

**From disparity to depth on mobile phone stereo cameras**

A Project

submitted in partial fulfilment

of the requirements for the Degree

of

Master of Science

by

Paz Ilan

I.D. 200326767

Tel Aviv University

September 2020

Table of Contents

[0. Abstract 7](#_Toc52808143)

[1. Background and problem definition 8](#_Toc52808144)

[1.1. Pinhole models 8](#_Toc52808145)

[1.1.1. Camera parameters 9](#_Toc52808146)

[1.1.2. Stereo vision 10](#_Toc52808147)

[1.2. Feature extraction 14](#_Toc52808148)

[1.2.1. Features 14](#_Toc52808149)

[1.2.2. Chessboard corners 16](#_Toc52808150)

[1.2.3. Facial landmarks 17](#_Toc52808151)

[1.3. Mobile camera handling 19](#_Toc52808152)

[1.3.1. Module differences 19](#_Toc52808153)

[1.3.2. Calibration process 21](#_Toc52808154)

[1.3.3. Rectification process 21](#_Toc52808155)

[1.3.4. From theory to reality 22](#_Toc52808156)

[1.4. Our research 25](#_Toc52808157)

[1.4.1. Basic motivation 25](#_Toc52808158)

[1.4.2. Research question 25](#_Toc52808159)

[2. Experiments setup details 26](#_Toc52808160)

[2.1. Modules used 26](#_Toc52808161)

[2.2. Charts used 26](#_Toc52808162)

[2.3. Physical setup and conditions 27](#_Toc52808163)

[2.4. Theoretical setup 29](#_Toc52808164)

[3. Disparity issues investigations 30](#_Toc52808165)

[3.1. Heating 32](#_Toc52808166)

[3.2. Distortion 35](#_Toc52808167)

[3.3. Non baseline translation 39](#_Toc52808168)

[3.4. Device orientation 42](#_Toc52808169)

[3.5. Focus 43](#_Toc52808170)

[3.5.1. Focus fail 46](#_Toc52808171)

[3.5.2. Wide to Tele focus 46](#_Toc52808172)

[3.5.3. Same camera focus repeatability 53](#_Toc52808173)

[3.6. Disparity to depth conversion 65](#_Toc52808174)

[4. Conclusions 71](#_Toc52808175)

[4.1. Phase 0 demo 71](#_Toc52808176)

[4.1.1. Limitations 71](#_Toc52808177)

[4.1.2. Suggested flows 74](#_Toc52808178)

[4.1.3. Block descriptions 75](#_Toc52808179)

[5. Future work 78](#_Toc52808180)

[5.1. Phase 0 demo continuation 78](#_Toc52808181)

[5.2. Further issue handling 78](#_Toc52808182)

[5.2.1. Temperature 78](#_Toc52808183)

[5.2.2. Focus 78](#_Toc52808184)

[5.2.3. Face data collection and extraction 79](#_Toc52808185)

[5.3. Physical improvements 80](#_Toc52808186)

[5.4. Combining with existing methods 80](#_Toc52808187)

[6. References 82](#_Toc52808188)

Figures

[Figure ‎1‑1 - Pinhole camera illustration. Taken from http://ksimek.github.io/2013/08/13/intrinsic/ 8](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807727)

[Figure ‎1‑2 - An illustration of the canonical case. Taken from https://users.cs.cf.ac.uk/Dave.Marshall/Vision\_lecture/node11.html 11](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807728)

[Figure ‎1‑3 - Relation between disparity and depth, Taken from http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12\_hres.pdf 11](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807729)

[Figure ‎1‑4 - Non cacnonical stereo camera case. Image taken from https://www.sanyamkapoor.com/machine-learning/an-introduction-to-epipolar-geometry/ 12](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807730)

[Figure ‎1‑5 - Features illustration. Taken from https://opencv-python-tutroals.readthedocs.io/en/latest/py\_tutorials/py\_feature2d/py\_features\_meaning/py\_features\_meaning.html#features-meaning 14](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807731)

[Figure ‎1‑6 - Chessboard chart used to test and verify our results. 16](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807732)

[Figure ‎1‑7 - Landmarks results of OpenFace. Taken from https://github.com/TadasBaltrusaitis/OpenFace/wiki/Output-Format 19](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807733)

[Figure ‎1‑8 - The two common lens distortion types illustrated. Taken from https://clickitupanotch.com/lens-distortion/ 20](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807734)

[Figure ‎1‑9 - Generic rectification example. Taken from https://web.stanford.edu/class/cs231a/course\_notes/03-epipolar-geometry.pdf 22](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807735)

[Figure ‎2‑1 - Verification chart used in the experiments. 26](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807736)

[Figure ‎2‑2 - Face chart used in the experiments. Faces have uniform IPD of 6.3[cm]. 27](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807737)

[Figure ‎2‑3 - Simulated chart depth scene 29](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807738)

[Figure ‎3‑1 chessboard corners disparity and depth estimation using naïve stereo theory. Left image is in [pixels], middle image is in [cm] and right is in [%]. 30](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807739)

[Figure ‎3‑2 - Disparity of faces and chessboard at the same distance 31](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807740)

[Figure ‎3‑3 - Disparirty change as a function of measured module temperature 32](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807741)

[Figure ‎3‑4 - Disparity change as a function of time. Each color tracks a specific chart point. 33](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807742)

[Figure ‎3‑5- Found distortion from flat scene for each focus distance 36](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807743)

[Figure ‎3‑6 - Lens distortion vs radius for several distances. Bright green is the found polynomial from the calibration. Right image is the normal polynomial and the left is the polynomial with the odd degrees added. Units of both plots and both axes are in [pixels]. 37](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807744)

[Figure ‎3‑7 - From left to right, Non disparity error, radial error component and radial error component with optimal non-disparity offset 38](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807745)

[Figure ‎3‑8 - Illustration of the y-axis translation remaining after rectification 39](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807746)

[Figure ‎3‑9 - Illustration of a truly canonical rectification result. 40](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807747)

[Figure ‎3‑10 - Disparity (DY) and non-dispartiy error (DX) for a dual depth scene, without (first row) and with (second row) the tranlsation fix. 41](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807748)

[Figure ‎3‑11 - Disparity result for different module orientations 42](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807749)

[Figure ‎3‑12 - Simplified example of focus distances. Image taken from https://c.mi.com/thread-904594-1-0.html 43](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807750)

[Figure ‎3‑13 - Focus position and scale between wide and tele image pairs analysis. Bottom right shows the division of the wide focus position by the tele (which should indicate scale to some extent) 45](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807751)

[Figure ‎3‑14 - Maximum disprity range per cluster over 7 focus attempts. 47](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807752)

[Figure ‎3‑15- Maximum disprity range per cluster over 7 focus attempts after scale correction. 48](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807753)

[Figure ‎3‑16 – folded module disparity(left) and non-disparity error(right), before (top) and after scale fix (bottom) with per corner range next to each corner 49](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807754)

[Figure ‎3‑17 - Close chart disparity (DY) and non-disparity error (DX) before (top) and after (bottom) fixing the optimal scale for it (1.0005) 51](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807755)

[Figure ‎3‑18 - Far chart disparity (DY) and non-disparity error (DX) before (top) and after (bottom) fixing the optimal scale for it (1.0023) 52](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807756)

[Figure ‎3‑19 -Entire image disparity (DY) and non-disparity error (DX) before (top) and after (bottom) fixing the optimal scale for it (1.0019) 52](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807757)

[Figure ‎3‑20 - Disparity range for each corner over 7 refocusings. 53](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807758)

[Figure ‎3‑21 - Averaging the disparity over different focus attempt 55](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807759)

[Figure ‎3‑22 - Simulation result before (left) and after (right) adding scale,rotation and translation between the scenes. 59](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807760)

[Figure ‎3‑23 - Essential matrices extracted from the simulated scene before (left) and after (right) the added rotation,translation and scale 59](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807761)

[Figure ‎3‑24 - Regular disparity range (left), optimal scale fix (middle) and the naive essential fix (right) 60](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807762)

[Figure ‎3‑25 - Rotation and translation from Essential decomposition of the baseline image (left), a random focus attempt (middle) and the same attempt that has been fixed with the optimal scale (right) 61](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807763)

[Figure ‎3‑26 - Disparity ranges for different optimization methods 62](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807764)

[Figure ‎3‑27 - Optimal scale corretion, disparity ranges (two left plots) and relative error (two right plots), before (leftmost of each pair) and after optimal scale correction (rightmost of each pair) 64](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807765)

[Figure ‎3‑28 - Results of a study done in America on human face interpuppilary distance. Picture from New biometrics Ear-Eye Pitch, by Leonid Naimark, Paul C. Briggs. 67](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807766)

[Figure ‎3‑29 - Theoretical relative error in distance estimation as a function of IPD estimation error. 68](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807767)

[Figure ‎3‑30 - Disparity from depth estimated from IPD (above), normal disparity calculated between matching landmarks (filled circles) and disparity between chessboard corners (empty circles) in normal images (below) of charts at the same depth, after the fixes we mentioned before 69](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807768)

[Figure ‎3‑31 - Results of fixing the disparity of a chessboard chart from the face's found offset correction 70](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807769)

[Figure ‎4‑1 - Focus calibration flow 74](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807770)

[Figure ‎4‑2 - Application phase flow 74](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807771)

[Figure ‎4‑3 - Learning phase flow 75](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807772)

[Figure ‎4‑4 - Face analysis flow 75](file:///C:\REPOSITORIES\MscProject\Project%20book%20-%20Paz%20Ilan.docx#_Toc52807773)

# Abstract

One of the most prevalent uses of stereo photography is estimating the depth of objects in the scene using epipolar geometry, the camera baseline, known camera parameters and the disparity between the same object in the different cameras. This enables the estimation of the actual object distance from the cameras.

The naïve model is not sufficient when dealing with real life images captured with mobile phone camera devices. This is due to their divergence from the pinhole model, focus mechanism, manufacturing tolerances, small setup size and usage of cheaper and less robust materials that is more sensitive to outside effects.

Our project focuses on researching the issues with converting the found disparity to actual depth, on a mobile stereo phone setup that consists of two cameras with different focal lengths. We focus on trying to isolate different factors that cause deviation in the disparity, analyze the effect of each one and try to formalize and test a solution if possible. We also look into adding an additional online learning scheme that will enable us to learn a fix during the phone's normal operation mode.

Our research shows that under some strict limitations we can devise a flow that can perform fix the disparity estimation error from around 30% relative error down to 5% relative error.

# Background and problem definition

## Pinhole models

Modeling of the imaging process is a fundamental task of computer vision. Many camera models have been proposed for this purpose as no single model fits all prospective applications. The pinhole camera model is the most basic model for perspective cameras, however in most case, it is quite accurate. Therefore, it is the most used geometric camera model in the field of computer vision.

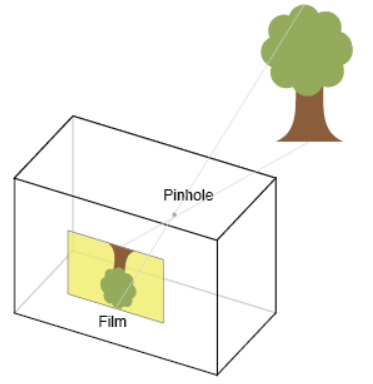


Figure ‎1‑1 - Pinhole camera illustration. Taken from <http://ksimek.github.io/2013/08/13/intrinsic/>

The pinhole camera model defines the geometric relationship between a 3D point and its 2D corresponding projection onto the image plane. When using a pinhole camera model, this geometric mapping from 3D to 2D is called a perspective projection. The pinhole model is composed of a set of intrinsic camera parameters and extrinsic camera parameters that indicate the position and properties of the cameras and in turn compose the perspective projection. The projection can be decomposed into two matrices like so:

*( 1 )*

Where x, y are image coordinates, X,Y,Z are real world coordinates, K is the intrinsic matrix and E is the extrinsic matrix.

### Camera parameters

The intrinsic matrix can be parameterized as

*( 2 )*

Each intrinsic parameter describes a geometric property of the camera.

**Focal length ():** The focal length is the distance between the pinhole and the sensor (a.k.a. image plane). In a true pinhole camera, both will have the same value, however in practice they can differ due to a number of reasons:

* Flaws in the digital camera sensor.
* The image has been non-uniformly scaled in post-processing.
* The camera's lens introduces unintentional distortion.
* The camera uses an anamorphic format, where the lens compresses a widescreen scene into a standard-sized sensor.
* Errors in camera calibration.

In all of these cases, the resulting image has non-square pixels. In the device tested in this project, we assumed that the pixels are square, so same focal length value is used for parameters. This assumption proved to be valid for our device.

**Principal point offset ():**  The camera's "principal axis" is the line perpendicular to the image plane that passes through the pinhole. Its intersection with the image plane is referred to as the "principal point". The "principal point offset" is the location of the principal point relative to the sensor's origin.

**Axis skew (s):** Axis skew causes shear distortion in the projected image.For our purposes we assume the skew is 0 as it has proven accurate for our models in the past.

The extrinsic matrix is:

*( 3 )*

Where R is a 3x3 rotation matrix and t is a 3x1 translation vector that signifies the orientation and location of the camera in relation to the chosen world coordinate system (the choice is arbitrary, but all other relative sizes most be relative to the same coordinate system).

### Stereo vision

As mentioned above, the camera projects the 3D world onto the 2D image plane. In this process the information about the 3D structure of the scene is lost. A common method to estimate the 3D structure, is to use a stereo camera setup and recover the depth of every pixel in the scene.

In the canonical case we assume:

* The cameras are identical.
* The optical axes of the cameras are parallel (i.e. there is no rotation between them).
* The line connecting the camera lens centers is called the baseline (marked by the size d in Figure ‎1‑2), and it is perpendicular to optical axis of the cameras. The cameras are otherwise identical in their location (meaning there is no translation between the camera centers other than the baseline).
* There is no distortion in both cameras.

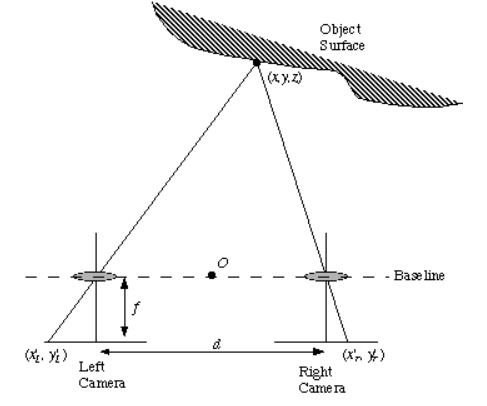


Figure ‎1‑2 - An illustration of the canonical case. Taken from <https://users.cs.cf.ac.uk/Dave.Marshall/Vision_lecture/node11.html>

In this simplified case, an object will appear in the same y pixel coordinate in both sensors and the difference will only be in the x pixel coordinate. This difference is called the disparity and we will mark it by Δ.

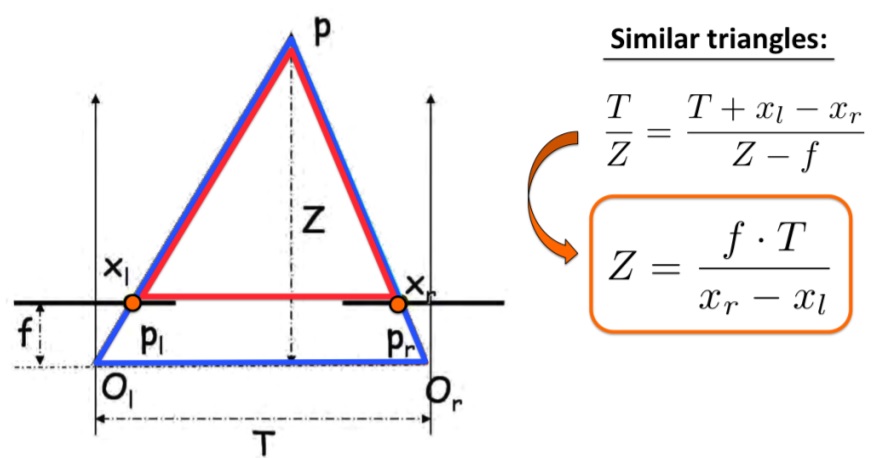


Figure ‎1‑3 - Relation between disparity and depth, Taken from <http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf>

By using triangle similarity, as shown in Figure ‎1‑3, the connection between the baseline (T), the focal length of the cameras (f), the object depth (Z) and the disparity (Δ) is:

*( 4 )*

And thus, as long as we can get pixel correspondence (as will be explained in ‎1.2) between the two images and we know the physical parameters of the setup, we can easily deduce the depth of each pixel (for which there exists a correspondence). Also, since each pixel will be in the same y pixel coordinates in the other image, the pixel matching problem is only 1D and not 2D as in the general case.

This canonical case is, however, very unlikely to occur in a real-world scenario. In order to use the connections derived above, we employ a calibration process for the stereo cameras (as we will discuss this in ‎1.3.2) and then rectify the images accordingly (as we will discuss in ‎1.3.3).

In the non-canonical case, we can still derive a connection between the images to help us turn the pixel matching between stereo images problem from 2D to 1D, but in a more complicated way using epipolar geometry.

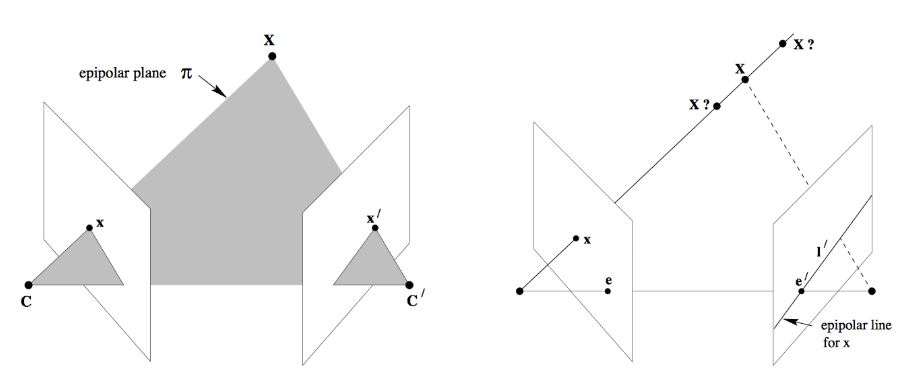


Figure ‎1‑4 - Non cacnonical stereo camera case. Image taken from https://www.sanyamkapoor.com/machine-learning/an-introduction-to-epipolar-geometry/

Figure ‎1‑4 shows a point in the world, X, and the camera centers C and C'. These three points define a plane. This plane is referred to as the epipolar plane. The points x and x' (the projections between the world point X and each camera) are located within this plane. This property provides the constraint needed to turn the point correspondence problem from a 2D problem to a 1D problem. More specifically the points lie along the line that is the intersection between the epipolar plane and the image plane, which is called the epipolar line. The intersection of the line between both camera centers (previously mentioned as baseline) and the image plane is called the epipole, and all epipolar lines go through it. Thus, if we know the epipoles, the camera center and a point in one image, finding its corresponding point in the second camera is a search along a line and it is 1D. It is important to note that this property is not dependent on the scene in any way, and it's true for any depth. The canonical case stated above, is the case in which all epipolar lines are horizontal and that is why it simplifies the problem greatly (as the epipolar plane is trivial to find).

Using vector calculus and some algebraic manipulations it can be shown that the relation between corresponding points in stereo images is formulized as the fundamental matrix, F, and the following formula:

*( 5 )*

Furthermore, the fundamental matrix can be deconstructed into:

*( 6 )*

Where K is the intrinsic camera parameters matrix we've mentioned above, t is the translation vector between both camera centers, is the cross product operator, and R is the rotation matrix between both cameras. Thus we can use the physical relations between both cameras to get the connection between pixels in the different cameras.

## Feature extraction

### Features

Image features, such as edges and interest points, provide rich information on the image content. They correspond to local regions in the image and are fundamental in many applications in image analysis: recognition, reconstruction, matching, etc.

Image features yield two different types of problem: the detection of area of interest in the image, typically contours, and the description of local regions in the image, typically for matching between different images. In the general case, they relate to the differential properties of the intensity function, for instance the gradient or the laplacian that are used to detect intensity discontinuities that occur at contours.



Figure ‎1‑5 - Features illustration. Taken from <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_features_meaning/py_features_meaning.html#features-meaning>

In Figure ‎1‑5 we can see an example of an image with several patches zoomed in that illustrate what is a good feature and what isn’t and why, by trying to locate them in the full image. Patch A and B are flat surfaces, without any unique information, and they appear across a large section of the original image. This makes it difficult to find the exact location of these patches. Patches C and D are better, they are edges of the building. You can find an approximate location, but the exact location is still difficult because, along the direction of the edge, it is almost the same everywhere. Perpendicular to the edge, we get high variance in the image, so the edge is a much better feature compared to the flat area, but still not good enough. Finally, patches E and F are some corners of the building. They can be easily located because they are unique and wherever you move this patch, it will look different. So they can be considered as good features.

Once features are extracted from the images, we apply feature matching. In order to perform the matching we need a way to describe our features, also called a descriptor. The most basic descriptor we can think of is simply a patch around the feature we found without any manipulations, and we can use simple cross correlation between it and the other image in a sliding window fashion and find the location with the highest score.

There are a lot of different variants of feature extractors and descriptors such as ORB, SIFT, SURF etc. In this study we used descriptors and feature matcher that are based mostly on SURF (speeded up robust features).

Our feature extractor uses difference of Gaussians (the difference between the image filtered with two different Gaussian kernels - DOG) to approximate Laplacian of Gaussian (LOG), and its local extrema points to mark feature points. It then uses the hessian matrix in order to disqualify 1D features and features that are not significant enough. The descriptor we use is quite basic and consists of a window around the feature point and the feature matching is simply cross correlation between different feature windows. In order to get sub-pixel accuracy we use peak refinement on the windows in the different images to get the maximum correlation score and disqualify bad matches using convexity tests. The naïve descriptors and matching are used since they are fast and proven reliable in our past experiments with rectified images (since they have no rotation and little scale invariance, they are unsuitable for unrectified images).

### Chessboard corners

In order to test and verify results we need a feature extractor and matcher which is robust to different cameras, orientations, and scales, and is accurate and repeatable to make sure that the errors we will see are not feature finding/matching errors. In order to achieve that, we opted not to use our normal feature extractors (as they sometime yield false matches and matches that are less accurate, especially when using extremely different cameras like we do) but to use a well-known chart to extract feature in a more reliable fashion.

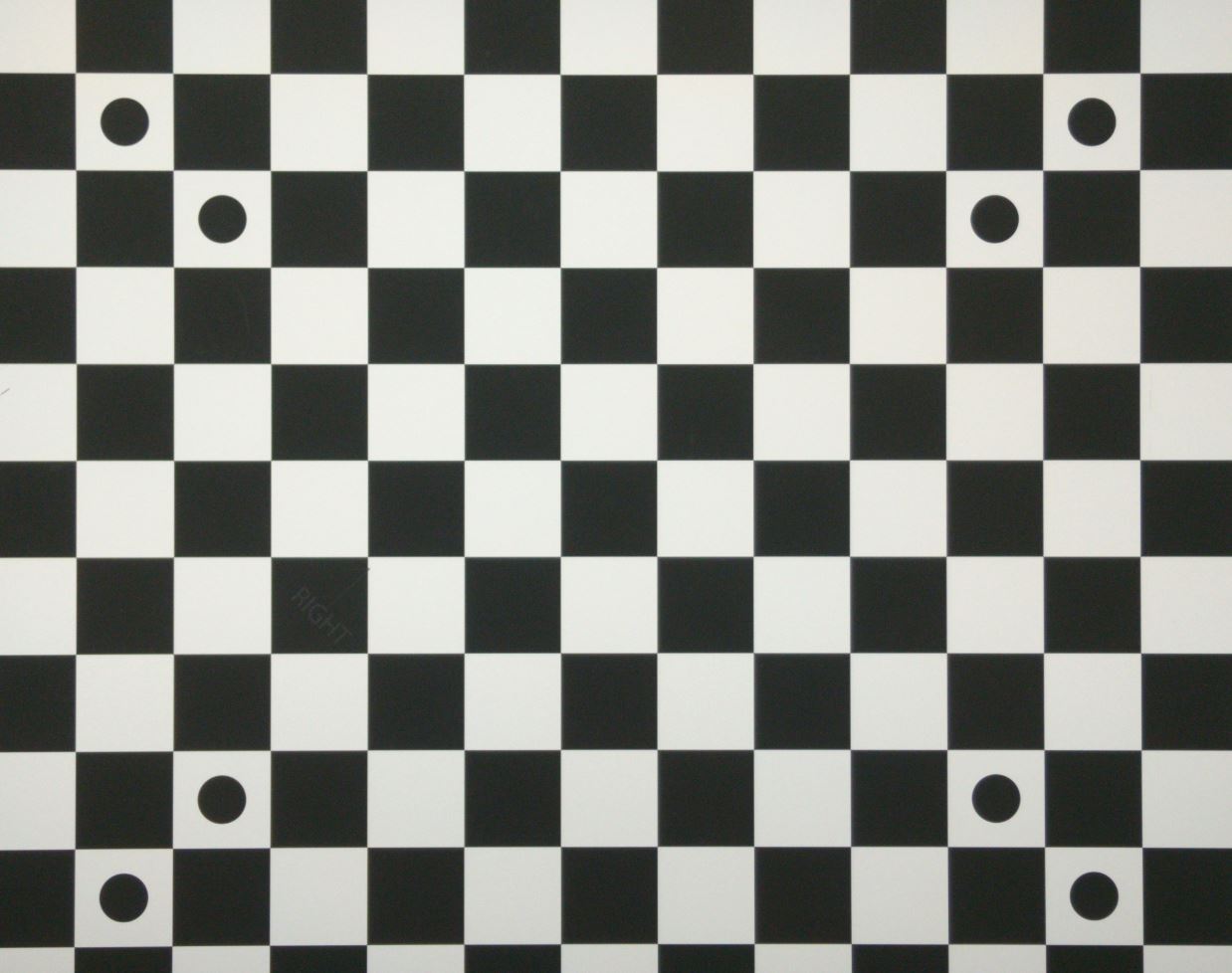


Figure ‎1‑6 - Chessboard chart used to test and verify our results.

The chart we chose to use is the chessboard chart with a known square size, as customary in most camera calibration algorithms, along with several circles marked on it as shown in Figure ‎1‑6.

In order to extract features from this chart and match them we start by finding the four circle pairs by using the gradient image, connected component analysis, circles relationship between area and circumference and radius similarity between adjacent circles. Since there are only 4 circle pairs in the entire chart, and we assume the rotation between the cameras isn't extreme to the point that one pair will be confused with another, the matching between circles in different images is trivial. We then continue to extract the corners between all of the squares by using Harris corner detector along with a window to check for crossing from white to black in all directions. This leaves us with features that are the corners between all of the squares that are sub-pixel accurate, and since we start from the circles that are already matched, the matching is inherent.

This procedure is applied in a lab\factory setting and is not commonly used for on-the-fly tasks.

### Facial landmarks

Using the chessboard chart can only be applied in the phone factory, during assembly, but during regular use, the end-user doesn’t have unique images with known real world sizes. For this reason, we chose using a common object in a real world scenario that has statistical data about sizes within it – human faces. For this purpose we use another type of features – facial landmarks.

Detecting facial landmarks is a subset of the shape prediction problem. Given an input image (and normally an ROI that specifies the object of interest), a shape predictor attempts to localize key points of interest along the shape. In the context of facial landmarks, our goal is to detect important facial structures on the face using shape prediction methods.

Detecting facial landmarks is therefore a two-step process:

* Step #1: Localize the face in the image.
* Step #2: Detect the key facial structures on the face ROI.

Face detection (Step #1) can be achieved in a number of ways. We used an off-the-shelf face detector to obtain the face bounding box (i.e., the (x, y)-coordinates of the face in the image).

Given the face region we apply Step #2: detecting key facial structures in the face region. There are a variety of facial landmark detectors, but all methods essentially try to localize and label the following facial regions:

* Mouth
* Right eyebrow
* Left eyebrow
* Right eye
* Left eye
* Nose
* Jaw

For our work we used an open source library called OpenFace. It uses the matlab's vision toolbox for face detection and then a pre-trained CLM - Constrained Local Model.

A Constrained Local Model is class of methods of locating sets of points (constrained by a statistical shape model) on a target image.

The general approach is to:

* Sample a region from the image around the current estimate, projecting it into a reference frame.
* For each point, generate a "response image" giving a cost for having the point at each pixel.
* Searching for a combination of points which optimizes the total cost, by manipulating the shape model parameters.

The response image is the result of training the model on the "faces in the wild" dataset.

The model uses both rigid and non-rigid parameters to adjust the landmarks. The rigid parameters include scale, rotation and translation of the landmarks while the non-rigid parameters compensate for deformations caused by expression and identity.

The resulting landmarks are:

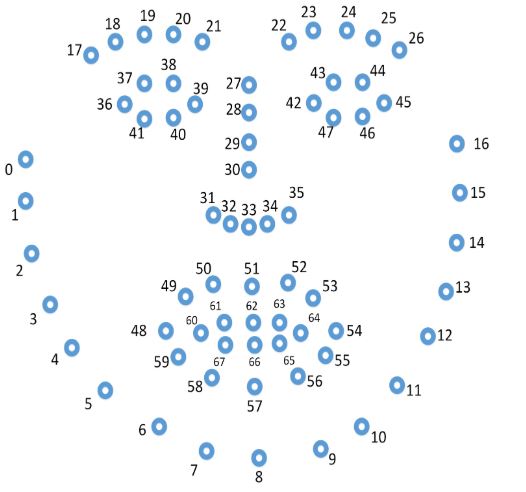


Figure ‎1‑7 - Landmarks results of OpenFace. Taken from <https://github.com/TadasBaltrusaitis/OpenFace/wiki/Output-Format>

We also use the rigid rotation parameter to fix the face orientation so that all landmarks will be parallel to the camera to suit our purposes.

## Mobile camera handling

### Module differences

Since our work is designed for mobile phone cameras, we need to consider several ways in which they differ from the theory mentioned earlier:

* Pinhole model difference: each camera can't be adequately represented as a pinhole camera, even if the theoretical assembly process is without any assembly errors. This mostly stems from the lens distortion that is introduced in each lens that is not represented in a basic pinhole model. In a nutshell, lens distortion is when a lens produces curved lines where straight lines should be. The two most common types of lens distortion are barrel distortion and pincushion distortion as shown in Figure ‎1‑8.

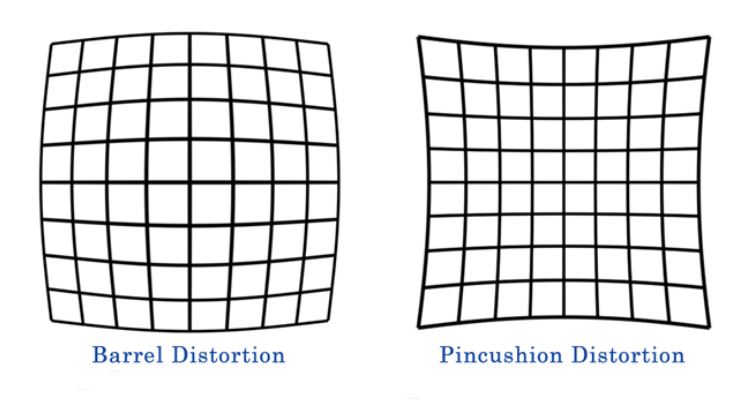


Figure ‎1‑8 - The two common lens distortion types illustrated. Taken from <https://clickitupanotch.com/lens-distortion/>

* Camera differences: Our stereo setup consists of a wide camera along a tele camera. As we've seen the theory for stereo photography is based on the same camera for each of the positions, so the difference between the cameras causes the results to vary from the theory. The module differences consist of different focal length between the cameras, different depth of field (meaning that some areas that are defocused in one camera might be focused in the other), and misaligned optical centers.
* Auto-Focus mechanism – The cameras have AF, which means that the distance from module to sensor changes for different focusing distances. For cameras with different focal lengths, this yields different focus shift for the same depth (meaning that the change in sensor to lens distance of the modules is not the same for different depths) and introduces scale changes.
* Non canonical setup: Since our works aims to use the disparity to depth conversion, the fact that the modules are not in the canonical form inhibits the use of the formulas we have seen in ‎1.1.2.

### Calibration process

In order to handle the non-canonical setup we use a stereo camera calibration algorithm. The camera calibration should output the rotation and translation between the two cameras along with their intrinsic parameters, so that we can use a rectification process to convert the normal images into the canonical form (more on this in ‎1.3.3).

Our calibration process consists of taking an image of a chart consisting of 3 flat chessboard planes combined together. The different planes corners are extracted using the known chart size, and the fact that it's flat. We then employ a gradient descent procedure to first determine the intrinsic parameters (and distortion) for each camera (fitting planes with equidistance points to each chart) separately and then determine the extrinsic parameters between the two cameras jointly. The gradient descent optimization goal is to find the best parameters for the projection of points from one camera to each plane in the first part and from one camera to the other in the second.

### Rectification process

After acquiring the parameters of the cameras, we apply them on captured images in order to convert them into the canonical form. Due to runtime and image viewing constraints, our rectification method only operates on the tele image to match it to the wide image as is. An interpolation map is created in order to use back projection of the tele into a rectified tele image, with fixed orientation and modified radial distortion so that it will fit the wide image's radial distortion. This is done under the assumptions that the wide's distortion in the shared FOV is minimal to non-existent and that the translation in the non-disparity axes are negligible. The image can be further downscaled by the constant zoom factor (the difference in focal length between the wide and tele) and the wide can be cropped in the relevant area (a parameter from the calibration, correct up to the difference in disparity mostly) to produce two images in the same scale with the same field of view more or less.

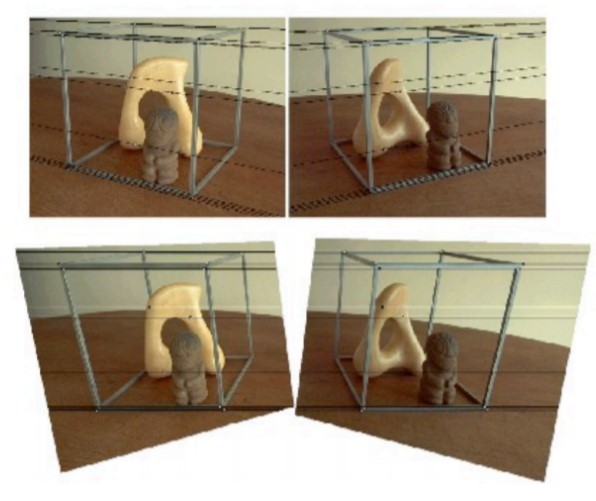


Figure ‎1‑9 - Generic rectification example. Taken from <https://web.stanford.edu/class/cs231a/course_notes/03-epipolar-geometry.pdf>

### From theory to reality

The assumptions above match a theoretical setup. The system we have can be closely modeled using the pinhole camera model, but suffers from inaccuracy with the disparity to depth conversion as follows:

* Compactness and small baseline: Since mobile phones are usually compact, and the baseline between both cameras is required to be quite small (in order for the tele to always be as close to the center of the wide as possible, and always in the FOV of the wide), it makes the conversion between disparity and depth quite sensitive, and requires sub-pixel disparity accuracy for any distance that might be of interest (this can be seen from equation 4).
* The calibration procedure we use, while sufficient for our current applications, does not achieve sub pixel accuracy for the epipolar lines being parallel. This is due to several reasons:
  + Optimization goal – the optimization goal is to project points from one camera to another, and not make the epipolar lines parallel.
  + Chart – our calibration chart is a 3D chart in order to make the calibration process faster (rather than taking several images of the same chart in different orientations).   
    The 3D chart results in two main issues:
    - The chart has inherent depth. This creates a different focus for different sections of the image.
    - Each of the chart's planes, covers only a part of the image, as opposed to taking several images of a full flat chart at different orientations.

Furthermore, one of the main assumptions of the calibration is that each chart's plane is completely flat, and since they can be a bit deformed, this assumption might not be entirely accurate.

* + Focus position – The cameras used have AF capabilities. The calibration is done at a specific focus position, and since we can't extract a focus position report from the device reliably, it can't be adapted to any other position.
  + Temperature – The calibration does not take into account any heating effect that might influence the device.
  + Distortion – The lens distortion of the module in some cases is a bit too difficult for the traditional radial distortion coefficients to handle.
* Our rectification process, again being suitable for most of our other applications, does not achieve sub-pixel accuracy. This is due to several reasons:
  + Only rectifying the tele image to match the wide point of view, while it has its benefits, leads to us not canceling the distortion of either camera. Therefore, we are left with distortion in both cameras. Moreover, even if we do manage to fit the tele's distortion to the wide's distortion in the shared FOV (it is close but not entirely non-existent in that part of the wide image), there is still the issue that the same object in both images is at a different radius from the camera center even in the most optimal case due to the existence of disparity, and therefore even if the distortion coefficients are the same, the distortion of each corresponding pixel might not be the same. This leads to more errors that are not accounted for.
  + The non-disparity axes translations are small, therefore can be ignored for most applications. However, converting disparity to depth with such a small baseline requires sub-pixel accuracy. Therefore the negligible translation still has an effect on the results.
* Focus: Non fixed focus cameras allow for focus changes between different images that cause variations between them. So, since we don't have a reliable focus position report, for the same camera and same scene we can get slightly different images.
* Orientation: Gravity affects differently on the assembled dual camera module, thus holding the device in different orientations produces different disparity measurements for the same object.

We elaborate more on these issues and explore them more thoroughly in the main chapter of our work, chapter 3.

## Our research

### Basic motivation

One of the main motivations to extract actual depth on the mobile phone camera is to enable the usage of augmented reality applications. In augmented reality, you need the actual depth of each area in order to plant an image or an object into the scene in the right scale.

Without additional hardware (TOF sensor for instance) the current stereo cameras in the mobile phones are insufficient for extracting depth from any random scene due to the limitations we mentioned above (a workaround is available by using the inner phone sensors, such as gyroscope or accelerometer, and structure from motion algorithms, but that's irrelevant to our work here).

### Research question

Following the analysis in the previous section we follow these questions in our research:

* What are the effects of the issues defined above on the found disparity from a mobile stereo camera setup?
* Can we find fixes for the issues above that, combined with an online learning scheme and various computer vision tools, will allow us to get a more accurate assessment of the true depth of an object from a mobile stereo camera setup?

# Experiments setup details

## Modules used

* Dual module without folded optics, wide + tele. Tele is equivalent to zoom x2.
* Dual module with folded optics, wide + tele. Tele is equivalent to zoom x3.

## Charts used

* Verification chart:   
  as we mentioned in the previous section, we use a chessboard chart to extract features in a uniform spread across the sensor. The corners between squares are also accurate and repeatable between both images, which is another added benefit. The chart is set to a known distance so that we also check the disparity to depth conversion on it in a per point basis if needed.

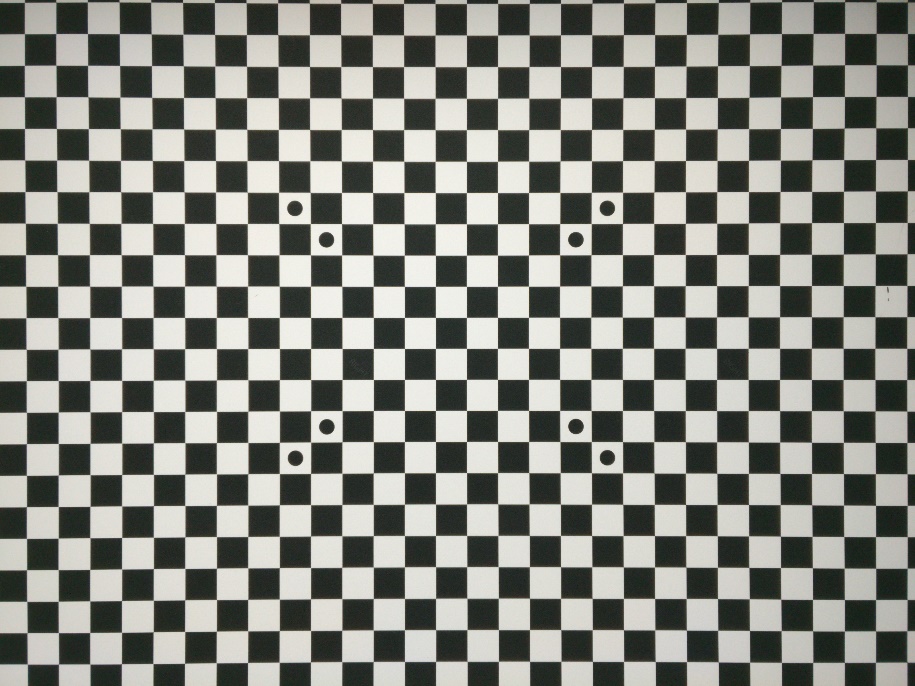


Figure ‎2‑1 - Verification chart used in the experiments.

* Face chart:   
  For online learning, we are measuring facial features. The interpupillary distance - IPD, has a known average (over large population). We constructed a face chart, where all the faces in the image has fixed IPD of 6.3[cm] in order to simulate the collection of face data in different location on the sensor without being sensitive to variation in face sizes. The faces are spread uniformly as possible. More information regarding the IPD is addressed in chapter 3.



Figure ‎2‑2 - Face chart used in the experiments. Faces have uniform IPD of 6.3[cm].

## Physical setup and conditions

Our experiments were held under certain conditions. In order to make chapter 3 clearer, we explain what each condition means according to our terminology here.

The conditions are as follows:

1. Shooting mode – handheld or on tripod
2. Temperature
3. Scene type
4. Phone orientation

Where:

* Shooting mode: The images were taken in one of two shooting modes, tripod or handheld. Tripod shooting mode means the cameras were placed on a tripod during the entire experiment and handheld means they were held by hand, so some movement is expected (though not too much). The latter is similar to the standard shooting mode when using a smartphone.
* Temperature: In chapter 3 we show that the temperature of the device has an effect on the results. Therefor we tested two shooting modes in regards to temperature. Either that the device was pre-heated - the application was left running for several minutes before capture starts, or not - the experiment took place without any specific prior application run time.
* Scene type: Two scene types were examined - flat scenes or depth scenes. Flat scenes should have only one depth in them (i.e. a chart at 2m) and depth scenes should have objects at several different depths.
* Phone orientation: Gravity has an effect on the results. We tested two orientations – normal landscape or flipped by 180 degrees (also landscape). We didn't analyze portrait mode images.

## Theoretical setup

To examine our theories, a theoretical work environment was constructed to allow isolation of outside influences on the results. In this environment we simulated several charts at different depths by randomly choosing several rectangles and their respective depths. The initial scene is simulated in the tele image and then the image coordinates are converted to world coordinates according to the generated depths and locations. The world coordinates are then converted to the wide image coordinates. All coordinate conversions were done using the calibration parameters of the specific module we are simulating. The scene was simulated across multiple frames adding handheld movements along with a random scale to simulate handheld refocuses.

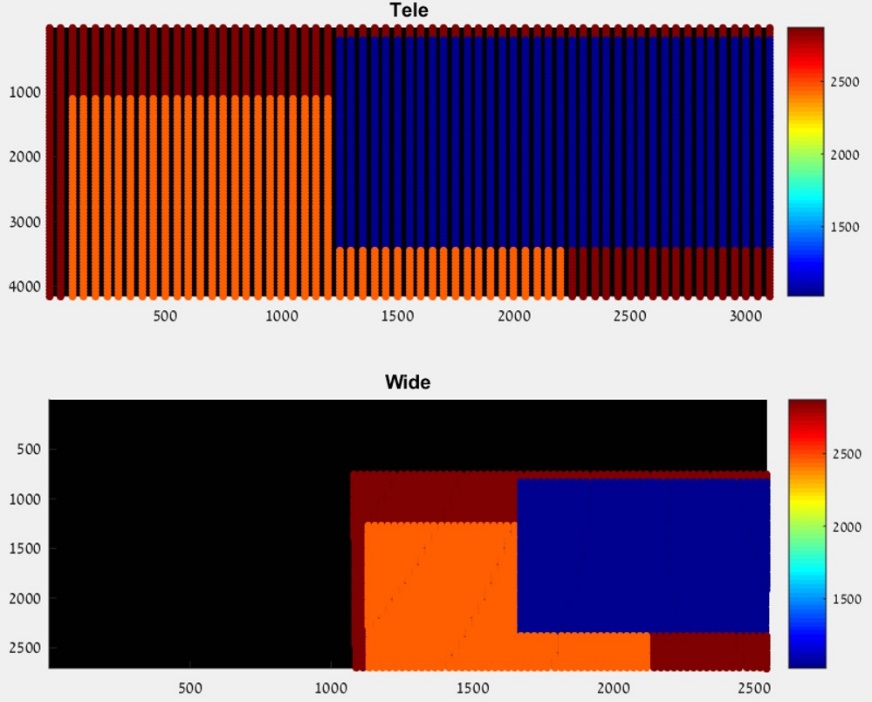


Figure ‎2‑3 - Simulated chart depth scene

# Disparity issues investigations

To start our investigations we checked a simple flat chessboard chart scene (see chapter 2 for details). The analysis steps:

1. Extract chessboard corners (as described in chapter 1) for both wide and tele images.
2. Rectify the found corners.
3. Calculate the disparity between matching corners.
4. Using the naïve disparity to depth formula (from section 1.1.2) given the extrinsic parameters of our setup from the calibration process, calculate the theoretical depth for each corner.
5. Calculate the relative error using.
6. Repeat the process multiple times (multiple captures) to evaluate the repeatability of the results.

In Figure ‎3‑1, we can see the results of a tripod, flat scene, chessboard chart experiment done with a heated folded module. The results show one aspect of our problem, using the naïve stereo theory leads to a large error in estimating the depth from our mobile stereo cameras.

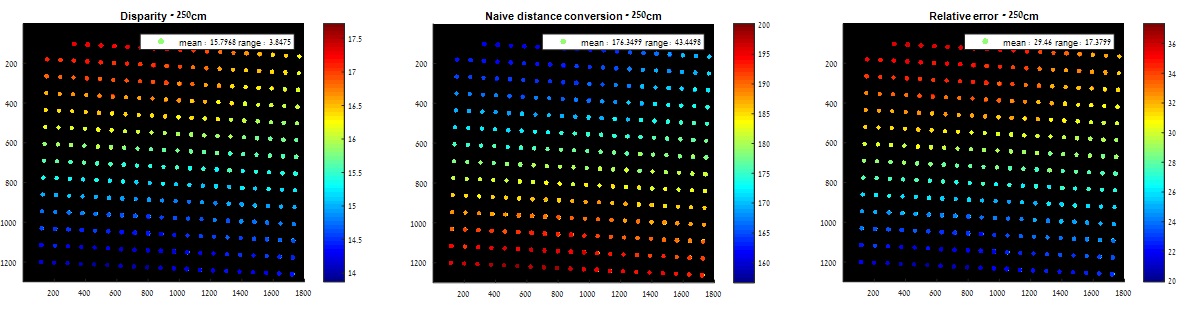


Figure ‎3‑1 chessboard corners disparity and depth estimation using naïve stereo theory. Left image is in [pixels], middle image is in [cm] and right is in [%].

The repeatability tests show poor repeatability. This means that for the same scene, same cameras and same camera location, we get different disparity values. So even if the disparity value was correct for one attempt, the others wouldn't be correct and we would have no way to tell the difference, since for us they are equivalent.

Figure ‎3‑2 shows the results of a test with the following conditions: folded, tripod scene for a chessboard chart and face chart at the same distance. We can see that the disparity results from the two charts are extremely different, and we can't associate the disparity from the face chart to that from the chessboard chart.

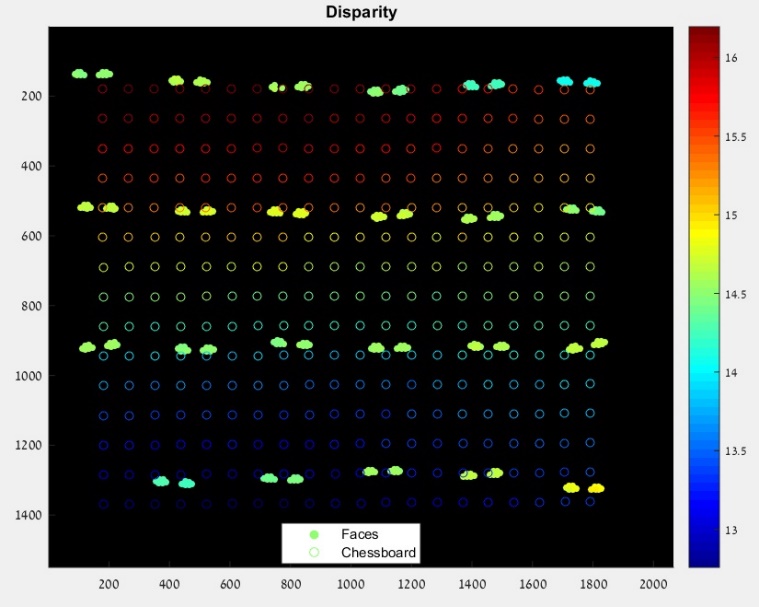


Figure ‎3‑2 - Disparity of faces and chessboard at the same distance

The following sub sections are our investigation into these two issues, and our fix attempts.

## Heating

We tested the heating effect on our results. For that we used a tripod and focus lock. We took images over 10 minutes and compared the disparity values over time. We noticed that, for a given module, we get a "drop" in the disparity values as a function of time. Figure ‎3‑3 shows the results of a tripod, flat scene, chessboard experiment done with a non-folded module with an attached temperature sensor directly to the camera modules. In this experiment we clearly see the connection between the temperature and the disparity. We see that the change is more drastic at lower temperature and reaches a saturation zone (almost) at the higher temperatures. This is demonstrated by the density of points (each one was taken at the same time interval). This can be caused by either:

1. The focus controller being dependent on voltages and as it heats up the voltages it sees change,
2. The lenses inside the lens barrel are not made of glass, so their optical properties can vary slightly with temperature changes
3. Other mechanical reasons we didn't think about.

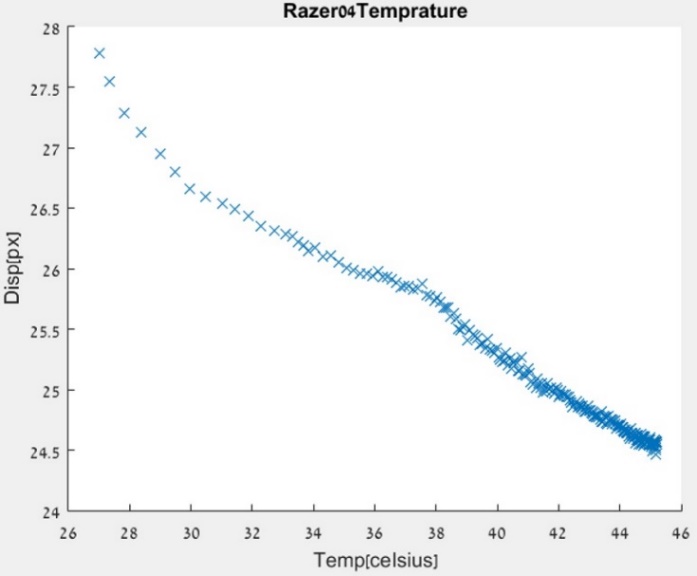


Figure ‎3‑3 - Disparirty change as a function of measured module temperature

Figure ‎3‑4 demonstrates the results of a tripod, flat scene, chessboard experiment done with the folded module at several distances. The conclusions we draw from these figures:

* The heating exists in different modules as well (Figure ‎3‑3 is for a non-folded module and Figure ‎3‑4 for a folded module).
* The heating effect has a saturation zone, meaning that after some time that the camera has been active, the heating no longer changes the disparity significantly.
* The heating effect causes an offset in each camera, but it is different between the two cameras, which is the reason for the disparity change.
* The change does not seem to be monotonic with the distance to the chart.

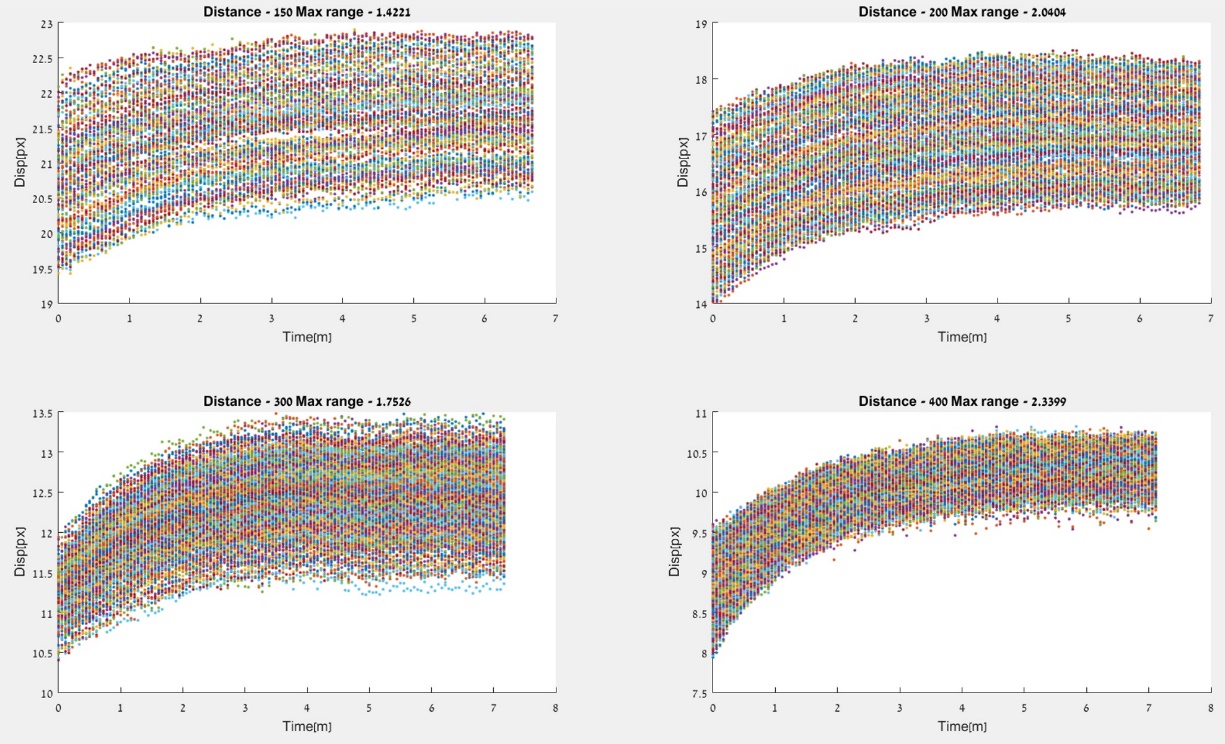


Figure ‎3‑4 - Disparity change as a function of time. Each color tracks a specific chart point.

In Table 1 we see the effect of the disparity range on the relative error, meaning that this is the error added, only from heating, to any estimation of the depth we get if we don't address this problem.

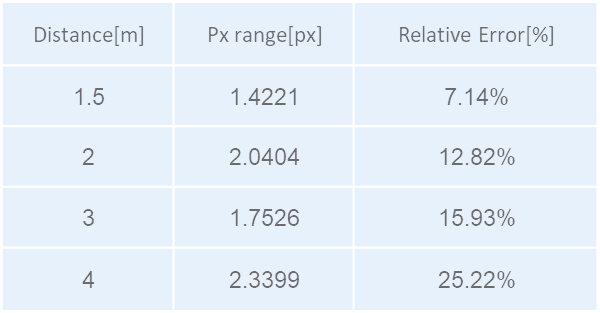


Table 1 - Disparity range due to heating effect on relative error

**Fix options:**

1. Get additional temperature data from the module. We investigated a bit and found out that the controller should have an internal temperature measurement (not absolute, but relative to a set point), using this measurement then allows the extraction of where we are on the temperature plot by comparing the temperature change over time. Unfortunately, this measurement is not supported in our setup.
2. With the lack of success in getting the temperature information, we resorted to limiting our depth-from-disparity conversion to the temperature saturation zone in each module. For example, for the folded module it takes around 3 minutes in order to reach the saturation zone, so the experiments are held after the cameras are active for 3 minutes at least.

## Distortion

During our investigations we noticed that there is some radial error pattern existing in the disparity. This could be due to radial distortions existing in the images even after they have been rectified. A few possible reasons for this:

1. The rectification method used converts the tele image’s lens distortion into the wide image’s lens distortion (as explained in ‎1.3.3). This procedure doesn't remove the distortion from both images. This causes an issue, as some items that appear in some radius X in the wide image, are by definition not in the same radius in the tele image even for fully canonical cameras since and (apart from objects that are far away enough from the cameras so that the disparity is ~0). This isn't a major issue usually, as the difference in radius isn't large, but due to our very small baseline we need close to sub pixel accuracy to get a "good" relative error for any distance above ~50cm.
2. The representation of the radial distortion is done only with the even degrees for a radial distortion polynomial. This is common practice in most calibration schemes we know of. This representation of the distortion could be insufficient for some modules.
3. Focusing – focus change is achieved by moving the barrel away/towards the sensor. This, by definition, changes the distortion in the image from the distortion found in the calibration process that was applied in a specific distance.

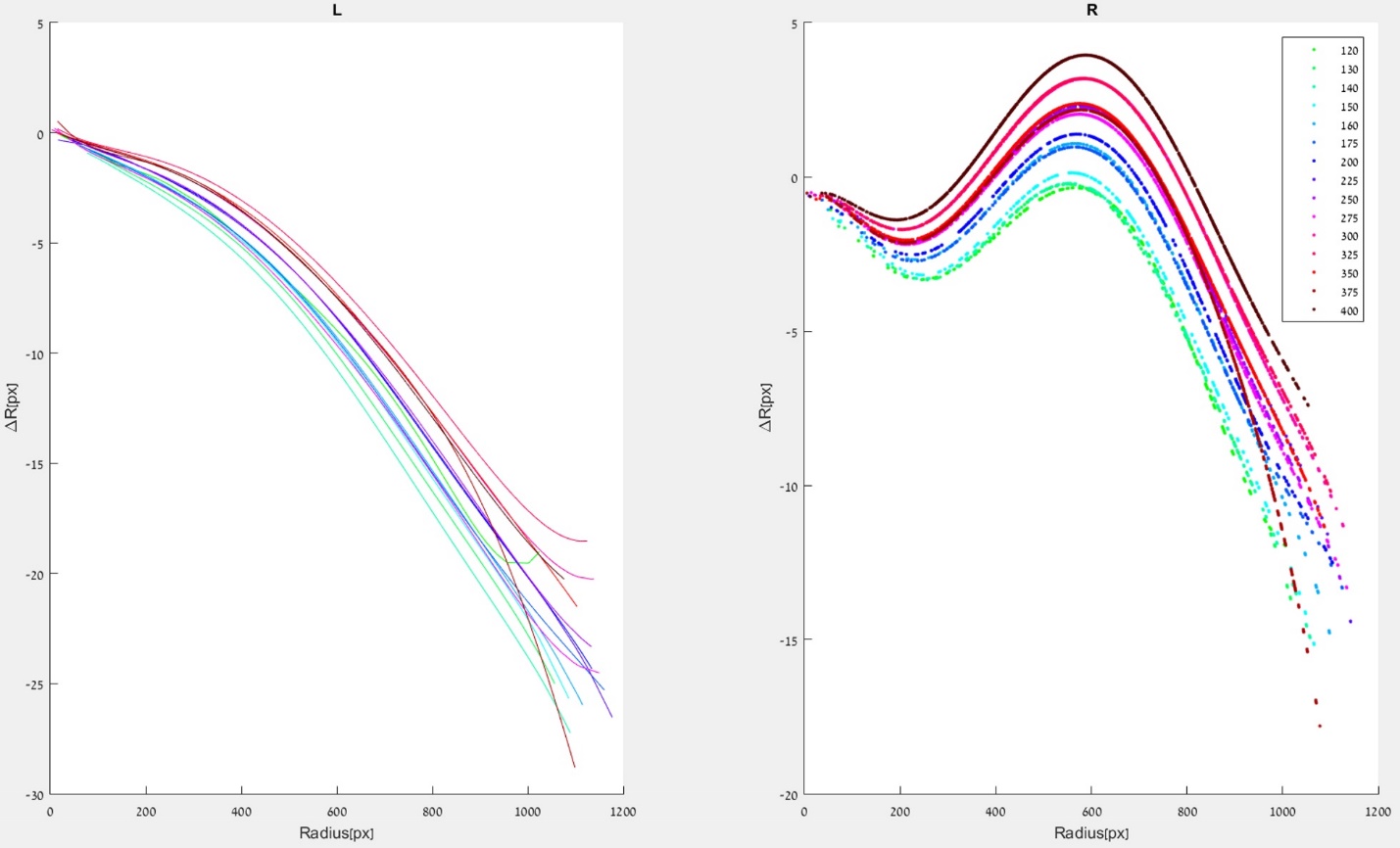


Figure ‎3‑5- Found distortion from flat scene for each focus distance

Figure ‎3‑5 shows the found lens distortion for a set of images taken at several distances on a tripod, flat scene, chessboard experiment done with heated non-folded module. We can see that the distortion indeed changes with the focus distance.

**Fix options:**

1. Use extra calibration steps to determine the lens distortion at each focus distance and apply the fix on the given image. This method is less desirable as it adds phases to the calibration process, which is not an optimal solution for mass production.
2. Switching a rectification mode that cancels the distortion for both cameras. This seemed to have helped, but not by much.
3. Adding the odd degrees to the radial distortion polynomial and by this enabling a more complex model for the distortion that better represents the real distortion.

Figure ‎3‑6 shows the found lens distortion for a set of images done with a heated non-folded module at several distances taken on a tripod, flat scene, chessboard experiment. The lens distortion was found by first finding the homography between the two cameras, fixing the corners according to it, and matching what remain with a radial polynomial that represents the distortion to improve the matchings. The results seemed to have improved, so the full degree polynomial lens distortion is closer to the one found from the scene than the normal polynomial, but not enough.

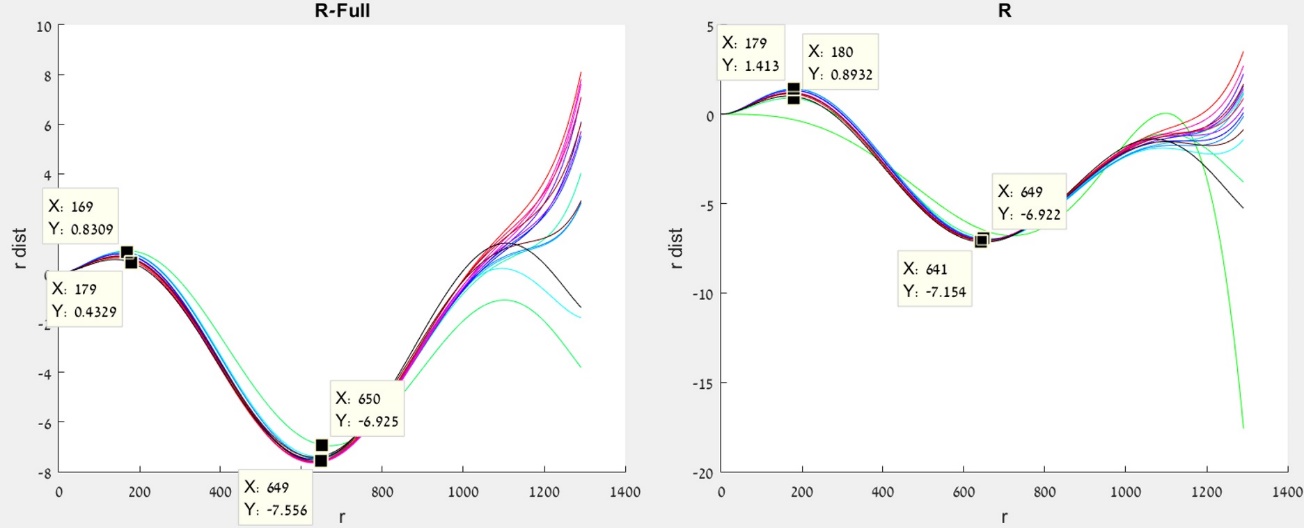


Figure ‎3‑6 - Lens distortion vs radius for several distances. Bright green is the found polynomial from the calibration. Right image is the normal polynomial and the left is the polynomial with the odd degrees added. Units of both plots and both axes are in [pixels].

1. Non-disparity axis constraint - after both the "easier" fixes above helped, but not enough, we looked for another way to fix the residual distortion. In order to do that, we decided to use the non-disparity axis in order to find a correction for the distortion. This follows from the fact that after the rectification with the above modifications, the non-disparity axis error should be close to 0 and mostly the same for any depth (since there is only a very small translation in that direction). This has the benefit of not adding phases to the calibration process and not being dependent on the scene type (flat or depth). All the example figures for the non-disparity axis fix were taken on a tripod, flat scene, chessboard chart with a heated non-folded module.

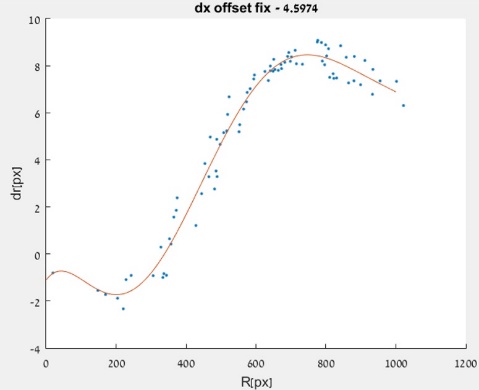
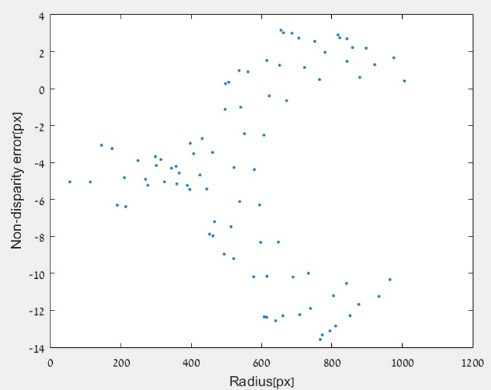


Figure ‎3‑7 - From left to right, Non disparity error, radial error component and radial error component with optimal non-disparity offset

We examined the non-disparity axis error as a function of image radius, and then we converted the error to radial error. The radial error is calculated by using each corner's angle with the optical center (noted as dr) and only taking the radial direction and discarding the tangential. The resulting graph (middle of Figure ‎3‑7) did not look to have a polynomial radial trend. To expose the radial trend we added an offset in the non-disparity axis. The result reveals a polynomial connection. Therefore, we used a minimization function that minimizes the difference from the best polynomial fit of dr to the data, to find the best offset and got the plot that we can see in the final graph of Figure ‎3‑7.

As we can see, we can definitely use the non-disparity error in order to fix the residual distortion using the method defined above. Theoretically, the suggested method will apply for both flat and depth scene, as the depth should not affect the non-disparity axis error after rectification. From our experiments, we showed that this indeed seemed to be the case for our non-folded model, however, for our folded model, due to its mechanical properties, there is a dependence on depth even for the non-disparity error due to Z axis translation (more on this in ‎3.3).

## Non baseline translation

Our usual rectification method, as mentioned in ‎1.3.3, removes the rotation between both modules but it does nothing to help with the translation. In the case that the only translation between the modules is the baseline, this is obviously not an issue, but when we are dealing with the real world scenario and especially with mobile cameras, the tolerances for shifts in the non-baseline axis are small but not negligible. For most applications this is not a major issue, since it doesn't have a large impact. However, in our case, since the baseline itself is small and the required accuracy is high, this could potentially be an issue and we wanted to investigate its impact. In Figure ‎3‑8 we can see an illustration of this problem.

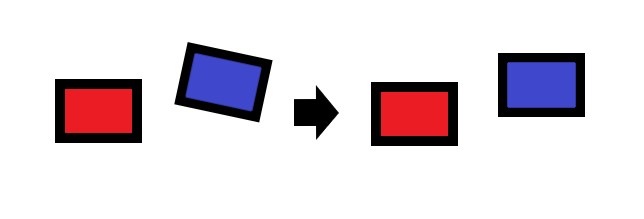


Figure ‎3‑8 - Illustration of the y-axis translation remaining after rectification

Theoretically, translation in the non-baseline axis as illustrated above (i.e. not the Z direction that is towards the object) should simply add disparity in that axis on top of the usual disparity. This could raise issues in several areas, such as feature matching. Also, if we still have some residual distortion, it could add to issue (a) we mentioned in ‎3.2. It could also inhibit our non-disparity axis fix that we explained in ‎3.2 since it undercuts our underlying assumption that the non-disparity axis should be independent of the object depth. The translation in the Z axis should, theoretically, cause objects at different depths to have different scales, which is quite a major issue. When combined with additional scale issues that will be addressed in ‎3.5 the problem increases

**Fix options:**

To fix the translation in the non-disparity axis (not in the Z direction) we rectify the images so that they are truly canonical. In order to do that we need to calculate the added rotations from the translation found by the calibration process and rectify the images accordingly. Figure ‎3‑9 shows an illustration of this rectification result.

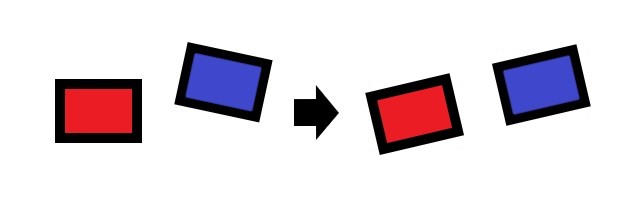


Figure ‎3‑9 - Illustration of a truly canonical rectification result.

Figure ‎3‑10, show the result of applying the fix on a tripod, dual depth scene composed of two chessboard charts at different depths on a heated folded module where the baseline is in the Y axis.

We can see that before the fix was applied the non-disparity axis error seemed to be different between both charts. We didn't expect the value to be zero or even constant (due to the many other issues we elaborate on in this chapter), but if the non-disparity axis was indeed negligible, we expected the trend to be the same on both chart in the same section of the sensor, meaning independent of object depth. Since this is clearly not the case, we can assume that there is indeed an impact of the non-disparity axis translation.

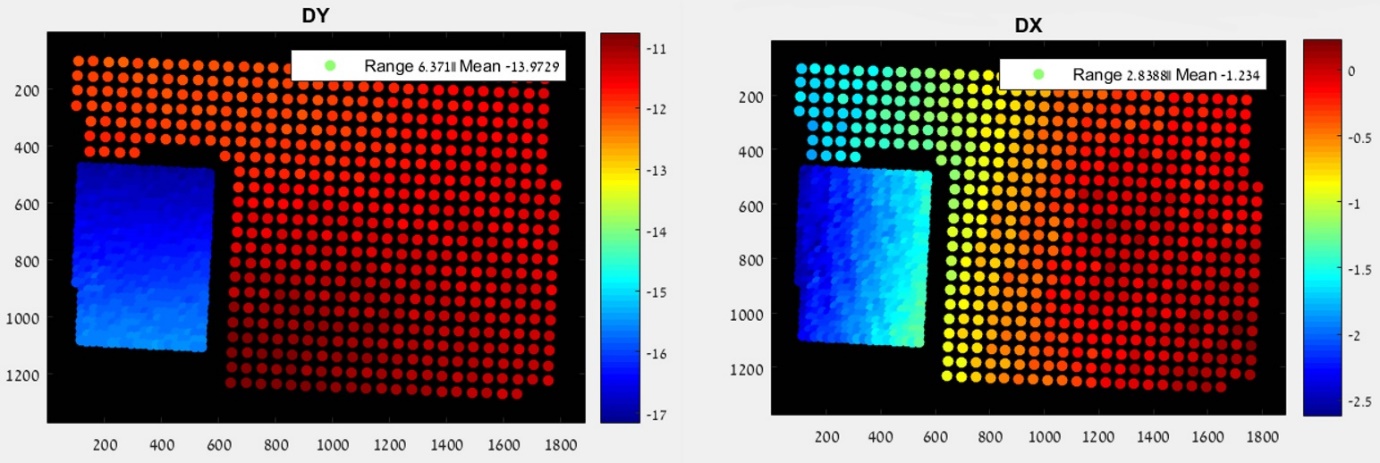
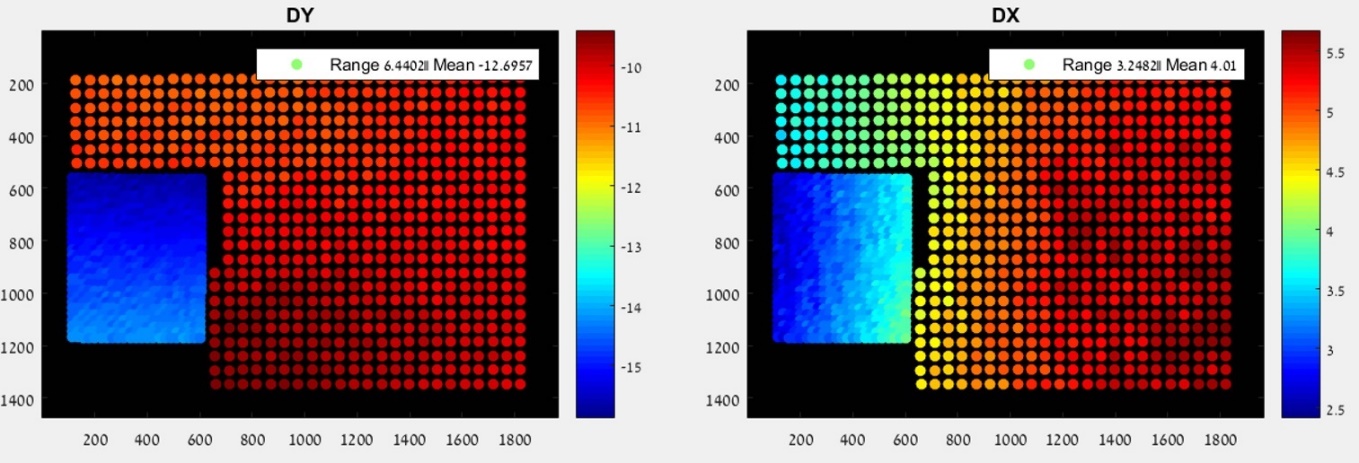


Figure ‎3‑10 - Disparity (DY) and non-dispartiy error (DX) for a dual depth scene, without (first row) and with (second row) the tranlsation fix.

The results after the fix seem slightly better, the non-disparity axis seem to continue with its trend even to the different chart (the colors are a bit more similar and the range is a bit smaller as well). This indicates that the non-disparity translation does play a certain factor, and if we want to use the non-disparity axis fix we mentioned in ‎3.2, we will need to apply the fix mentioned here before trying to use it.

However, the results are still not perfect and there is still a difference in values for the non-disparity axis between different depths. This could be caused by inaccuracy in the calibration process, since the non-disparity axis translation is very small and our accuracy in estimating translation isn't great. This could also be caused due to the Z direction translation. The Z direction translation is an issue we still don't know how to fix. Since we don't know the depth of each object in the scene, and since our folded model has a significant Z translation due to its mechanical properties, but we can't accurately measure it in the calibration (we have seen this empirically), we still haven't found a fix for this (more about this in ‎3.5 where it plays a major role).

## Device orientation

In order to adapt to a real world use case, where the orientation of the phone could vary, we decided to investigate whether the phone's orientation effects our results.

Figure ‎3‑11 demonstrates the results of two different orientations taken on a tripod, flat scene, chessboard chart with a heated folded module. From the results we can see that the different orientation adds an offset to the disparity. This indicates that we definitely need to address this issue.



Figure ‎3‑11 - Disparity result for different module orientations

**Fix option:**

To address this issue, the gyro information is required while we learn a fix as well as when the fix is applied. This suggests that we can only use fixes learned from the same gyro orientations.

## Focus

One of the main issues we have regarding any variation of a learnable fix to disparity is repeatability, as we explained at the start of this chapter. In order to improve the disparity repeatability of our cameras we investigated the focus's effect on it. The focus process in the mobile cameras is comprised of moving the lens barrel towards and away from the sensor and by doing this we effectively change what distance will be in focus on the sensor. An easy example of this is the effective focal length, which is the distance between the lens barrel and sensor that will focus objects at infinity distance from the camera, as seen in Figure ‎3‑12.

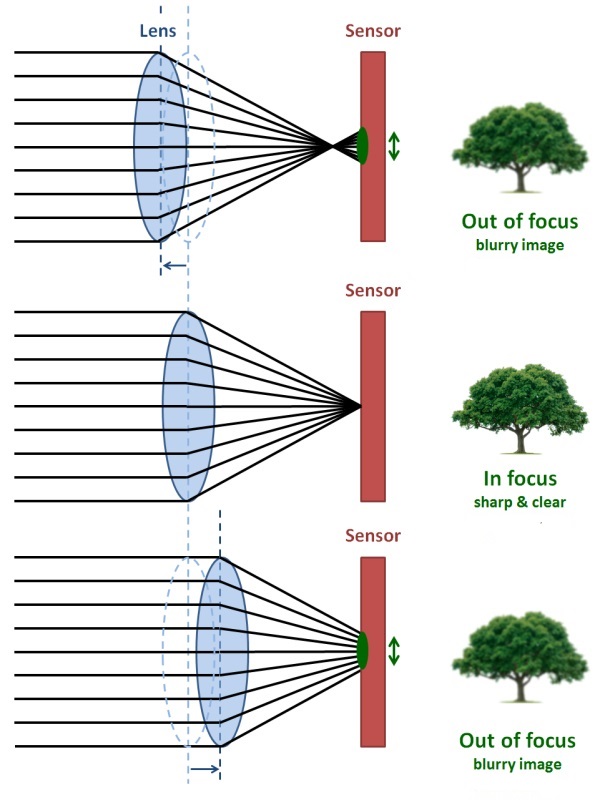


Figure ‎3‑12 - Simplified example of focus distances. Image taken from <https://c.mi.com/thread-904594-1-0.html>

To achieve focus to an object in a certain distance, an estimation process is applied (auto-focus) to try and find the best lens to sensor distance that will give the selected object the best score according to one focus metric or other (a simple example is getting the highest average gradients in the region, since blurry images usually have smaller gradients).

Since focus position estimation is done per image capture, we witness variation in the focus results even when taking an image of the same scene with the same focus object and the same camera distance. Having a different focus positions has direct influence over the disparity. It produces a different scale for each image, which in turn causes objects at the same distance to have different disparity values in two different wide-tele image pairs.

A trivial fix for this problem could be to get an accurate read on the focus position of each camera and either forcing a discriminative set of focus positions for each camera or using the focus position to learn a disparity fix for each one individually (and then when we apply the fix we can choose from the current reported focus position's learned fix). However, in our experiments, we haven't managed to get any useable focus position report from the module.

In Figure ‎3‑13 shows the results of an experiment taken on a tripod, flat scene, chessboard chart with a heated folded module. We extracted and matched the corners as described before and found the scale between each wide-tele image pair. We then checked the following:

1. Scale's dependence on time and on the reported focus position for each camera
2. The reported focus position's dependence on time.

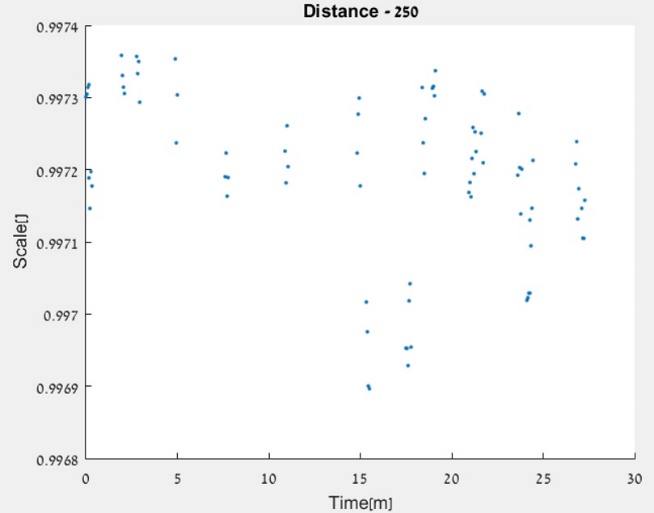
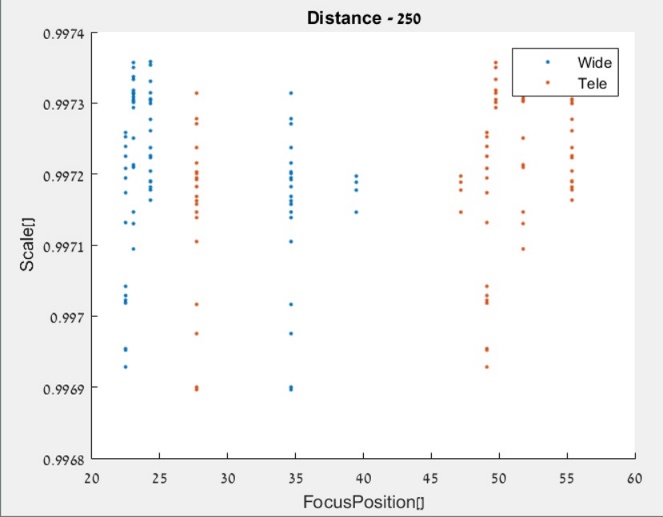
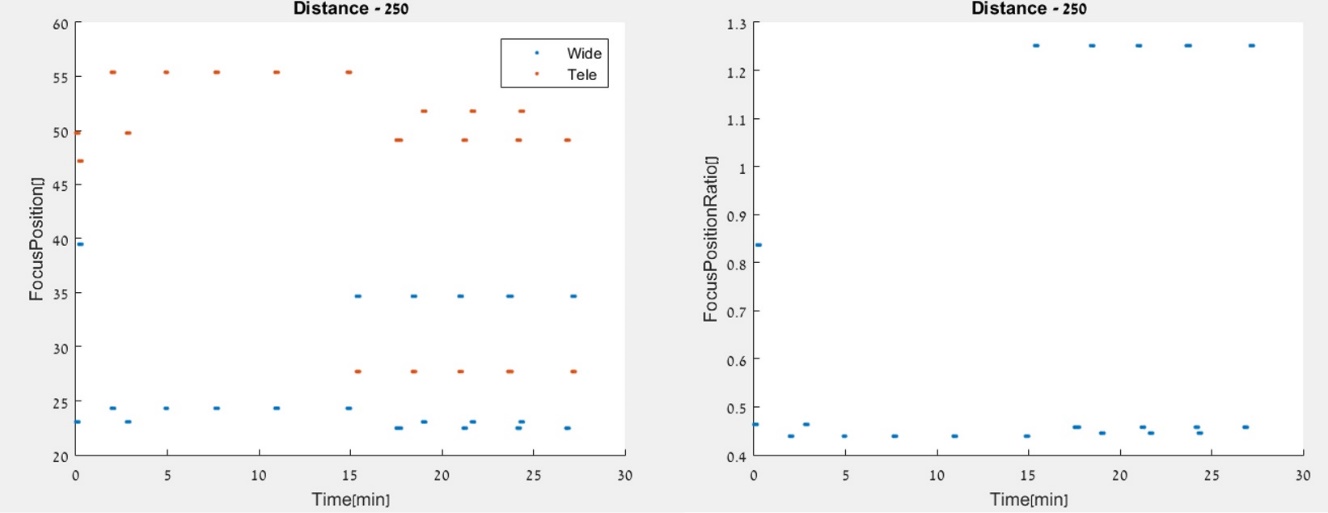


Figure ‎3‑13 - Focus position and scale between wide and tele image pairs analysis. Bottom right shows the division of the wide focus position by the tele (which should indicate scale to some extent)

As we can see, there seem to be no dependence on time for the scale nor the focus position reported (this validates our assumption that the disparity change hits saturation when using a heated module). We can also see that the scale doesn't have any dependence on the reported focus position of neither the wide nor the tele image, not even on the ratio between them (as we would have expected). This is either the result in some bug in the android application we are using to read the focus position or the inherent inaccuracy with the focus position.

This led us to the conclusion that we can't rely on the reported focus position for our modules and that we need to find a different solution. We first divided the issues that the different focusing can create into three separate categories, and we attempted to tackle each one on its own.

### Focus fail

This is the case where one or both cameras have entirely missed the focus distance for the desired object. This case is a major issues for us since extraction of features in blurry images is very unreliable and definitely not sub-pixel accurate, as we require due to our small baseline.

**Fix options:**

1. Define a focus metric of our own or use the score from the existing one in the module. This solution is not feasible since defining a focus metric is a project in its own right, and we can't get access to the focus metric score of the module.
2. Use manual focus - a predetermined focus position that will handle the normal use case for most photos. This way we can ensure that the focus miss will only really occur to items that are very close to the camera, and since their disparity is bigger anyway, the relative error due to localization is smaller, and we can define a range of distances for our depth estimation to work in.

### Wide to Tele focus

In our experiments we noticed that even in a focused pair of images we can see quite a large disparity range for the same object distance for different focus attempts. After some investigation we concluded that this is due to the different focusing of both cameras, i.e. in one case the wide focused a bit closer and the tele a bit farther and vice versa in another case. We visualize this in Figure ‎3‑12, the tele could be focused like the top image (just behind the sensor plane) and the wide could be focused like the bottom (just before the out of focus zone in both images). This leads up to a large disparity range even in the same location on the sensor.

Figure ‎3‑14 shows the results of a tripod, flat scene, face chart experiment done with a heated non-folded module. We can see the landmarks detected and the maximum disparity range per landmark in each cluster over seven focus attempts using auto focus.

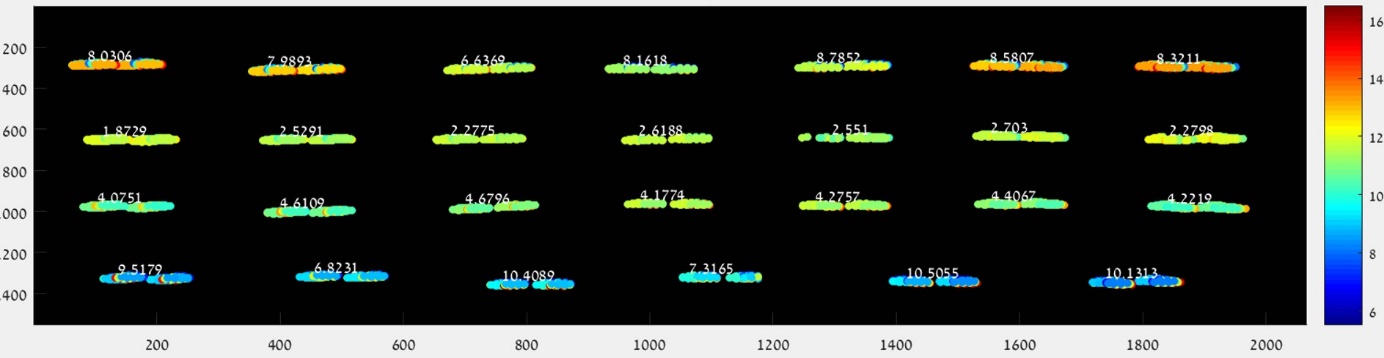


Figure ‎3‑14 - Maximum disprity range per cluster over 7 focus attempts.

As we can see, the range is indeed very high, up to the point that we can't find any disparity to depth fix with this ambiguity (having an object at the same distance with 8 or 16 disparity is too much of a difference).

**Fix options:**

Investigating this problem led us to the conclusion that the wide-tele focus difference introduces scale between the images.

To overcome the scale change, we suggest to rescale the current wide-tele image pair so they will be in the same scale. This solution is not straight forward, since the focus change also changes the radial distortion (both cameras have radial dependencies) so it is hard to separate the two. One way we have found that lets us obtain the scale between the images is to first find the homography between the scenes, then find the residual distortion and only then find the scale after fixing the residual distortion without fixing the homography. This is obviously only applicable to flat scenes, as you can't find a homography for a depth scene when the cameras have translation between them, but using this we fixed the range from maximal of 10 pixels in Figure ‎3‑14 and to maximal of 1.5 pixel range shown in Figure ‎3‑15 (same experiment setup).

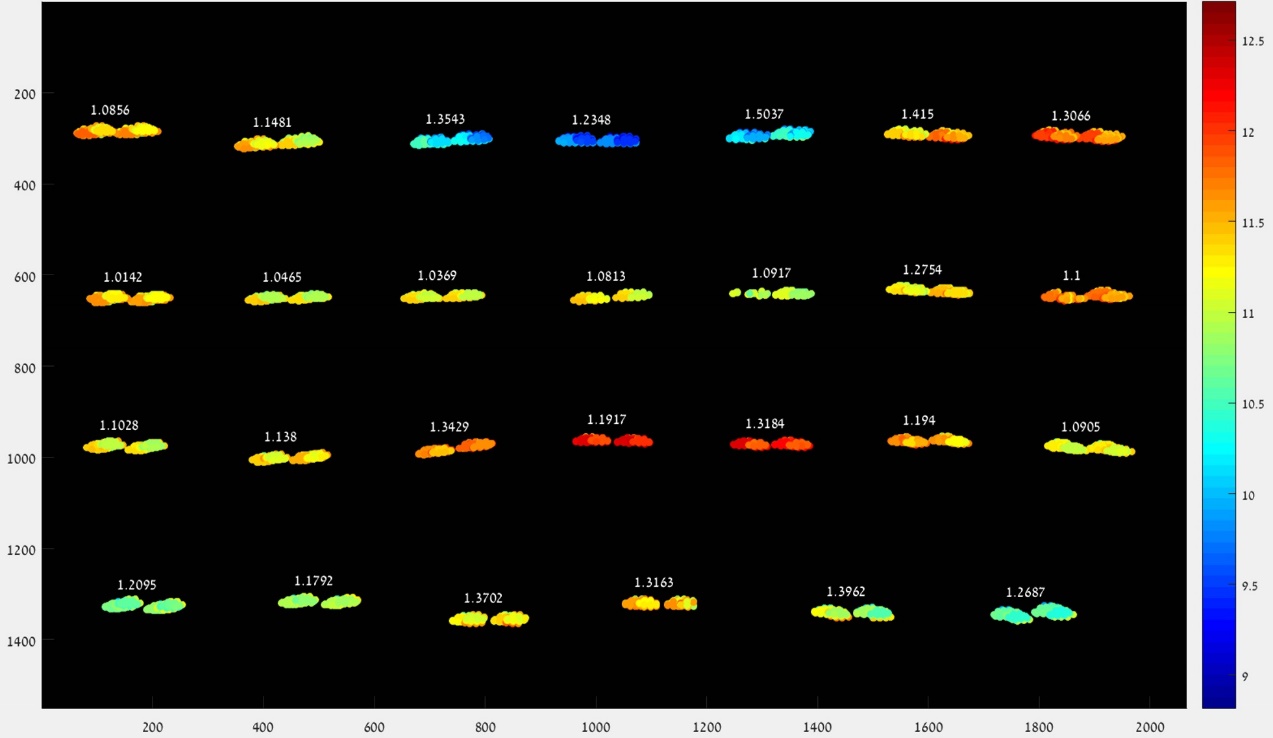


Figure ‎3‑15- Maximum disprity range per cluster over 7 focus attempts after scale correction.

This was done mostly to show the importance and viability of scale correction. At this point we decided to start using the folded module exclusively, since the tele's residual distortion in this module is almost non-existent (probably due to the small overlapping FOV) and it allowed us to tackle one less issue.

All of the following experiments were done with a heated folded module with manual focus position setting (as we suggested above), on a tripod with a flat chessboard scene.

In Figure ‎3‑16, we see the disparity range for each corner over 7 refocus attempts (setting the manual focus to the same values each time). We can see that even though we set the same manual focus values for each attempt, we still get a variation in the disparity for each corner. We found the scale here using an optimization function that is aimed to minimize the non-disparity axis range, this way the offset in that axis doesn't have any real effect and we can theoretically handle depth scenes as well. We see that, while the per corner range decreased only by a little, the chart's disparity range decreased by a lot. This is a favorable result as it would allow us to learn a fix from all the image and apply it to the entire image instead of each section having its own localized fix. Table 2 shows the relative error with learning the fix from only the center of the image and applying it to the rest of the image, before and after the scale correction.

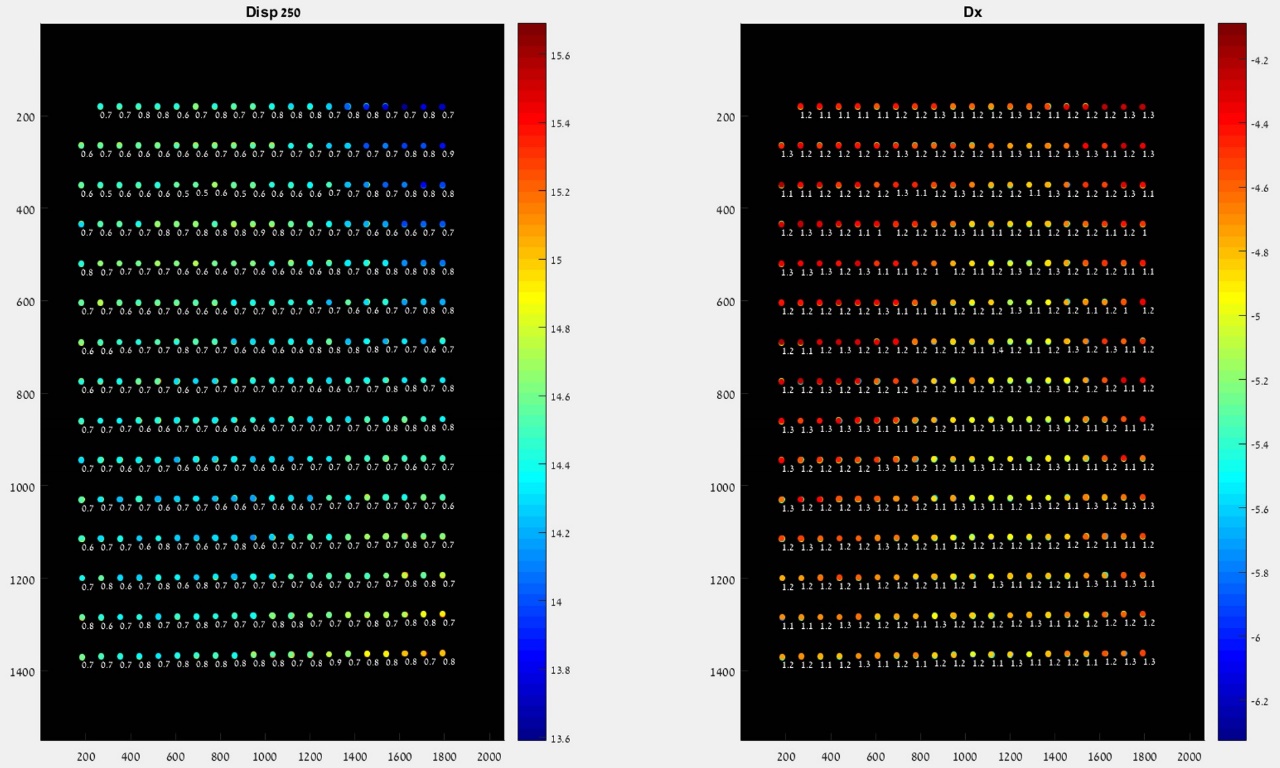
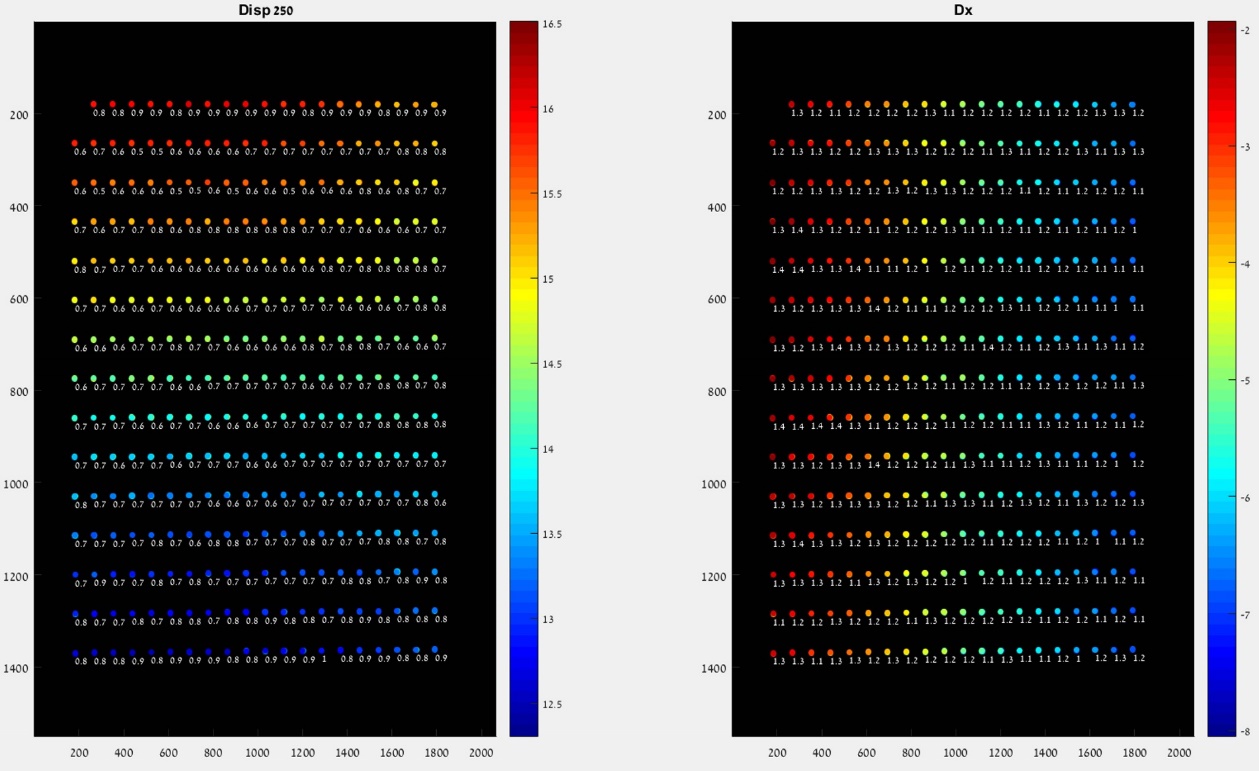


Figure ‎3‑16 – folded module disparity(left) and non-disparity error(right), before (top) and after scale fix (bottom) with per corner range next to each corner

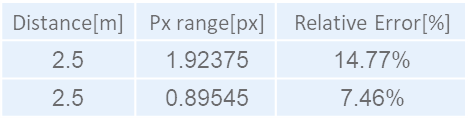


Table 2 - Applying disparity fix learned from image center to the rest of the image results.

Figure ‎3‑17, Figure ‎3‑18 and Figure ‎3‑19 show the results of running the same experiment as for Figure ‎3‑16 but with a dual depth chessboard chart scene and applying the non-baseline translation fix as we described in ‎3.3. In Figure ‎3‑17 and Figure ‎3‑18 we can see the results of taking each chart separately and applying the scale minimization to it, and in Figure ‎3‑19 we can see the result for applying the scale minimization for both charts combined. The scale we got for only the close chart is 1.0005, and for only the far chart we got 1.0023, and the combined scale is 1.0019. As we can see, each distance has its own different scale, and the combined one is probably influenced only by the amount of points each chart has in the image, rather than one true underlying scale (the far one has obviously more points, so it has a greater influence on the combined scale). This is probably caused by the translation in the Z axis as we've mentioned in ‎3.3, which causes each depth to have a different scale factor and this, in turn, makes the task of finding the "correct" scale factor, so that the wide and tele images will be at the same scaling in regards to each other in every image taken, very difficult.

**Fix options:**

We propose to cluster features according to disparity and fix each disparity's scale independently, i.e. fix each depth's scale, and this way we can apply this method to both the learning and the application phases and increase repeatability. This also fixes the Z translation problem along the way. However, for scenes where there is only a limited amount of points for a certain disparity value, or if their spread is limited then the fix could be very biased and incorrect.

We conclude that for flat scenes, or if we have a sufficient spread of the desired object's disparity cluster in the image, we can use the wide to tele scale fix to be able to fix the entire image from a small amount of data. However, for a depth scene, we would probably need a localized wide to tele fix per location (but that requires some further investigations).

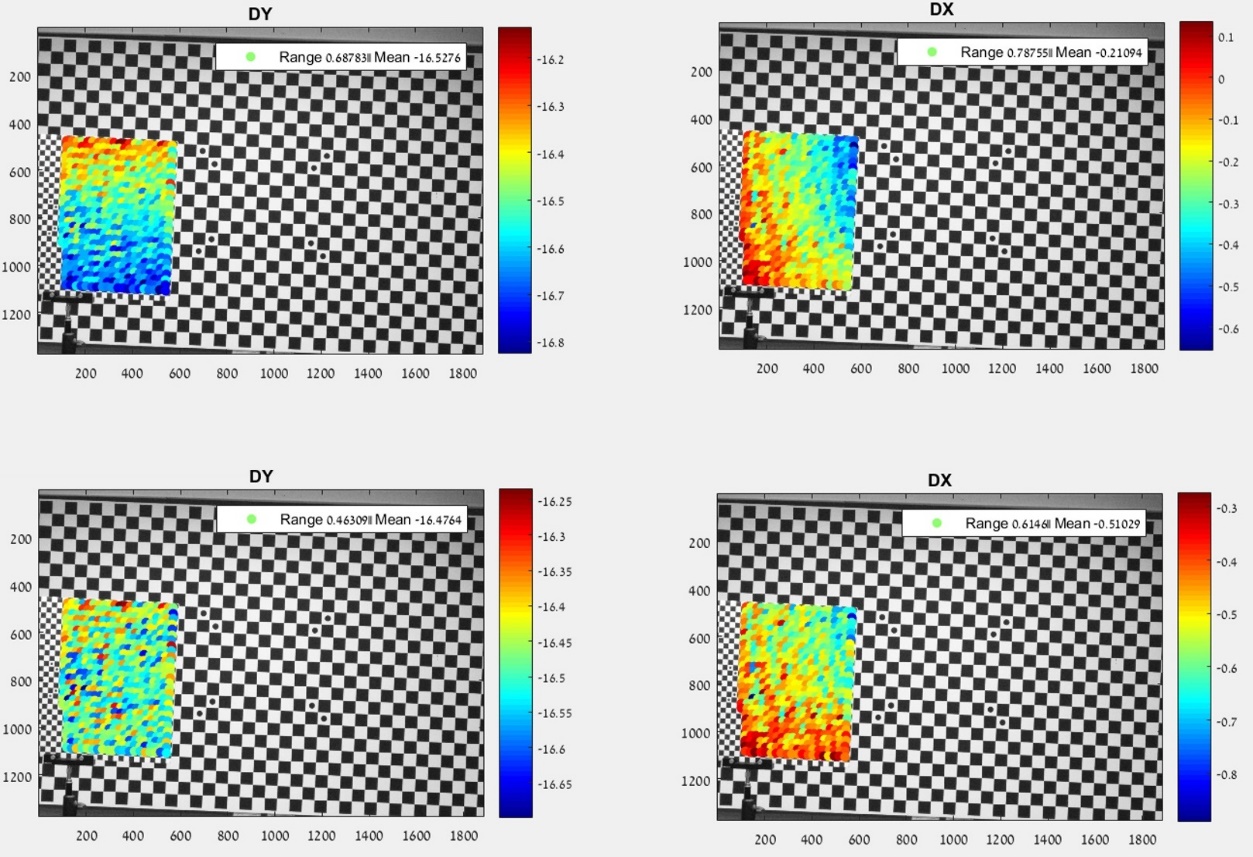


Figure ‎3‑17 - Close chart disparity (DY) and non-disparity error (DX) before (top) and after (bottom) fixing the optimal scale for it (1.0005)

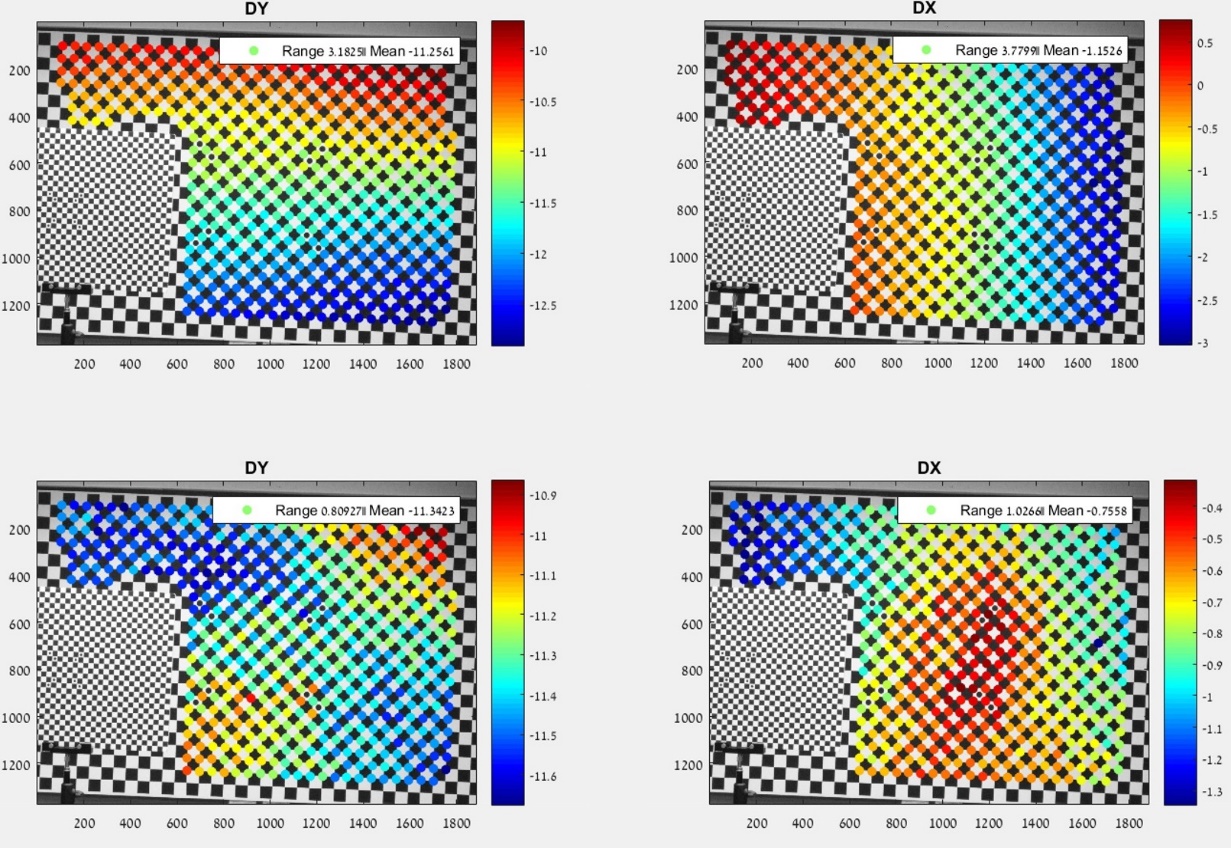


Figure ‎3‑18 - Far chart disparity (DY) and non-disparity error (DX) before (top) and after (bottom) fixing the optimal scale for it (1.0023)

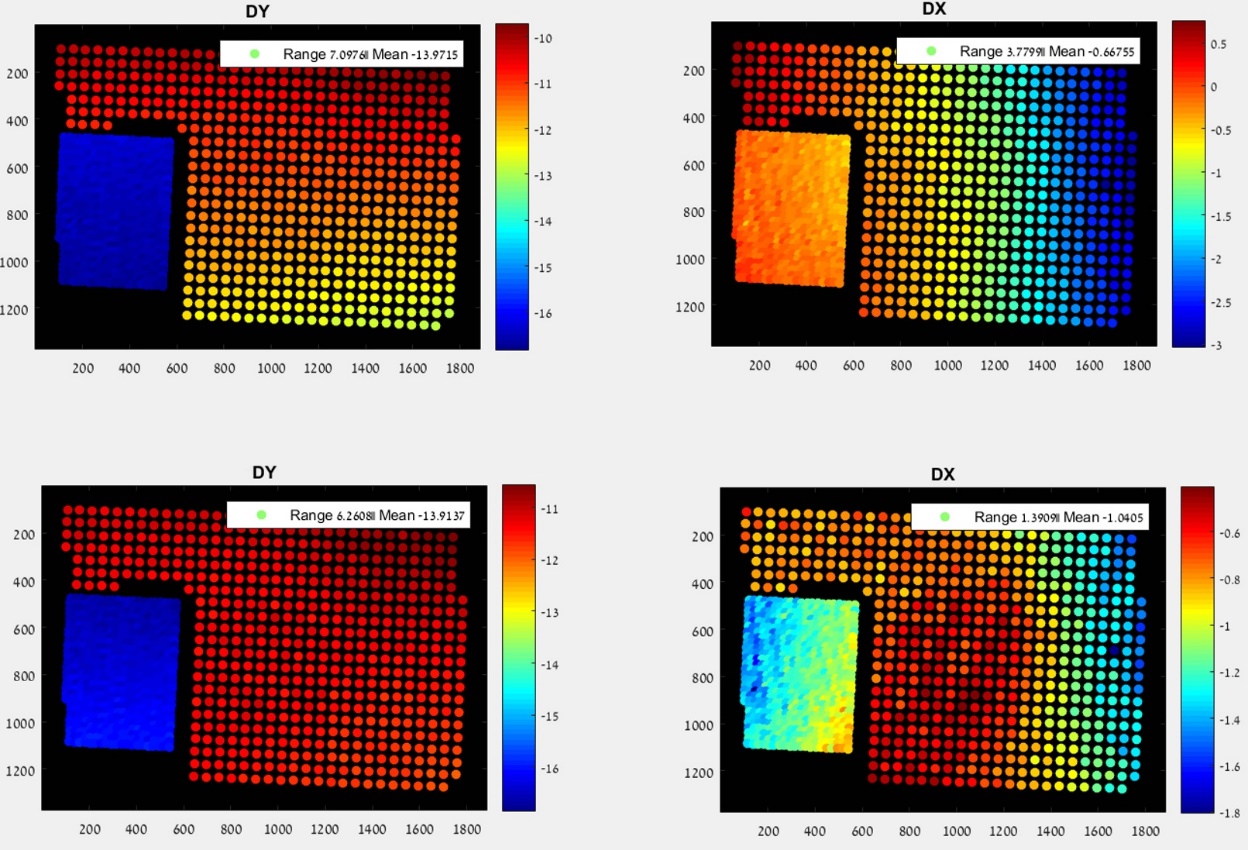


Figure ‎3‑19 -Entire image disparity (DY) and non-disparity error (DX) before (top) and after (bottom) fixing the optimal scale for it (1.0019)

### Same camera focus repeatability

Even if the focus between the wide and tele is fixed, since we don't have any "real world size" in a normal scene, we don't know if the focus is correct for the desired object in the ROI. It could be that one focus attempt chose to focus a bit farther away and the other a bit closer, as we mentioned in the previous section, only in this case assuming that both cameras are in sync and the focus difference is between different captures. Again, we can visualize this in Figure ‎3‑12, the first case we described could be the top image and the latter case could be the bottom. This causes some extra disparity difference for the same object distance in the same sensor location. Figure ‎3‑20 shows the results of an experiment done with a heated folded module with manual focus position setting, on a tripod with a flat chessboard scene. As we can see there is a range in the disparity between each corner location even with a tripod and manual focus setting.

Most of the difference seems to be from different scaling or some offset (since we moved to the folded module, the lens distortion change is insignificant).

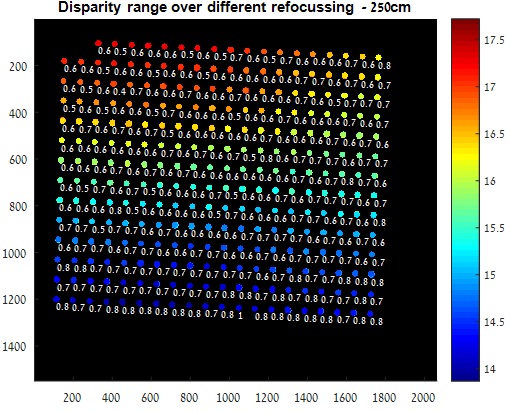


Figure ‎3‑20 - Disparity range for each corner over 7 refocusings.

**Fix options:**

1. Use the reported focus position and learn a disparity fix per focus position. This option isn't viable for now since, as we've shown before, the reported position is very unreliable and could not be trusted.
2. Take several images for several focus attempts of each of the learning phase photos and the application phase photos. Using this stack of images, we can try a few options for fixes:
3. Wide-Tele matching and average disparity - for each focus attempt, extract the features and find their disparity. Then match the features between all the tele images and all the wide images. For each feature, average the disparities found. Figure ‎3‑21 shows the results of the same experiment as described above, with averaging the disparity over 3 and 5 focus attempts (chosen randomly from our set of 7 attempts), and the resulting ranges. In Table 3, we summarized the effect of averaging the disparity on the relative error. We can see, that by averaging 3 focus attempts we can get a decent improvement and by averaging 5 focus attempts, we manage to get a large improvement.

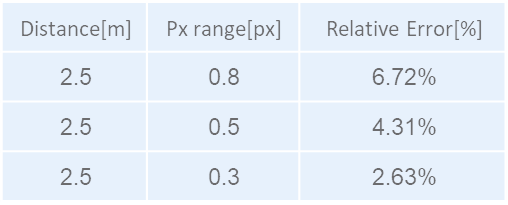


Table 3 - Disparity averaging result summary. Top row is without averaging at all, middle row is with averaging 3 image disparities chosen randomly, bottom row is with averaging 5 image disparities.

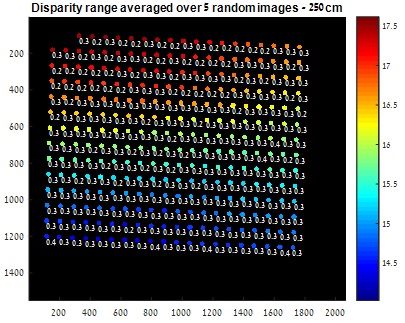
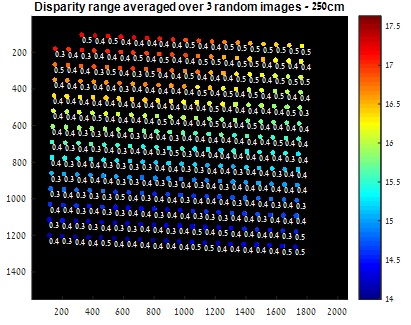


Figure ‎3‑21 - Averaging the disparity over different focus attempt

Theoretically, this seems like a good fix. However, the problem arises when we consider the real world scenario. In the real world scenario, matching a feature across 5 different focus attempts while the cameras moved (even a little) during the refocus attempts is quite difficult. From our handheld experiments we saw that only a small portion of the features in a scene get matched between different focus attempts and most features are dropped since they don't exist in one of the scenes (aren't recognized as features by the extractor), which makes finding features for the specific selected ROI a bit difficult. Moreover, using this would require us to take 3 or 5 refocus attempts for each depth estimation separately (since the cameras can move and the scene can change). Even if we restrict the camera motion between depth estimations, and hold all the previous attempts in a buffer, we still need to run the feature matching at each depth estimation.

1. We tried to find a better solution than the above, in order to be able to handle the real world scenario without holding a bunch of images in the memory all the time and with as little additional calculations on new depth estimations as possible.

We started off with the assumption that the only thing that would change with the focus would be the distance between the sensor and the lens barrel in the Z direction. If that is the case, then the only thing that we need to change in our intrinsic camera parameters would be the focal length, as we defined it in ‎1.1.1, and a change in focal length is equivalent to a change in image scale. In order to find the scale that is related to the current focus position we thought of two different approaches:

1. Normalization to base image - The first, simpler approach, is to take all the wide-tele image pairs from the focus attempts and find each pair's feature matches and disparity. Then, take each tele image and find its feature matches with the first one, and the same with the wide images. From these matches we can calculate the disparity difference between each pair. We then try to find the best scale and offset for each wide image and tele image separately, that will minimize this disparity difference. In this way, we use our first image pair as a base and try to scale and offset all other pairs to fit to it. Since choosing the first image as a base is arbitrary, we try to find the "real" base for each camera from the set and normalize the rest according to it (i.e. dividing all other scales by the base scale and subtracting offsets by the base offset). We decided that to approximate the "real" scale and offset we will choose the median in the set for each value.

The possible issue with the above solution is that in the case of a handheld camera, the time between focus attempts is small but not insignificant. During this time the handheld camera moves, even if and the user tries to stay as still as possible. This causes some rotation and translation between images from the same camera which could potentially cause a change in the disparity that is not due to scale and offset, but a change due to the actual change in distance. If such a change occurs, our previous method will try to compensate for it wrongly with different scale and offset than the "correct" one. This led us to try and explore a different solution that will be more robust to scene changes.

1. The essential matrix - the theoretical solution we came up with relies on using the essential matrix (as we described in ‎1.1.2). If we take the formulation of the essential matrix from equations (5) and (6)and use the known properties for inverting upper triangular matrices, we can get the following equation for the essential matrix:

*( 7 )*

If we assume there is no change in the rotation and translations between the wide and tele cameras during refocusing, i.e. the extrinsic parameters remain the same, and that the only real change is with the value of the focal length, the new essential matrix equation for the refocused image is:

*( 8 )*

So since we don't know the new focal length or the scaling for either camera, when we get the essential matrix for the refocused image using the original focal length value, we get a variation on the real essential matrix. Now, if we use the assumption that the real essential matrix doesn't change between refocuses, we get the following equation:

Where A is a scalar factor that comes from the fact that the essential matrix is always found up to a scale. From this equation we can extract the following relations in order to find each camera's scale under said assumptions:

*( 9 )*

So, we can see that if our theory holds up, we should be able to extract each camera's scale simply from knowing the essential matrix during calibration and the current focus's essential matrix. However, since we are dealing with a real world scenario here, and there are probably some other variations during refocusing that come into play (like the movement of the lens barrel from the sensor is most likely not only in the Z axis etc…). So, if using the calibration's essential matrix won't work, we could use each focus set's first image as the baseline essential matrix and calculate each of the other set images scale from it using equation (9). This has the benefit that the changes between refocusing to the same distance more or less, should be smaller and possibly only the lens barrel Z direction shift will really have any significant change.

To validate this assumption without any other factors effecting the result, we created a simulation environment (as we described in ‎2.4) in order to test whether under the previously stated assumptions the theory holds, and we can extract each single camera scale.

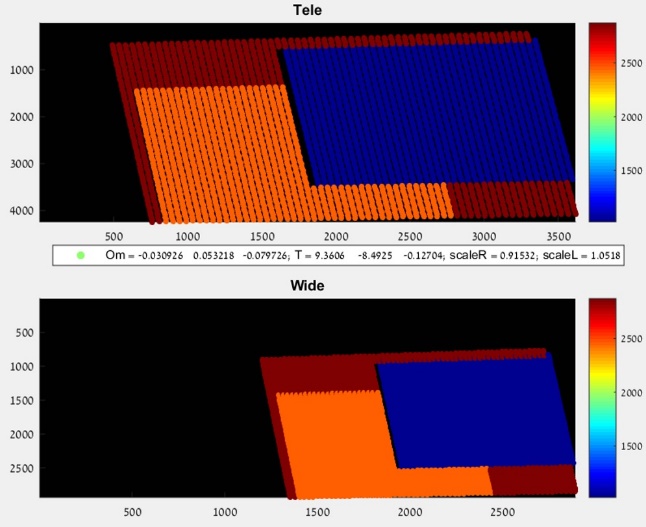


Figure ‎3‑22 - Simulation result before (left) and after (right) adding scale,rotation and translation between the scenes.

Figure ‎3‑22 shows the result of a simulated depth scene. On the left we see the original scene, and on the right is the same scene after both cameras went through the same random rotation and translation (so the relative rotation and translation between them remained the same) and a different random scaling for each camera. The simulated scaling for the wide is 1.0518 and for the tele its 0.91532.

Extracting the Essential matrices from both scenes using only the calibration intrinsic parameters (i.e. not changing the focal length according to the added scale) we get the following matrices:



Figure ‎3‑23 - Essential matrices extracted from the simulated scene before (left) and after (right) the added rotation,translation and scale

Where the left one is the original and the right one is the new Essential matrix after the changes. Using these matrices and our relations in equation (9) we got that the scale for the wide camera 1.0518 and for the tele its 0.91532. So, we can safely say that in the purely simulated scene the theory definitely holds up.

We tested our theory on a real world scene, but still on a tripod, as it is easier to find the real scale value when there is little to no change between the images other than the focus. Figure ‎3‑24 shows the results of 5 refocuses, three chessboard at varying depths scene, on a tripod with a heated folded module. We extracted the chessboard corner for each chart and calculated the disparity range for each corner for the original images, the optimized scaled images (as defined in (a) in this section) and the images with scale calculated with the above formulations. We can clearly see that the above formulation **does not** hold up in practice.

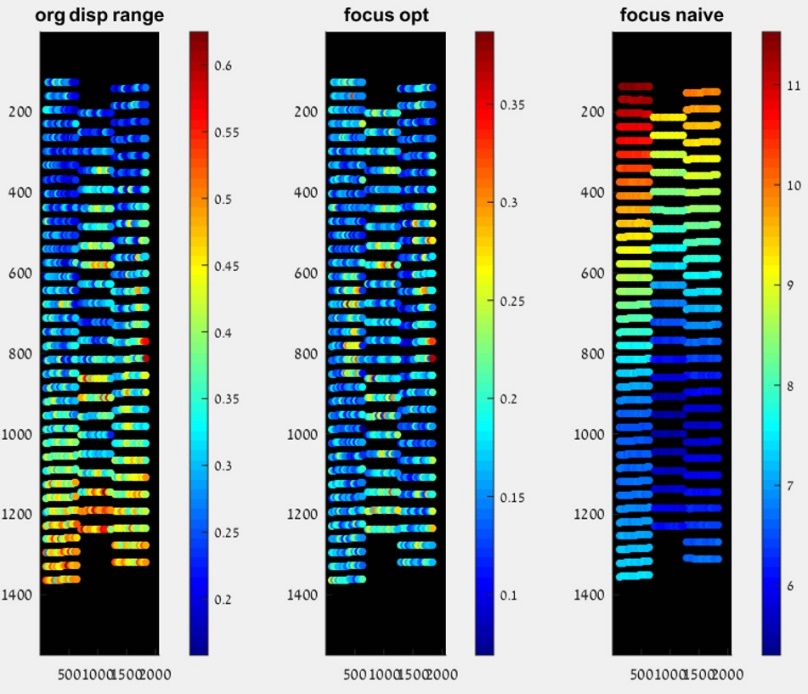


Figure ‎3‑24 - Regular disparity range (left), optimal scale fix (middle) and the naive essential fix (right)

We decided not to give up on the direction of using the essential matrix, but instead we wanted to explore its decomposition and try different optimization methods based on its decomposition's values. Figure ‎3‑25 shows the results for the essential matrix decomposition's rotation (om) and translation (t) between both cameras, of both the baseline images and a random refocus attempt, before and after the optimal scale fix (as defined in (a)) we get:

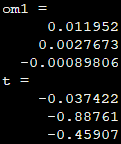
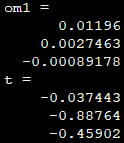
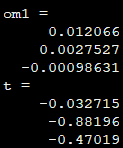


Figure ‎3‑25 - Rotation and translation from Essential decomposition of the baseline image (left), a random focus attempt (middle) and the same attempt that has been fixed with the optimal scale (right)

As we can see, after the optimal scale fix the rotation does seem to come a bit closer to the original one, but the translation seems to remain the same as the current one. This led us to try and use optimizations on the rotations and translations calculated from the essential matrix. We also tried to use the optimization for the calibration parameters (i.e. optimizing the rotations to suit the rotation from the calibration), since if this can work out we can skip the fix that requires taking several refocuses for each image.

The optimization options we tried:

* Optimal (focus opt) - This is the optimization method as mentioned in (a) in this sub-section.
* Match essential matrix (focus/calib E) - Try to get the current essential to match the first focus attempt/calibration essential up to a scale factor using only scaling for each camera.
* Match rotations (focus/calib Om) - Try to get the current rotation to match the first focus attempt/calibration rotation using only scaling.
* Match rotations and keep translation (focus/calib OmT) - Try to get the current rotation to match the first focus attempt/calibration rotation while keeping the translation the same. This came up as an option after the essential decomposition analysis above.

Any title that has 'AndOffset', also optimized the offset and not only scale (optimal did offset optimization as well). Figure ‎3‑26 shows the results of all the above optimizations on the disparity ranges.

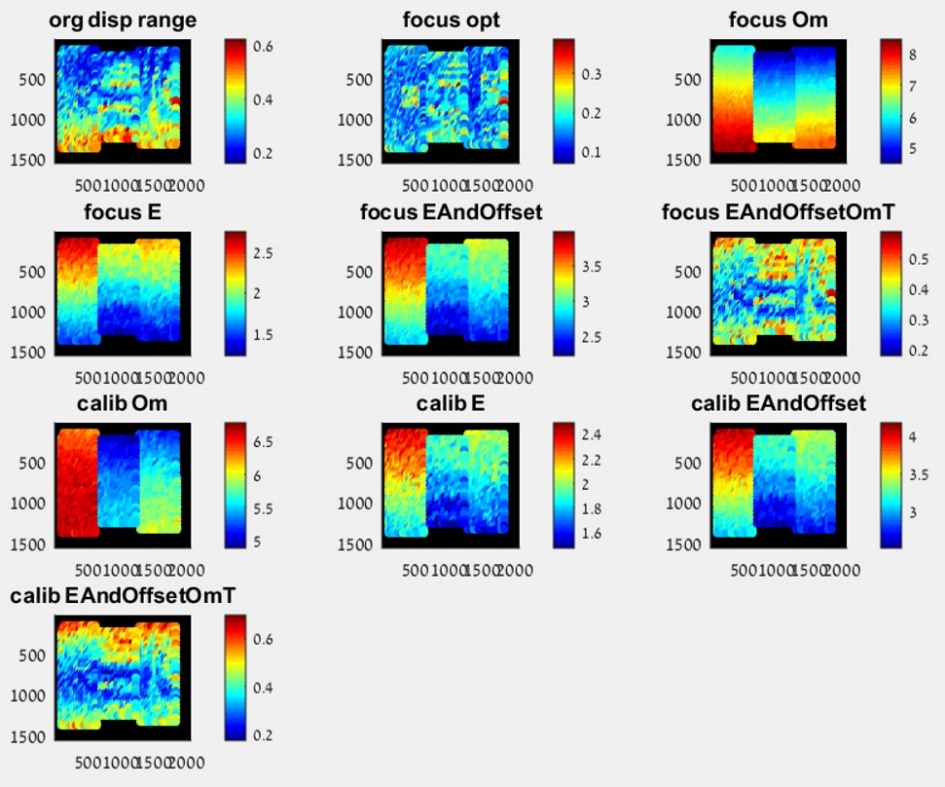


Figure ‎3‑26 - Disparity ranges for different optimization methods

The optimal scale is by far the best option, with none of the essential based optimizations really holding up. This is probably due to the fact that the base assumptions for what changes with refocusing are probably wrong, it would seem that more than the focal length changes. It could be that the result of any number of factors:

* Optical axis location changes (sensor position with respect to the lens barrel).
* Translation between the cameras change. This makes sense, and in fact in the Z axis it's quite obvious since the lenses don't move together. This could also be an issue in the other axis since the lens barrel doesn't move completely in one axis for the focus mechanisms.
* Relative rotation between the cameras change.
* Other reasons we didn't think of.

The conclusion here is that we need some more info from the camera modules in order to actually use the essential fix, like gyro info or focus position info. Currently our attempts at accessing these values lead to unusable results as we've shown before.

After reaching the above conclusions, we decided to check how much the handheld scenario effects the optimization method described in 1 in this section (since that is the only one that seemed viable for the tripod case). Figure ‎3‑27 shows the result of the optimization on the same scene as before only in the handheld case and not the tripod case.

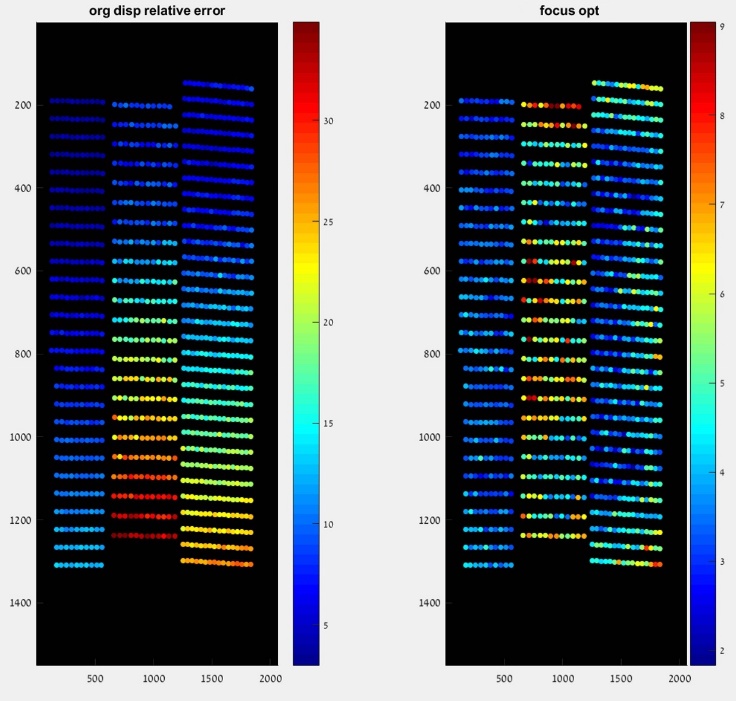
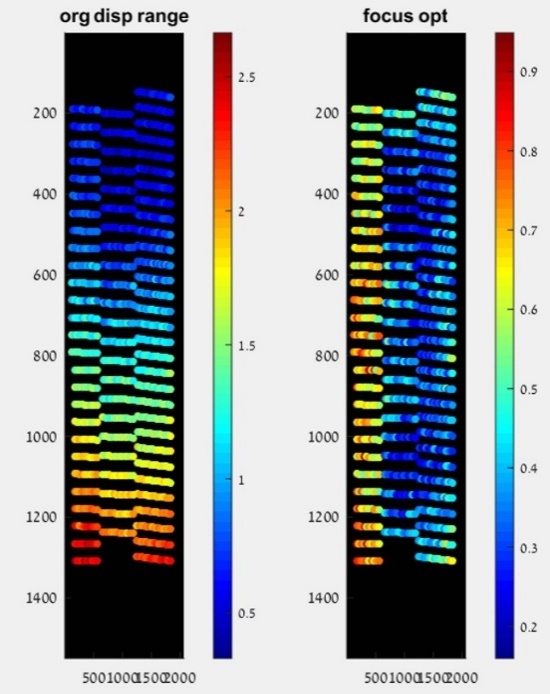


Figure ‎3‑27 - Optimal scale corretion, disparity ranges (two left plots) and relative error (two right plots), before (leftmost of each pair) and after optimal scale correction (rightmost of each pair)

As we can see, the optimization method works very well even in a handheld scenario. We can also see that the only "large" ranges in the disparity are in the closer chart, for which the disparity is larger anyway so the relative error is much smaller.

In conclusion, we should use the optimal scale correction with the optimization metric we defined in 1 in this section and it should work well even for handheld scenarios. We believe there is some merit to the theoretical essential based method and that they are worth exploring further once we manage to get some additional sensor data from the modules (gyro position, focus position etc…) or understand the mechanical focus procedure better.

## Disparity to depth conversion

All the above fixes are mostly in order to get repeatable disparity for different objects at the same depth in different scenes and exploring whether the same sensor location is needed or not. They do not, however, deal with the underlying issue that is the conversion from disparity to depth.

As we can see in Figure ‎3‑1, there is a large error when trying to use the basic stereo theory for the conversion. This could be caused due to several facts:

1. The calibration minimization function (as described in ‎1.3.2) does not attempt to minimize the depth estimation from the stereo images.
2. The calibration process is done with only one focus distance, which is very close (60cm). This could be an issue for several reasons:
   * Close object's disparity is quite large anyway, so the relative error is small even for a set offset (which will not be the case for farther distances).
   * As we've seen in our investigations, the disparity results could vary between different refocusing, even for the same distance, so using only one focus attempt could be unstable.
   * Focus change effects are not linear with distance. For example, the change between 60cm focus and 120cm focus is much more noticeable than 120cm and 180cm. So taking only a very close focus distance could be a very bad estimate for the normal use case scenario since most natural images are taken at a farther distance, and the calibration parameters are effected by the focus but, as we mentioned before, are not adjusted to different focuses.
3. The temperature of the modules isn't regulated, i.e. the calibration isn't done with heated modules. We've seen how this could affect the disparity, and therefore the extracted module parameters.
4. Our calibration is done with a 3D chart that is not close to flat. This could be an issue as we've seen that the depth of the object has an effect on its scale and non-disparity axis error (which is a part of the minimization process).

As we've seen, even if we fix all the issues investigated in this chapter, we still remain with some offset from the normal stereo theory that we need to fix.

**Fix options**

1. Calibrate each module at several distances, while heated (or possibly during several heating points, initial, middle and saturation), and refocus several times to handle focus variance. This would create a bunch of calibration parameters that we can use and along with a more accurate focus position report should enable us to get a better conversion from disparity to depth even using the naïve theory. The main issue here is the effect on production line - the additional time per module this would take, and the extra space that calibration from far distances (along with the pretty large chart that will be needed) will take. These issues make this solution a much less viable one.
2. Using an online learning algorithm and a known world size, find the required fix for each disparity and sensor location to convert it to depth accurately during the normal camera operation. One such known size could be human faces. Figure ‎3‑28 shows the result of a study done in America on some common sizes in an adult face, and specifically the IPD (interpupillary distance).

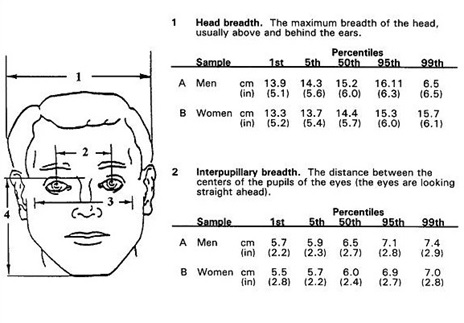


Figure ‎3‑28 - Results of a study done in America on human face interpuppilary distance. Picture from New biometrics Ear-Eye Pitch, by Leonid Naimark, Paul C. Briggs.

The study on which the result in Figure ‎3‑28 is based, claims that for males the median IPD is 6.5cm with 3.6 standard deviation and for females it is 6.0cm with 3.5 standard deviation. From statistics we know of Chebyshev’s inequality:

*( 10 )*

Using this inequality, we can deduce that we can get within 1mm accuracy of IPD with a sample of around 13 people of the same gender with ~1% probability. We note that the study was done on Caucasian adults, so we would need a system to identify age (at least in a binary sense, adult or child), gender and probably ethnicity. The system will probably be comprised of a neural network (possibly two) and we will discuss more about it in the future work section.

Figure ‎3‑29 shows the resulting relative error in distance estimation as a function of IPD estimation error. The relative error has no dependency on depth, so we get the same relative error for every depth as long as our IPD estimation error remains the same.

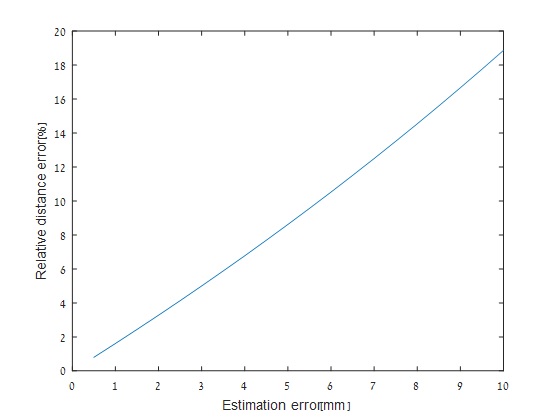


Figure ‎3‑29 - Theoretical relative error in distance estimation as a function of IPD estimation error.

We can see that if we indeed manage to get below 1mm error, the relative error in the depth estimation is ~1%, which means that if we collect enough samples as we stated before and average the results, we can get a depth estimation from IPD with ~1% relative error.

Following the conclusion from above, we can extract faces from images taken by our module, extract their landmarks using the method defined in ‎1.2.3, calculate the disparity between matching faces in the wide and tele images, and the use the following equation to determine each face depth:

*( 11)*

Using this depth estimate, we can find a naïve fix, a simple offset correction for the current disparity (after the previous fixes and limitations to improve repeatability) for each location on the sensor.

Figure ‎3‑30 shows the results of a face chart image and verification chart, taken at the same distance with the same heated folded module on a tripod.

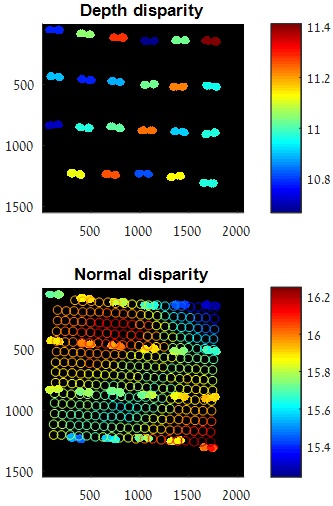


Figure ‎3‑30 - Disparity from depth estimated from IPD (above), normal disparity calculated between matching landmarks (filled circles) and disparity between chessboard corners (empty circles) in normal images (below) of charts at the same depth, after the fixes we mentioned before

We can see the disparity we expect to see above from the depth estimated by using equation (11) and the disparity found in the images (below) after all the fixes we discussed above.

Figure ‎3‑31 shows the relative error remaining after using the disparity from estimated depth from the face chart image as a correction for the disparity in the verification charts in different ways:

* Top left depicts the relative error remaining after all the fixes mentioned in this chapter.
* Top middle shows the relative error result of finding the closest face landmark and taking its appropriate disparity fix.
* Top right shows the relative error result of taking the mean of all the offsets between the disparity in the face chart images and the disparity estimated from depth and using this as a global fix for all corners.
* Bottom left/middle shows the relative error result of simply taking the maximum/minimum offset found from faces respectively as a global fix.
* Bottom right shows taking the offset from the center of the image as a global fix.

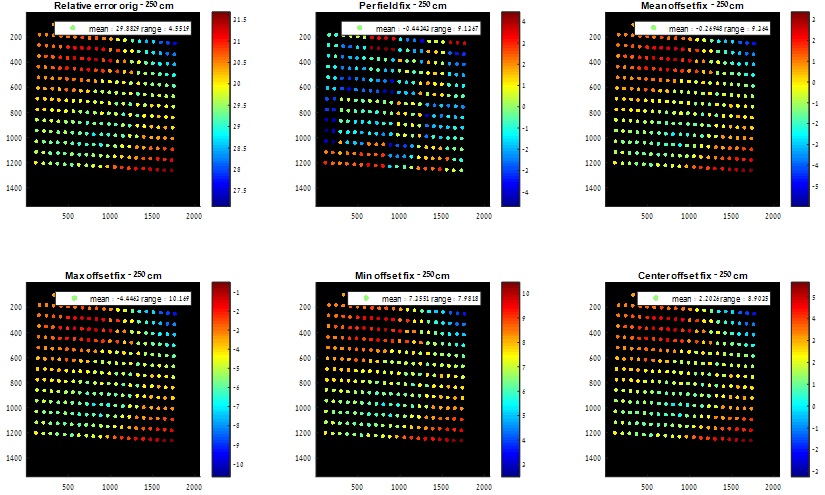


Figure ‎3‑31 - Results of fixing the disparity of a chessboard chart from the face's found offset correction

As we can see, taking the closest fix yields the best results, but also simply taking the offset from the center of the image yields promising results with a much reduced need for the amount of faces needed for corrections.

# Conclusions

As we have explained in chapter 3, there are a lot of issues that we face when we try to get an accurate depth estimation from the disparity of mobile stereo images, starting from module heating all the way to focus issues. We have suggested fixes for most problems, however there were still some issues for which we don't have a fix without more accurate information from the camera module itself.

As a culmination of our research, we designed the following phase 0 demo flow with its stated limitations below using the fixes described in chapter 3. As we have seen in Figure ‎3‑30 and Figure ‎3‑31, if we use our fixes we can get a repeatable disparity value and use the learned fix to reduce the relative depth estimation error from 30% down to 5%.

We have started implementing it but it’s a separate work than this project, in here we simply show the overall design and its limitations. We have plans for more advanced demo phases, where we will have dedicated application to enable us more control on the different camera operations, but for now we focus on the phase 0 demo which assumes no specific application is available.

## Phase 0 demo

### Limitations

#### Focus management

As we have seen in ‎3.5, the focus of the camera has a large effect on the image taken in several ways, and as a result of focus differences we can see different disparities for the same distance on different sensor locations or in different focus attempts. In order to handle this issue we restrict our application in two ways:

1. **Fixed focus position** –in order to increase the repeatability of the disparity results between different scenes, we don’t want to rely on the auto-focus at all. For this matter, in further phases of our demo we will create an application that works only in a specific focus position, but for phase 0 (where we won't have a designated application) we simply choose a value for both camera's focus position so that 1.5m is in focus. Applying this restriction limits our ability to fix close items mostly, but as we mentioned before, their disparity is larger anyway so it has a lesser effect there. For the far off items, the focus difference isn't too large (it's not linear with distance, and most of the difference is from small distances to slightly larger ones) and we think that the most common use case will be for a "normal" distance range of 1m-4m.
2. **Focus calibration** – as we have seen, the above restriction does indeed help to limit the disparity range, but testing it out empirically shows that setting the focus position manually does not achieve the exact same results each time. This could be a downside of the manual focus position setting and when we reach the application stage it might be better, or this could be the result of the focus control mechanism in our module and will be an issue in the application phase as well. In order to lessen this effect to some extent, we add a focus calibration phase to both our learning phase and the application phase.

The focus calibration phase consists of:

1. Refocusing to the fixed manual focus position 5 times and retaking captures at each time.
2. Use the fix option described in ‎3.5.3, as option 2.b.1.
3. After we find all the scales for all the images and find the median one, we correct the final image to fit that scale.
4. Set the application to “focus lock mode” from that point on so that the focus won't change.

This is our way of finding the most likely scale correction for our scene using several focus attempts and their median scale, and then fixing this scale without allowing any more focus changes to improve repeatability.

#### ROI specific depth

In order to limit the issues we found in ‎3.5.2 of the difference in the focus of the wide and tele cameras, we decided to output the depth of the chosen ROI only (of course the chosen ROI could contain multiple depths, in this case we will output the majority depth) and not a dense depth map. This will enable us to find a scale correction that is suitable to this specific depth (as we have mentioned, due to the Z axis translation, there is a different scale per depth). In this manner, if we apply the same procedure to the learning phase and the application phase, we could help alleviate some of the wide to tele focus issues.

#### Heated modules

Due to the issues we have seen in ‎3.1, we decided that, for now, we will only enable our learning and application phases after the cameras are sufficiently heated. This means that with our folded module we will require the camera application to be active for 3 minutes before allowing for our application to run. In the real world scenario, we would either need a temperature measurement or some other mean of pre-heating the modules to handle this.

#### Face chart

For our phase 0 demo, we decided to focus on the fixes themselves and not on the statistical learning convergence, since real faces have some variance in IPD as we have seen in ‎3.6. For this purpose we decided that our learning phase will use images of the face chart, to get multiple faces in each image as well as to eliminate errors caused by the IPD variance. In the real world scenario we would expect a large amount of face images to ensure convergence before activating our depth estimation fix, or at least giving a wider margin of error before a certain amount of faces are collected.

#### Offline training

Since phase 0 is demo oriented and the training is a difficult concept to present in a demo, we have chosen to do all our learning phase training offline, and only the application phase will be done online.

### Suggested flows

#### Focus calibration

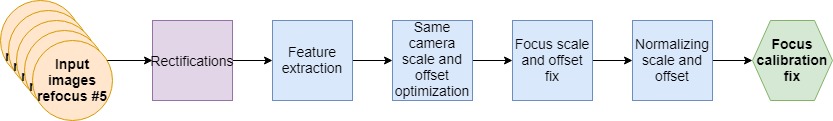


Figure ‎4‑1 - Focus calibration flow

#### Application phase

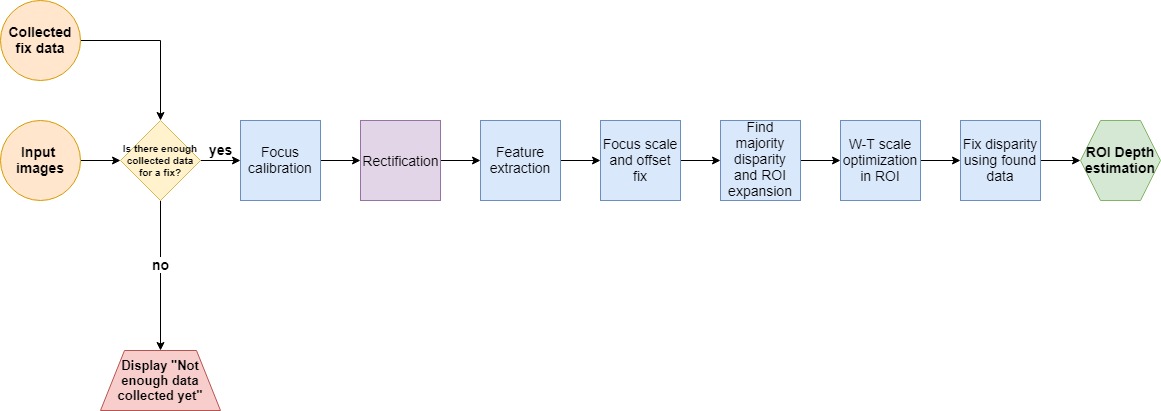


Figure ‎4‑2 - Application phase flow

#### Learning phase

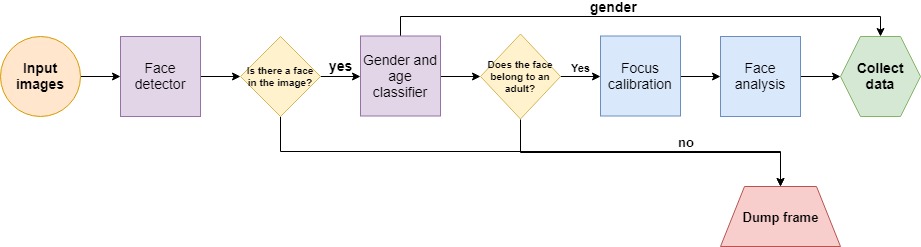


Figure ‎4‑3 - Learning phase flow

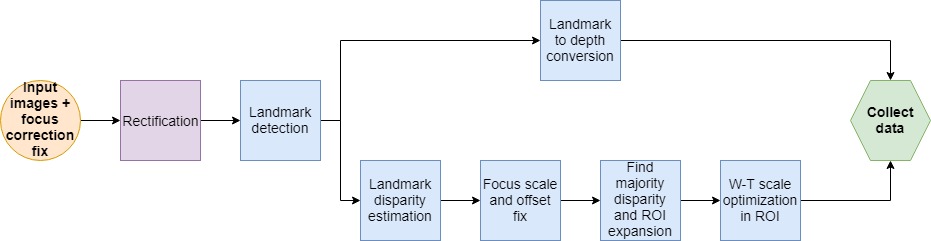


Figure ‎4‑4 - Face analysis flow

### Block descriptions

#### Feature extraction

This block uses feature extraction and matching as described in ‎1.2.1, and then it refines and filters the matches by using peak refinement, histogram based filters and some global heuristic filters to get more reliable and accurate results.

#### Same camera scale and offset optimization

This function preforms feature matchings as described in ‎1.2.1 between the features of the same camera in different focus attempts (each focus attempt is matched with the first). Then it uses an optimization function to find the scale and offset between each image in each camera and the first image from that camera, so that the disparity difference between the focus attempts will be minimal.

#### Normalizing scale and offset

This function takes all the found scales and offsets and normalizes them to their mean. Meaning that it finds the mean offset and scale in the set, divides all scales by the mean scale and subtracts the mean offset from all offsets.

#### Focus scale and offset fix

Use the found scale and offset from the focus calibration for each camera to fix the disparity to a more stable value.

#### Find majority disparity and ROI expansion

Find the majority disparity within the reported ROI from the camera modules. Expand the ROI to include all nearby features that have a close disparity to the majority's disparity. Find the new majority disparity after expansion.

#### W-T scale optimization in ROI

Take all the features and matches within the ROI and find the optimal scale to minimize the non-disparity error range in it.

#### Fix disparity using found data

For each feature in the ROI, find the closest one to it in terms of sensor location and disparity within the fix dictionary, and estimate the disparity offset needed to "fix it". Then use the majority depth of them all to determine ROI depth.

#### Landmark detection

Use open face to find faces within the images and extract the landmarks from them, as described in ‎0.

#### Landmark to depth conversion

Using the found landmarks, find the face's IPD in pixels from the center of mass of each eye's landmarks. Then, using equation (11), find the depth estimate for each found face in the image.

#### Landmark disparity estimation

Take the found landmarks in each image, match each set from one image to the other by constraining the maximum non-disparity error and the found depth. Then use peak refinement to improve the quality of each match and estimate the disparity of the face.

#### Fix dictionary

Using the focus calibration scale and the W-T scale, for each face collect the found disparity, found depth and location on the sensor into a fix dictionary.

# Future work

## Phase 0 demo continuation

As we elaborated in chapter 4, we have planned a demo flow to show the feasibility of some of our fixes under strict restrictions. Therefore our first line of future work would be to implement the flow and test it out.

## Further issue handling

The phase 0 demo will work under some strict restrictions to avoid some of the issues we couldn't fix yet. Those issues will need to be addressed in order for our application to be viable on any mobile phone. The issues are:

### Temperature

In order to be able to run our application on the device, we'd need to be able to handle the temperature issue. Fix options to explore:

* Getting the temperature report from the module itself.
* Pre-heat the module once our application starts.

### Focus

Handling the focus issues we've mentioned in ‎3.5 without forcing focus lock. Fix options to explore:

* Figure out how to get accurate and usable focus position report from the device and then use it to learn fixes per focus position.
* Use the device's own focus procedure to extract some valuable useable data. Among the data we can use is:
* The focus metric score to disqualify out of focus images for instance.
* The focus stack itself (the batch of images the device takes to find the best focus) – we can use this to fix the scale issue without having to retake images a bunch of times, we will also have an order to the focus change in the images.
* And possibly other data from the focus procedure we are not aware of.

### Face data collection and extraction

In order to be able to run our application on a real world scenario, we can't rely on faces having the same IPD or even to converge to an expected value, since, as we have shown before, there is a big variance depending on gender, age and even ethnicity. Fix options to explore:

* Create a neural network for landmark detection.
* Create a neural network (or possibly several neural networks) to detect the age (could be a binary detection – adult/child), gender and ethnicity of the faces located in the image taken in order to group similar data together for it to converge to an expected value.
* Create a neural network to re-identify face we have previously seen, so that we can deduce further information from them without any additional cost.

Most of our thoughts about exploring the fixes for the stated issues involve an application tailored specifically for our needs, to allow us greater control on the device's functions, which is why they weren't explored in this project (or that they were big tasks themselves and therefore out of the scope of this project, like training neural networks).

## Physical improvements

Another way we can use to improve our results is to physically alter our camera setup. Possible options to explore:

Check if by using better quality materials we can avoid some of the issues we found above - for instance if we use a glass lens, maybe the heating won't affect the results so much, or by using a better focus actuator we can either improve the repeatability or get accurate focus position report. Check the cost benefit trade off of such improvements.

Check the benefit of adding an additional camera to the setup. Possible beneficial additional cameras:

* TOF camera – while this might reduce the need for the disparity to depth conversion, standard TOF cameras are sparse and not dense so if we use the TOF camera as "real world sizes" instead of faces we can adjust the disparity to depth conversion in a per scene basis and get a dense depth map from the disparity map. This is quite a costly addition.
* Fixed focus wide camera – using the fixed focus camera as a point of reference, we can scale both images to a constant, unchanging focus, and therefore avoid most focus issues we described. This could be a low resolution camera, and since it's fixed focus the addition could be quite cheap.

## Combining with existing methods

* Use a neural network that is pre-trained to extract actual depth from the scene (this is not an easy problem) or alternatively a network that is trained specifically for depth extraction from faces. This can save us the entire statistical part of our problem and will allow us to collect a lot of useable data points from a small amount of faces/scenes.
* Use a structure from motion algorithm along with data from the phone's gyro and accelerometer to get a depth estimate over a few frames from either or both cameras as a known world size.

# References

[1] Multiple View Geometry in Computer Vision Second Edition. Richard Hartley and Andrew Zisserman, Cambridge University Press, March 2004

[2] Camera Calibration Toolbox for Matlab, Jean-Yves Bouguet

[3] OpenFace: an open source facial behavior analysis toolkit, Tadas Baltruˇsaitis, Peter Robinson, Louis-Philippe Morency