: an open source library designed to fully control (configure, capture and grab captured images) digital cameras via the USB port. <http://gphoto.sourceforge.net/>

Chapter 4

End-To-End Solution

This chapter describes the complete solution from sight evaluation to panorama stitching designed for this thesis.

4.1 Sight Evaluation

The remote CoreStation relies solely on solar panels for its power, therefore we want to avoid unnecessary movements or picture capturing of the CamZilla robot. One important way to save power, is to only trigger an image capturing sequence with the DSLR if the view is free from unwanted obstructions, such as fog, low clouds or snow on the lens. Since the whole system should be autonomous, we've developed a sight evaluation algorithm which assesses the view conditions from an evaluation image taken by the low resolution webcam. Taking the image from the webcam and not the DSLR camera yields several advantages. The webcam needs less power to capture one frame and the resulting images require less memory, which in return leads to less network traffic and power consumption. Furthermore, the lens of the webcam is wide angle and can overview a large portion of the landscape of interest, thus only few evaluation images are needed to make a decision. The evaluation process is run on the server after an evaluation image was grabbed from the webcam. The evaluation could be done directly on the CoreStation in the future, however, to remain flexible and experiment with various algorithms, it is run on the GSN at this stage. To evaluate the preview images, we introduce the following measurements.

4.1.1 Entropy

The entropy of an image is a statistical measure of randomness. The less random an image is, the lower its entropy will be. In our image analysis the entropy is calculated with the help of the grayscale representation of the image. We calculate the entropy with the help of the probability P_i , that the difference between two adjacent pixels is i [6]. The entropy is then calculated as:

$$H = - \sum_i (P_i * \log_2(P_i)) \quad (4.1)$$

This is the same formula used by the Galileo Imaging Team¹. A landscape with free view will have a high entropy in comparison with obstruction by some uniform field (fog). One drawback of this measure, is that it is prone to noise: the entropy rises as noise increases since then the difference between two adjacent pixels is more random. For images with high noise, the entropy will also be high.

¹Galileo: a NASA spacecraft mission launched in 1989. <http://solarsystem.nasa.gov/galileo/>

4.1.2 Sharpness

This evaluation gives a measure for sharpness of an image. The idea is derived from [10]. It is based on a difference of gaussians² to obtain a mask of high frequency (edge) neighborhoods. All calculations for this measure are performed per element (pixel) on a grayscale version of the image. Let h_i be the original image and h_b the gaussian blurred version with appropriate σ . The mask h_d is calculated as follows:

$$h_d = h_b - h_i. \quad (4.2)$$

For images obstructed by fog, the blurred and original images will be almost identical and thus the difference will be near zero.

Next, let P be the set of pixels in the image, the sharpness factor s is then defined as:

$$s = \frac{\sum_{p \in P} h_d(p)^2}{\sum_{p \in P} h_b(p)^2}. \quad (4.3)$$

For completeness, and since [10] does not supply the exact implementation, a pseudocode of our algorithm is given below (Algorithm 1). This sharpness factor s provides a way to measure the relative amount of high frequencies in an image. In our experiments, we found that an appropriate σ for the low pass filter is 1.5.

Algorithm 1 Sharpness factor calculation

```

function SHARPNESS(image  $I$ )
  convert  $I$  to grayscale
  apply a low pass filter (Gaussian Blur) to  $I$  to obtain  $I_{low}$ 
  compute high frequency content  $I_{high} = I - I_{low}$ 
  sumHigh = 0, sumLow = 0
  for each pixel position  $x,y$  do
    sumHigh +=  $I_{high}(x,y)^2$ 
    sumLow +=  $I_{low}(x,y)^2$ 
  end for
  sharpness = sumHigh/sumLow
end function

```

4.1.3 Average Intensity of Image

If the average intensity of an image falls below a predefined threshold the picture is discarded. The assumption is that it is then either a night shot with insufficient exposure or heavily obstructed. To measure the intensity of every pixel, we simply take the grayscale value at each position.

4.2 Aligned Image Series

If the evaluation yields a positive result, a panorama job is triggered. The existing PermaSense Framework provides ready-built functions for capturing aligned image series. Such a job contains a starting time, starting coordinates $[x,y]$, number of images in x and y direction, offset between neighboring images as well as several camera parameters. The job schedule is described in a cronjob like format which allows for exact timing of events. For a detailed example of such a schedule please refer to Appendix C.

²Difference of Gaussians (DOG): A grayscale image enhancement algorithm to increase visibility of edges and other details.

4.3 Panorama Stitching

Once the CamZilla robot is done with one image series, all images are transmitted to the server. As soon as all images are available on the server, the stitching task is started. Since this task is computationally very complex, its process is completely independent from the GSN. This ensures that the GSN does not lock during the hours (or even days) when the stitching is in progress. Hugin was used to stitch the raw images to one panorama. Hugin is an open source panorama stitcher which provides very simple yet effective functionalities to stitch together a mosaic of overlapping images. It also provides a set of command line tools for panorama scripting, which was of special interest for this thesis. Since the exact camera and lens parameters are automatically extracted from the Exif of the images, Hugin can also compensate distortions and vignetting. These properties simplify the stitching task to a great degree. An example of the used panorama script is given in Appendix A.

4.4 Enhancement

Since the images taken by the DSLR are essentially taken “blind” (e.g. no visual feedback by a user) some series might have a very low color intensity or contrast. For this reason, a final color and contrast correction is applied. The method behaves much like Adobe Photoshop’s “Auto-Levels” function and is based on the contrast stretching algorithm described in [8].

4.4.1 Contrast stretching

Contrast stretching is a method to compensate low dynamic range in an image. If the values for one channel of an image do not cover the whole range, then this channel is stretched to fill the complete range of values. Consider Figure 4.1 below with a very low contrast raw image. The pixel value histogram indicates, that only approximately 20% of the possible values covered.

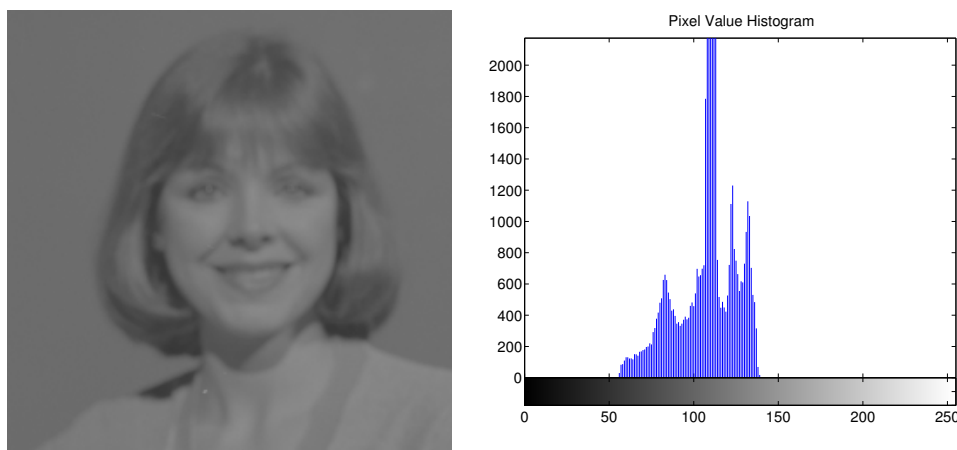


Figure 4.1: Raw image and its corresponding pixel value histogram³

By applying the contrast stretch algorithm, the pixel values spread across the whole range and the resulting image looks much more appealing (Fig. 4.2).

³We thank Georgia Institute of Technology for the use of this image acquired from their online image database

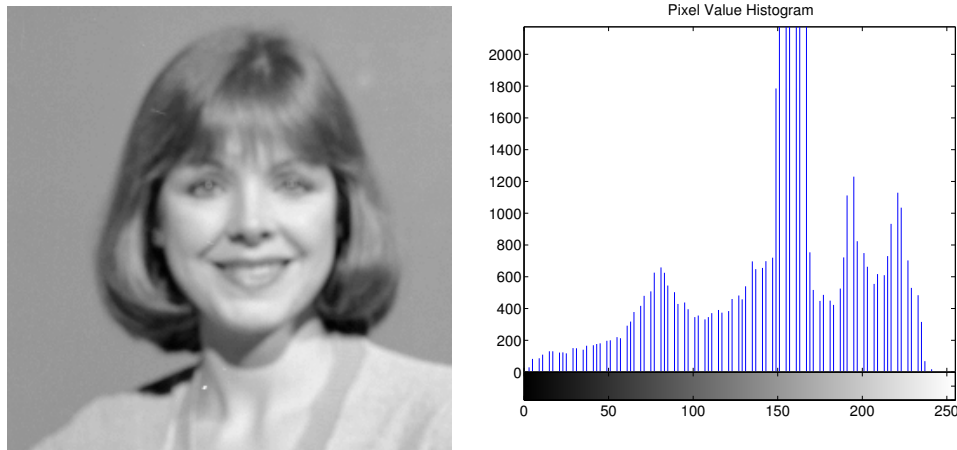


Figure 4.2: Contrast stretched image and its corresponding pixel value histogram

The images relevant for the PermaSense project are in full RGB. A simple contrast stretch on each color channel would result in significant color shifts (see Figure 4.3b). To keep color shifts at a minimum, a slightly more sophisticated method is applied: In the first step, the finished panorama is converted to the **CIE L*a*b** or simply **Lab**⁴ color space. The **Lab** color space has the advantage, that one can stretch the luminosity without color shifts and thus increase contrast without changing the feel of the raw image. After converting the panorama back to the **RGB** color space, each color channel is stretched. This method yields an image with high dynamic range while keeping color shifts at a minimum (see Figure 4.3 for an example).

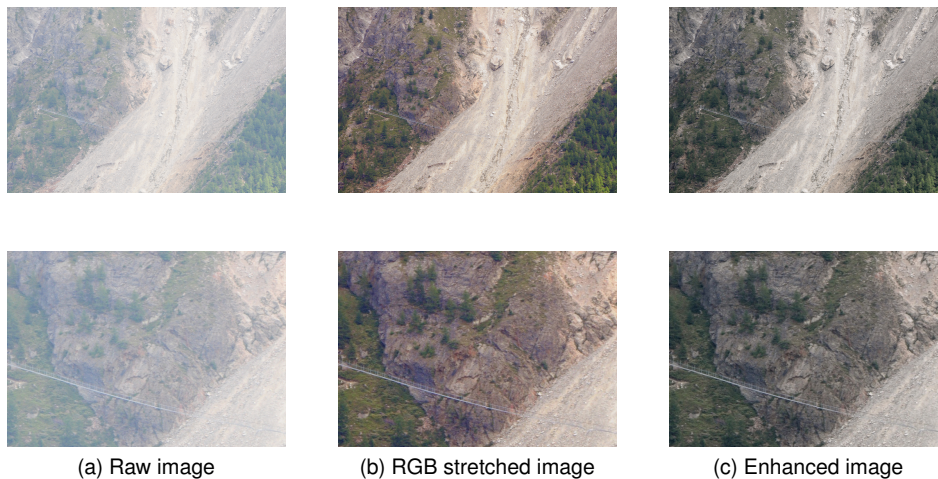


Figure 4.3: Raw image (a), simple RGB stretched with color shift (b) and enhanced image (c) from the Dirruhorn deployment of PermaSense.

4.5 Publishing

Gigapixel panoramas are huge images and can contain well beyond 1 GB of data. This makes it nearly impossible to view them with standard image editing applications without overloading the PC. There are several plugins which enable a user to view full panoramas directly in a web browser without the need of downloading the whole image. Since there is no consensus on which plugin is most suitable for the PermaSense project, the publishing could not be automated yet. One explored possible plugin is the HD View SL⁵ plugin.

⁴**Lab** color space is a color space with parameters **L** for lightness and **a** and **b** for the color values. See http://en.wikipedia.org/wiki/Lab_color_space for further details

⁵<http://research.microsoft.com/en-us/um/redmond/groups/ivm/hdviews/>

Chapter 5

Experiments and Results

5.1 Sight Evaluation

The adjustment of the separate thresholds used in the sight evaluation was achieved with a small, manually labelled test set of evaluation images. The test set contains 77 images with good sight and 70 images with obstructions, additionally it contains 28 night pictures to calibrate the lightness threshold. The thresholds for the different measures were calibrated by visually evaluating the experiments on the test set.

5.1.1 Calibration

The lightness factor provides a clear distinction between day and night shots (Figure: 5.1), but virtually no information about the view can be made. Therefore the lightness is only used to sort out really dark images and pictures captured at night.

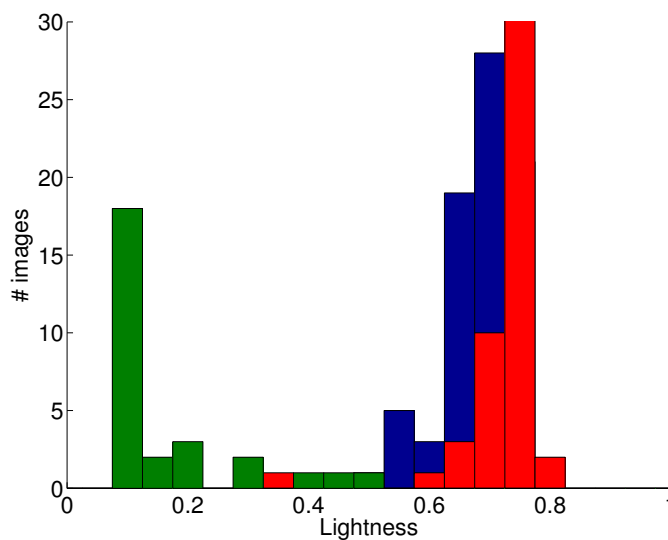


Figure 5.1: Luminosity factor of daytime images (blue and red) compared to nighttime images (green).

From Figure 5.1 above, one can clearly determine, that the lightness factor provides a suitable measure to distinguish between day (blue and red) and night (green) images. However the difference between good (blue) and bad (red) images is not significant which is to be expected. To distinguish between good and bad shots, we further compare the entropy and the sharpness measure:

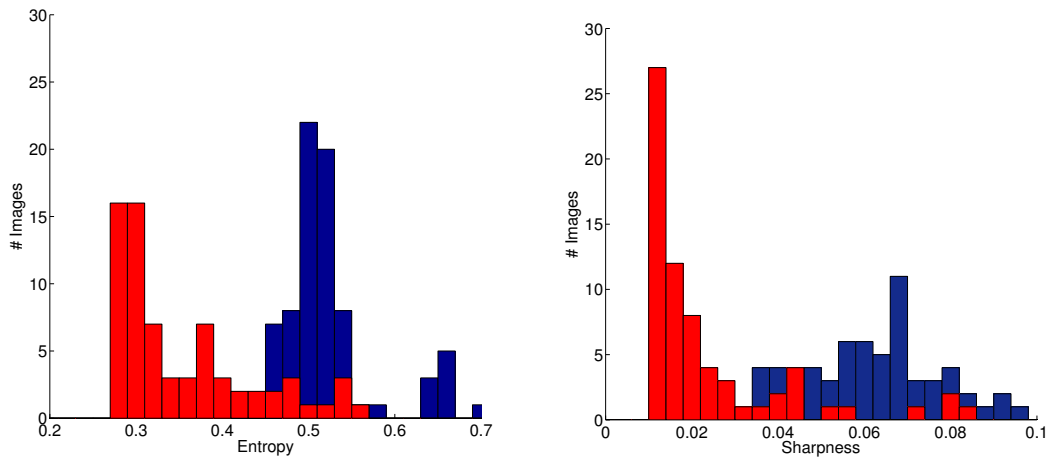


Figure 5.2: Histograms of entropy and sharpness measures on good (blue) and bad (red) sight images.

The entropy and sharpness values provide more information about the landscape in the field of view (see Figure 5.2). From the plot above we manually derived the thresholds for the factors. In [10] the threshold for clear view images was found to be at a sharpness value of 0.0289. Since the calibration of the threshold was not the core of this thesis, we adopted their threshold value. The examples in Figure 5.5 and the histogram in 5.2 show, that this threshold is also suitable for our situation. For the entropy we set the threshold to 0.45. In [10] the threshold for the luminosity is set to 0.2. For panorama images, we only want views with very high intensity. By visually analyzing Figure 5.1, we set our threshold to 0.4. In normal circumstances the intensity value rarely falls below this threshold, unless the image is indeed captured at night. The resulting decision process from this calibration is illustrated in Figure 5.3.

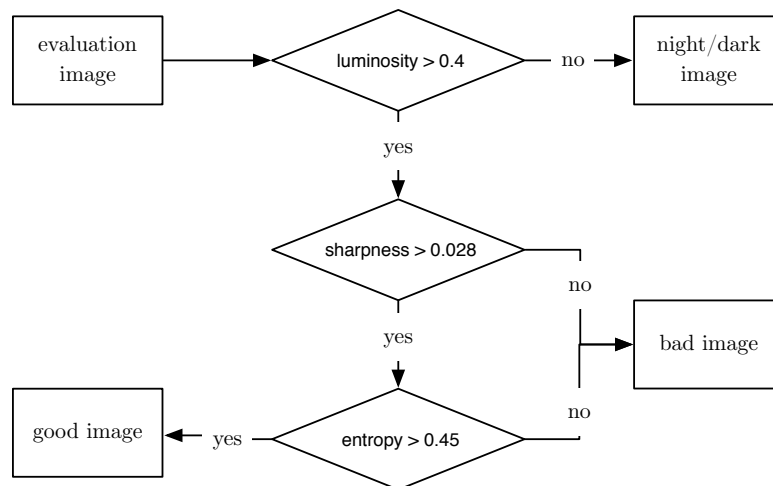


Figure 5.3: Evaluation Process.

Some example images with their corresponding evaluation values are given in Figure 5.5.

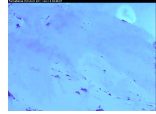






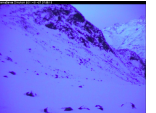
				
Sharpness	0.018154	0.027244	0.017800	0.036046
Entropy	0.277451	0.469841	0.257486	0.567545
Luminosity	0.773879	0.626628	0.750467	0.730179
				
Sharpness	0.026903	0.061987	0.040478	0.028246
Entropy	0.444921	0.438315	0.583938	0.405904
Luminosity	0.739392	0.115705	0.531375	0.577035

Figure 5.4: Example images with evaluation results. Values below the threshold are red, values above green.

It has to be noted however that these thresholds depend strongly on the field of view of the image. An image containing mostly sky will have different values than an image containing mostly land. In an ideal case we have the exact same field of view for every evaluation image. The thresholds can be calibrated in more detail, but that would exceed the scope of this thesis.

5.2 Example Panoramas

This section provides some visual results from one finished panorama image. The resulting resolution in the example below is well below one gigapixel (0.38 gigapixels) and still contains 1.5 GB of data in its raw form. Even at this rather small resolution resolution the image contains huge amounts of information. The example was stitched from 64 raw images with 12 megapixel each. The reason why the resulting panorama does not simply contain 12×64 megapixels = 0.77 gigapixels, is that the overlap covers on average roughly 50% of the image.

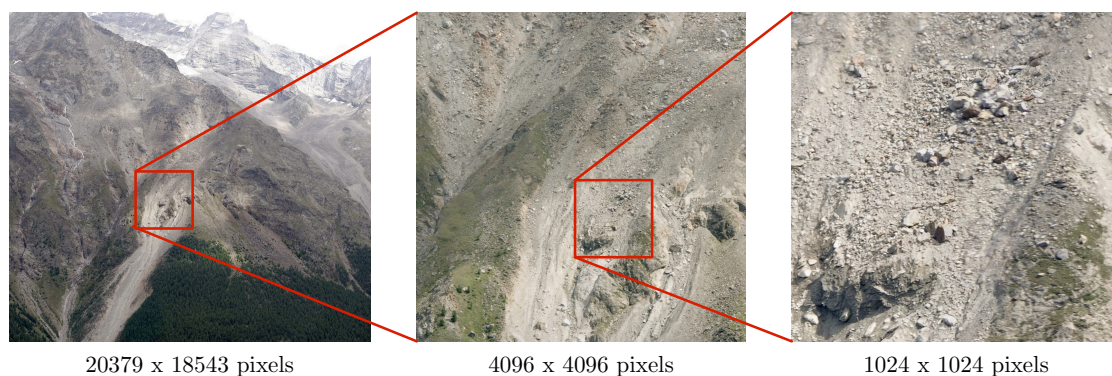


Figure 5.5: Example panorama image with a resolution of 0.38 gigapixels.

Chapter 6

Conclusion and Outlook

6.1 Conclusion

This thesis presents the design of an unsupervised, end-to-end solution for capturing gigapixel images. With the presented method, the available resources are used efficiently. Various methods for evaluating the sight conditions were introduced and integrated into the control loop. It is expected, that with prior evaluations, the power and network bandwidth are used more efficiently, since there will be less pictures captured. The need for human interaction is almost completely eliminated which would also allow for deployment in very hostile environments.

6.2 Outlook

Future work could go towards calibrating the camera settings for every image series, based on a more detailed analysis of the evaluation image. For example, one could adapt the exposure time based in the luminosity of the evaluation image to allow for night shot panoramas. The evaluation of sight condition can be refined by implementing a special feature detection. With such a method one could define points of interest which must absolutely be visible in the finished gigapixel image, such as known landslide areas. An other method to improve the evaluation would be to store good reference images from a specific site for each time of the year and compare the evaluation image to the corresponding reference image. This would surely increase the accuracy, however, it is not at all generic. For every new site a new set of reference images needs to be captured. Also, one would have to guarantee, that the images are always taken with the same field of view. Other generic approaches would be to analyze the histogram of the image or segment the image (sky, snow area, rock) and define rules for each segment (e.g. the sky should be blue).

Appendix A

Panorama Scripting

Example Panorama Script

Generate control points

Generate project file and find corresponding points (control points) in neighboring images:

```
match-n-shift -o camzilla.pto *.jpg
cpfind --multirow -o camzilla.pto camzilla.pto
```

Pruning control points

Control points in clouds are generally bad, since clouds move during a picture sequence. Celeste removes any control points in detected clouds. Ptoclean removes points with large error distances.

```
celeste-standalone -i camzilla.pto -o camzilla.pto
ptoclean -v -o camzilla.pto camzilla.pto
```

Optimize geometric parameters

This step aligns the images in a mosaic and optimizes the geometric parameters (distortions)

```
autooptimiser -a -l -s -m -o camzilla.pto camzilla.pto
```

Optimize photometric parameters

Photometric parameters are optimized to counter exposure differences or vignetting.

```
ptovvariable --vignetting --respons --exposure camzilla.pto
vig_optimize -o camzilla.pto camzillaTEST.pto
```

Creating panorama

The optimal size and crop are calculated and the panorama makefile is generated

```
pano_modify -o camzilla.pto --center --straighten --canvas=AUTO --crop=AUTO camzilla.pto
pto2mk -o camzilla.mk -p panorama camzilla.pto
```

Make panorama

Make the panorama described in the makefile 'camzilla.mk' the -j variable defines how many threads are started. For a quadcore processor one can set this argument to '-j 4';

```
make -j 4 -f camzilla.mk all
```

Final output

The finished panorama is run through a series of contrast enhancement algorithms (see chapter 4.4)

```
./enhancer panorama.tif
```


Appendix B

Example Exif

To get an idea of which informations are contained in a standard Exif of an image, an exert from the Exif created by a Nikon D300s is provided below:

Aperture	8
Auto Focus	On
Lens ID	AF-S VR Zoom-Nikkor 70-300mm f/4.5-5.6G IF-ED
Lens	70-300mm f/4.5-5.6 G VR
Preview Image	(24,935 bytes binary data)
Shutter Speed	1/250
Create Date	2012:08:12 13:00:25.93
Date/Time Original	2012:08:12 13:00:25.93
Modify Date	2012:08:12 13:00:25.93
Thumbnail Image	(8,795 bytes binary data)
Blue Balance	1.21875
Light Value	11.3
Red Balance	1.761719
Scale Factor To 35 mm Equivalent	1.5
Circle Of Confusion	0.020 mm
Depth Of Field	1.57 m (20.38 - 21.95)
Field Of View	4.5 deg (1.67 m)
Focal Length	300.0 mm (35 mm equivalent: 450.0 mm)
Hyperfocal Distance	561.63 m

Appendix C

CamZilla Schedule

The schedule handler for the CamZilla robot is provided by the PermaSense Framework. It is based on the cron job scheduler in Unix and thus provides the possibility to trigger events at predefined times. The following is the schedule's line for a panorama capturing sequence on the CamZilla robot. The first five characters define the time at which the task is triggered (minute, hour, day, month, weekday), an asterisk '*' stands for every possible value. For the example below the trigger is set to 6:00 AM every day. The arguments are:

- plugin: tells the CoreStation that the task corresponds to a plugin.
- CamZillaPlugin: the plugin for the CamZilla robot.
- start(x,y): starting position.
- pictures(n,m): capture a mosaic of n×m images.
- rotation(u,v): rotate u degrees between horizontally aligned images and v degrees between vertically aligned images.
- delay(t): delay t between each picture in seconds.
- batch([1,0]): download all images in one package true/false (1/0).
- gphoto2(...): a configuration string for the gPhoto library.

```
0      6      *      *      *      plugin CamZillaPlugin
>start(220,73)
>pictures(8,6)
>rotation(2.8,1.8)
>delay(2)
>batch(1)
>gphoto2(...)
```


Appendix D

Workflow

The figure below gives a detailed overview of all the processes that are running when the control loop is active. The Communication stream is in the form of a database, where all the components of the GSN regularly report their status. By querying that database, the panorama creator (PC) or any other external process can retrieve the current and all past states of any desired component of the GSN. The first step for the PC is to listen for a positive evaluation result. As soon as the GSN reports a successful evaluation, the PC will 'listen' for successful completion of the panorama job. The stitching is not started until the GSN reports, that all images have been transmitted successfully. The PC is run at regular intervals and is totally independent of the GSN. Since the database also contains old status messages, the execution of PC does not have to be synchronized with the GSN or any other special event.

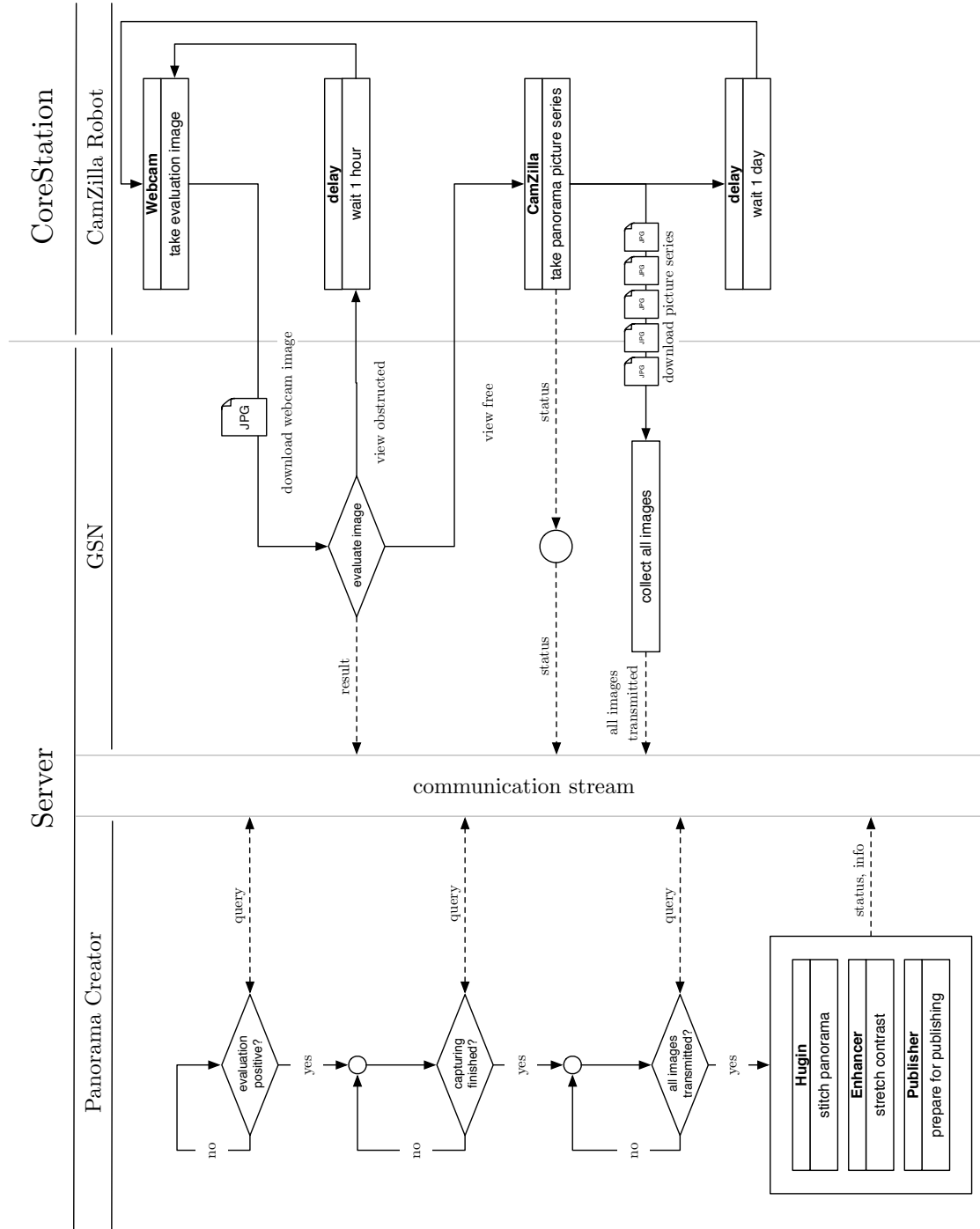


Figure D.1: Process flow of the CamZilla robot.

Appendix E

Original Problem

SEMESTER THESIS

for

Benjamin de Capitani

Advisor: Matthias Keller

Co-Advisor: Dr. Jan Beutel

Start date: July 2, 2012
End date: September 28, 2012

CamZilla: A GigaPixel Panorama Robot in the Mountains

Introduction

For automatically capturing GigaPixel-sized panorama images in the mountains, we developed an outdoor camera platform that allows us to remotely pan, tilt and control two cameras that are mounted on a single, moving head: While a low-resolution, low-resolution network camera is intended for visually evaluating the current sight conditions, the installed Nikon DSLR camera is used for capturing series of high-resolution images. To this day, the system is able to automatically coordinate the movement of the robot head and the capturing of high-resolution images using the Nikon DSLR camera. The decision if a new panorama is to be taken and the process of generating panorama images out of several, single images is currently done manually.

Goals of this Thesis

To further automate the creation and publication of high-quality panorama images, the goal of this thesis is to create a closed data capturing loop consisting of a panorama capture decision, the capturing of a series of single images for creating panorama images, the post-processing of captures images, *i.e.*, stitching, and the publication of generated images on a server.

The solution has to be developed within the existing PermaSense software architecture, it is recommended to reuse previously gained results as much as possible. In particular, it is recommended to make use of a recently developed software component that automatically generates job schedules for an actuator based on real-time sensor readings [1].

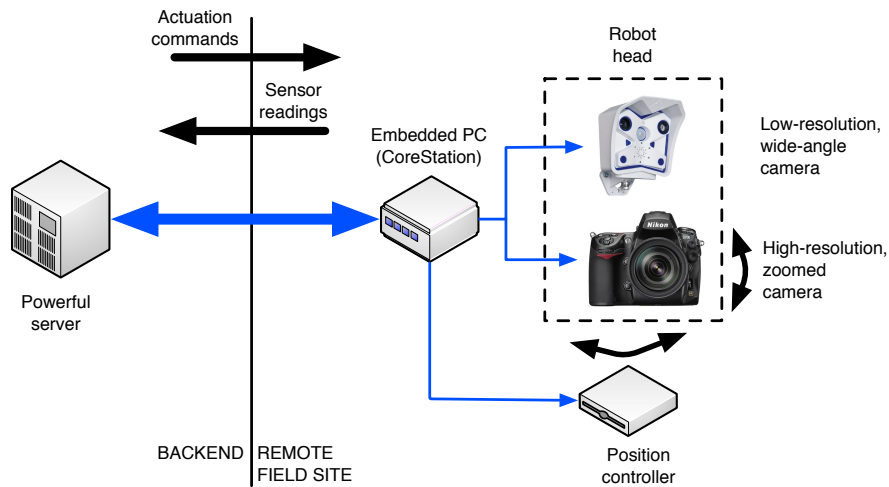


Figure 1: CamZilla scenario. Devices installed on the remote field site are limited to independently, *i.e.*, also when connectivity is lost, fulfilling actuation schedules that have been calculated at a powerful backend server beforehand.

In order to reach these goals, the project should proceed according to the following steps:

1. Familiarization with PermaSense software architecture, in particular GSN and BackLog. Setup of an own development system consisting of an instance of GSN and a so called CoreStation running the Backlog software.
2. Implementation of a simple, exemplary decision process: After positioning the low-resolution IP camera that is also mounted on the robot, a very small number of images is to be taken and transferred to a backend server. After running a simple filter, *e.g.*, counting the number of dark pixels, a new schedule is automatically generated and transferred to the panorama robot.
3. Automated creating and publication of the captured image. Images should be made available at different resolutions allowing users to preview captured contents before downloading potentially very large full-size images.
4. Refinement of decision process, *e.g.*, by evaluating the number of usable images features, or by evaluating the sight conditions from pictures taken by the IP camera.
5. The student should interpret gathered results and document findings in the report.

Organization

- Duration of the Work:
This Semester Thesis starts July 2, 2012 and has to be finished no later than September 28, 2012.
- Project Plan:
A project plan with its milestones is held and updated continuously. Unforeseen difficulties that change the project plan have to be documented and should be discussed with the advisors in a timely manner.
- Subversion Repository:
All relevant files, *i.e.*, presentations, scripts, and text documents, should be stored in the Subversion repository that has been assigned to the project. Timely and frequent data synchronization prevents data loss and allows to collaborate more easily.

- Weekly Meetings/Reports:
In regular (weekly) meetings with the advisors, the current state of the work, potential difficulties as well as future directions are discussed. The day before the weekly meeting a brief status report should be sent to the advisors commenting on these issues, in order to allow an adequate meeting preparation for the student and the advisors.
- Initial Presentation:
Approximately two to three weeks after the start the student presents the objectives of the work as well as some background on the topic. The presentation should not exceed 5 minutes and consist of no more than three slides.
- Final Presentation:
By the end of the project, the student will present the achieved result. The presentation should not exceed 15 minutes.
- Documentation:
At the end of the project, no later than *September 28, 2012*, the student will have to hand in a written report. Together with the system implementation/software this report is the main outcome of the project. Code has to be commented extensively, allowing a follow-up project.
- Evaluation of the work:
The criteria for grading the work are described in [2].
- Finishing up:
The required resources (e.g., laptop, keys) should be cleaned up and handed back in.

References

- [1] D. Burgener, "Alpthal Hydrological Catchment WSN." Computer Engineering and Networks Lab, ETH Zürich, Switzerland, June 2012.
- [2] E. Zitzler, "Studien- und Diplomarbeiten, Merkblatt für Studenten und Betreuer." Computer Engineering and Networks Lab, ETH Zürich, Switzerland, Mar. 1998.

Bibliography

- [1] Bernhard Buchli, Mustafa Yuceel, Roman Lim, Tonio Gsell, and Jan Beutel. Demo abstract: Feature-Rich experimentation for wsn design space exploration. In *Proceedings of the 10th International Conference on Information Processing in Sensor Networks (IPSN 2011)*, pages 115–116, Chicago, IL, USA, 2011. ACM/IEEE.
- [2] Reynald Delaloye Fabian Neyer. Displacement detection on rock glaciers using webcam images: A case study in the mattertal. *Swiss Geoscience Meeting*, 2011.
- [3] Ali Salehi Karl Aberer, Manfred Hauswirth. A middleware for fast and flexible sensor network deployment. *VLDB: Very Large Data Bases*, 2006.
- [4] Matthias Keller, Mustafa Yuceel, and Jan Beutel. High resolution imaging for environmental monitoring applications. In *International Snow Science Workshop 2009: Programme and Abstracts*, pages 197–201, Davos, Switzerland, 2009.
- [5] Douglas A. Kerr. The proper pivot point for panoramic photography. 2008. http://dougkerr.net/pumpkin/articles/Pivot_Point.pdf.
- [6] Dave O'Brien. Cassini lossy compression, image entropy. <http://www.astro.cornell.edu/research/projects/compression/>.
- [7] Belzner H. Rigoll G. Ilic S Pavlic, M. Image based fog detection in vehicles. *Intelligent Vehicles Symposium, IEEE*, pages 1132–1137, 2012.
- [8] Ashley Walker Robert Fisher, Simon Perkins and Erik Wolfart. Image processing learning resources, 2004. http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm.
- [9] Nicolas Hautière Romain Gallen, Aurélien Cord and Didier Aubert. Towards night fog detection through use of in-vehicle multipurpose cameras. *Intelligent Vehicles Symposium (IV), IEEE*, pages 399–404, 2011.
- [10] Dominic Rüfenacht. Temporally consistent snow cover estimation for the PermaSense field camera. *Ecole Polytechnique Fédérale de Lausanne, Master Semester Project*, 2011.
- [11] P. F. Alcantarilla S. Bronte, L. M. Bergasa. Fog detection system based on computer vision techniques. *IEEE Conference on Intelligent Transportation Systems, Saint-Louis, Missouri, USA*, pages 1–6, 2009.
- [12] Philipp Schneider, Daniel Burgener, Jan Beutel, Andreas Wombacher, and Jan Seibert. Sensor-based actuation of water samplers in wireless sensor networks. In *American Geophysical Union (AGU) Fall Meeting*, San Francisco, USA, 2012. AGU.