technische universität dortmund

BOSCH
Technik fürs Leben

Eliminating Ghost Artifacts in the Overlap Area of a Surround-View System

Sumukha Manjunatha
(Matrikelnummer: 162927)
February 2, 2015

Supervisors:
Prof. Dr.-Ing. Gernot A. Fink
René Grzeszick, M.Sc.
Dr.rer.nat. Hartmut S. Loos
Mr. Patrick Klie

Fakultät für Informatik
Technische Universität Dortmund
http://www.cs.uni-dortmund.de

Robert Bosch Car
Multimedia GmbH
http://www.bosch.de

ACKNOWLEDGEMENT

I would like to take this opportunity to thank Prof. Dr.-Ing. Gernot Fink and Mr. René Grzeszick, M.Sc. for accepting to supervise my thesis. I appreciate their time, continued guidance and support during the course of the thesis. I would also like to thank them for reviewing my final thesis. I would like to thank Mr. Grzeszick for his immense patience in periodically reviewing my progress, giving me suggestions and discussing the various probable solutions to the problems I came across.

I wish to thank Robert Bosch Car Multimedia, Hildesheim for giving me this opportunity to complete my Master Thesis at their premises. I would like to take this opportunity to express my heartfelt gratitude to my supervisor, Dr.rer.nat. Hartmut S. Loos for having the confidence in me and giving me this responsibility. I extend my sincere thanks to Mr. Patrick Klie, who was always a strong support, helped me record all the real world sequences and guided me technically all through the course of this thesis. I would also like to thank Dr.rer.nat. Thomas Reinert for his support in helping me understand the mathematics behind the surfaces and meshes. I thank my fellow interns at Robert Bosch Car Multimedia, Hildesheim who helped me keep my morale up during difficult times.

Lastly I would like extend my sincere thanks to my family back in India, and friends in India and Germany who have been a constant support during the course of this work.

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

In recent times, with increasing car sizes and luxuries, driving safety has gained paramount importance. As per a report of the US Department of Transportation from the National Highway Traffic Safety Administration [Adm08], most fatalities and injuries during backing are caused by passenger vehicles. "Among cases where the type of the striking vehicle is known, 78 percent of the backover fatalities and 95 percent of the backover injuries involved passenger vehicles" [Adm08]. The report also states that backover injuries are more frequent in nonresidential parking lots than other places. The investigations revealed that majority of non-occupants were approaching the vehicle from the side. There is also a law enacted in the US recently with all cars under 10000 pounds to be fitted with rear view cameras and display units to aid the driver [Tra10].

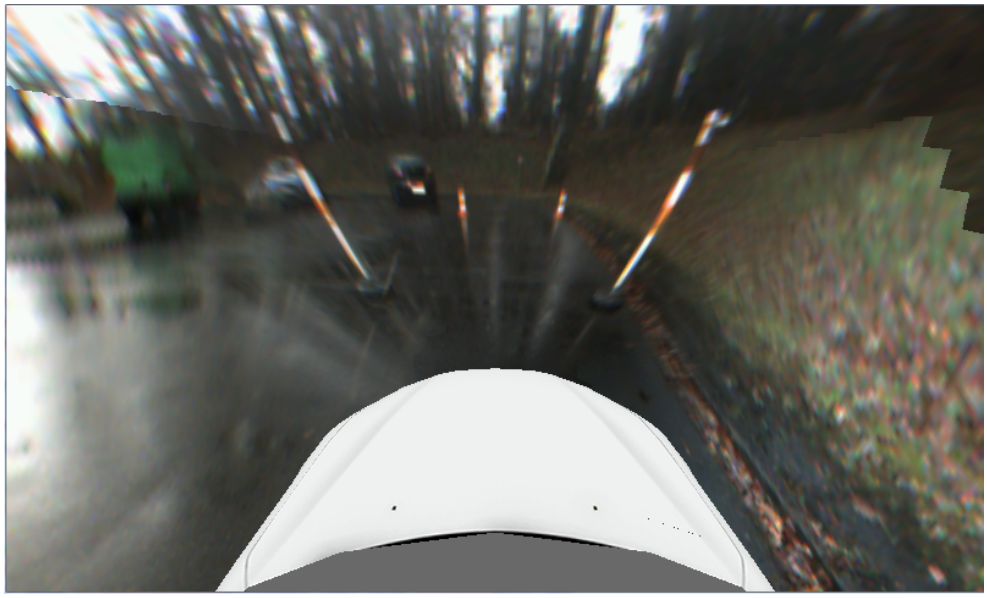


Figure 1.1.1: An example of a parking scenario - Image provided by Robert Bosch Car Multimedia GmbH

Similar laws would be enacted all around the globe. In future, driver assistance systems would be a necessity of any automobile, rather than a luxury. Our reliance on these systems would be on the rise and hence, these systems need to be extremely accurate. In case these systems were not accurate, then they would cause more accidents or would be rendered useless and drivers would not use them. Consider an example of a driver parking into a slot in between two poles for guidance as shown in Figure 1.1.1. Assume, for example, that the driver assistant systems shows a view as shown in Figure 1.1.2. As can be seen, imaging artifacts were introduced and it is very obvious that the driver sees a different view as compared to the assistance system. This would prompt the driver not to use the system and switch it off. This thesis analyses such a situation and presents an unconventional approach to solve this challenge.

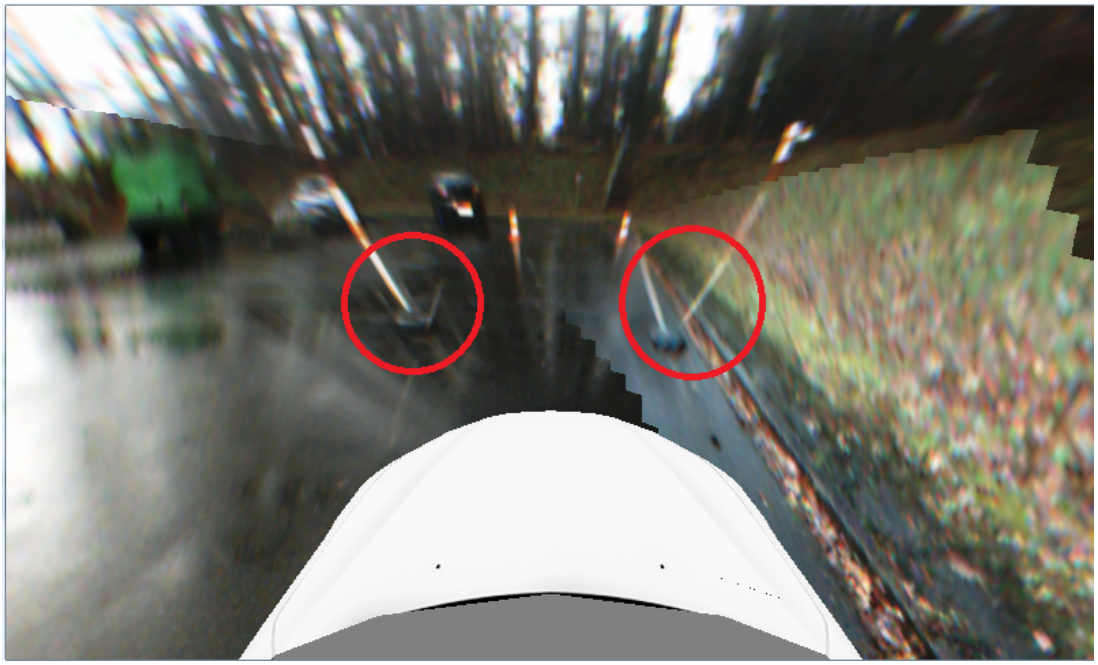


Figure 1.1.2: Visualization on Head Unit of the parking scenario with imaging artefacts introduced (two poles are seen instead of a single pole at both locations) - Image provided by Robert Bosch Car Multimedia GmbH

1.2 FRAMEWORK AND SETUP

The driver assistance setup considered in this thesis, consists of a car with four cameras and a head unit for visualization. The cameras are mounted on the front, the rear and the two side mirrors. Each of the cameras has a wide field of view (more than 180° on one direction and about 137° on the other). The cameras are mounted in such a way, that each camera has an overlap with the adjacent camera. Therefore, the four cameras put together, cover the complete surrounding area of the car.

For visualization purposes, the car is thought to be placed on the flat surface of a bowl. The base of the bowl is circular and the walls of the bowl follow a parabola. The transition from the flat surface to the walls is made as smooth as possible. The top view is as shown in Figure 1.2.1 and the sketch of side view is as shown in Figure 1.2.2. The dark blue triangles (indicated by red circles) in Figure 1.2.1 are the positions of the four cameras, the light blue areas are the regions visible to one camera only and the dark blue regions on the four corners are the regions visible to two adjacent cameras. The bowl is made up of a triangular mesh (explained in detail in Section 1.3) and rendered using OpenGL® (ES) ("OpenGL® and the oval logo are trademarks or registered trademarks of Silicon Graphics, Inc. in the United States and/or other countries worldwide."). The images are projected onto this mesh assuming a flat surface. The images are captured by the cameras in real time. These are rendered on this triangular mesh and are visualized on the head unit of the car. This system is used to assist the driver, especially during parking scenarios, where the driver cannot see everything around the car.

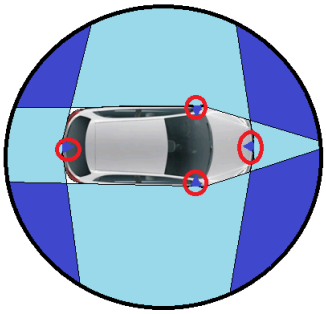


Figure 1.2.1: Top View of the driver assistance setup

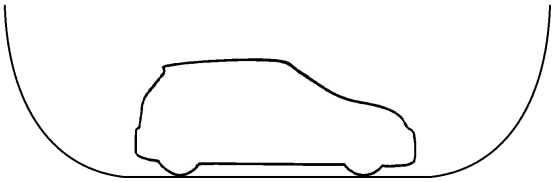


Figure 1.2.2: Side View of the driver assistance setup

For testing purposes, a simulator as shown in Figure 1.2.3 is available. The simulator simulates all the features of the head unit and can be used on a PC under Windows® platform. A tool is used to grab images from the cameras in real time and save them as a sequence. These can be replayed on the simulator (to visualize the same events that would occur in the real world with the head unit).

Figure 1.2.3: Screen-shot of Simulator simulating head unit of the driver assistance system setup - Image provided by Robert Bosch Car Multimedia GmbH

1.3 FUNDAMENTALS OF IMAGE PROJECTION AS USED IN THE SURROUND-VIEW SYSTEM

1.3.1 *Feature/Point Correspondence*

The Surround-View system is made up of images. There are four different images, and the visualization must look seamless across images. For such a seamless transition between images, image stitching can be employed which uses feature correspondences

between images to form the transformation from one image to the next. A *Feature/Point Correspondence* is always made between two images that contain/capture two different views of the same scene and have some part of the world in common [Sze10]. See Figures 1.3.1 and 1.3.2 for an understanding. The two images are two different perspectives of the same scene. A Point Correspondence is the set of pixel values in both images corresponding to the same point in the world. In the Figures 1.3.1 and 1.3.2, the pixels marked with a red cross correspond to a single world point. The pixel positions of the two points (in the two images) form a Point Correspondence. The area of the images that contains a common part of the scene is called the area of overlap.

Figure 1.3.1: One Perspective of a Natural Scene[1]

Figure 1.3.2: Another Perspective of the same Scene[2]

1.3.2 *Image Stitching*

Once we have feature correspondences, we need to use it to "stitch" the two images into one larger image. This is achieved by *Image Stitching* [Sze10]. Image Stitching is a domain of research focused on resolving challenges arising out of generating a single huge image from multiple smaller images of different parts of the same scene. There has been substantial research already done on the basic topic and currently more advanced challenges are under research. Consider the images shown in Figures 1.3.1 and 1.3.2. Using image stitching the composite image formed after stitching the two images would be as shown in Figure 1.3.3.

1 Image sourced from http://hugin.sourceforge.net/tutorials/two-photos/974-1.jpg on 20th January 2015
2 Image sourced from http://hugin.sourceforge.net/tutorials/two-photos/975-1.jpg on 20th January 2015

Figure 1.3.3: Composite - Stitched Image using algorithms of Image Stitching[3]

1.3.3 *Alpha Blending*

Once the stitched composite is available, it still has certain artefacts of stitching. Normally, the line across which the images are stitched is noticeable. This is removed using various blending methods. In our case, *Alpha Blending* is used. Alpha Blending is normally used to blend a background colour with a foreground color. In our scenario, the same is used a little differently. All images are made up of three channels, red, green and blue (in case of RGB color space). In computer graphics, a fourth channel is added, called the *Alpha Channel*. This channel decides the transparency with which the image is rendered/displayed on screen [SG09]. When displaying, there is always an opaque background (in our case, this is colored black). Images are rendered on this background with varying alpha values. In regions where a single camera views the scene, the images are displayed opaquely. When there are two cameras viewing the same portion of the scene, as in the case of region with overlap, the images are rendered in such a way that at any point in the overlap region, one of the images is always drawn opaquely. This is done so as to avoid viewing the background when both the images are drawn transparently. The way the alpha values change is controlled so that the transition between the images is as smooth and intuitive as possible.

3 Image sourced from http://hugin.sourceforge.net/tutorials/two-photos/en.shtml on 20th January 2015

1.3.4 *Mesh/Mesh Grid*

Once the blended, stitched image is available, it needs to be presented to the user in a way that is pleasing. This is also referred as *visualization*. This is achieved, in the field of Computer Graphics, using *Mesh/Mesh Grid* and *Image Projection*. A Mesh/Mesh Grid is a set of vertices, edges and faces that describe a 3D object [Edeo1]. Typically, triangles are used as a simplex in space. See Figure 1.3.4 for an explanation of how a pyramid is formed using these basic geometries. However, more complex shapes can also be formed using these basic structures. A *Vertex* is a point in 3D space. An *Edge* is a straight line segment connecting two vertices in space. A *Face* is a closed surface enclosed by edges. Meshes are used extensively in 3D graphics for displaying complex 3D characters/objects. The most popular meshes used are *Triangle Meshes* where, the faces are made up of three edges to form a triangle as shown in Figure 1.3.4. However, with the evolution of more powerful Graphic Processing Units (GPUs), *Quad Meshes* (where faces are formed by four edges to form a quadrilateral) and higher dimensional meshes are also used.

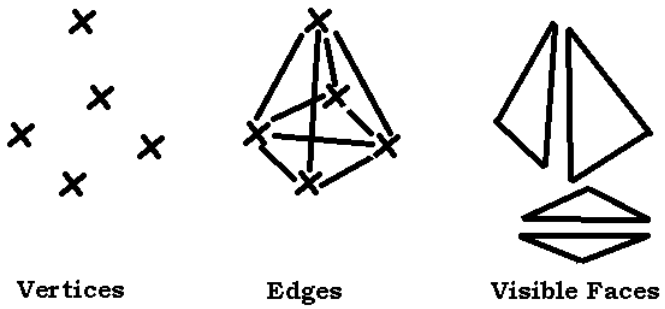


Figure 1.3.4: Vertices, Edges and Faces in the context of Computer Graphics

1.3.5 *Image Projection*

Once the skeletal Mesh Grid is available, it needs to be filled with images for visualization. This is dealt by "projecting the images" to the mesh. This process is also called *Texturing* [SGo9]. *Image Projection* is the process of presenting the images captured by the cameras to the user. For visualization purposes, the triangle mesh created above is used and the images captured are displayed onto this mesh. In our scenario, the images so captured will first be rectified to remove the distortion effect caused by the fish-eye optics. Then, they will be projected to the mesh surface (this process is similar

to that of an overhead projector). The projection is done assuming a flat surface. That is to say, any vertical object in the scene, in the close vicinity will appear to lie on the ground, as shown in the top left and top right sketches of Figure 1.4.1. Since, the projected surface slowly transitions to a vertical plane, it gives a realistic view of the scene.

1.4 PROBLEM DESCRIPTION

The visualization of the Surround-View system is created by stitching the images and blending of alpha values as described in Section 1.3, in the overlap areas of the bowl. As described earlier, the flat surface assumption is used for displaying the images. This assumption is valid for far away scenes, where, the changes in depth of scene is much smaller than the distance of the camera to the scene. However, when there is a small obstacle in the region of overlap, this assumption no longer holds. And due to the alpha blending of the two images, it gives rise to dual artefacts of the same object. Such dual artefacts will be referred to hereafter as "***Ghost Artefacts***". Figure 1.4.1 shows a sketch explaining the scenario assuming a pole (an obstacle) in the front-left overlap region of the car. The same scenario in the real world with an obstacle (a fire hydrant post) in the front-right overlap region, is as shown in Figure 1.4.2.

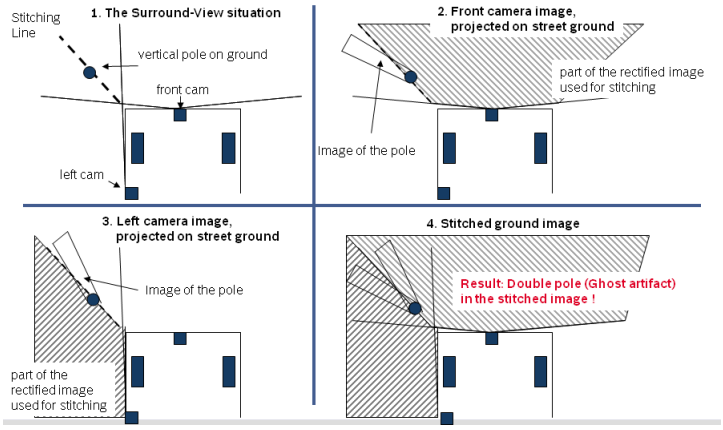


Figure 1.4.1: Ghost artefacts in the Surround-View situation, left top is the real world situation, right top is the image projected by front camera, left bottom is the image projected by left camera, right bottom is when both left and front cameras project the image giving rise to ghost artefacts - Sketches provided by Robert Bosch Car Multimedia GmbH

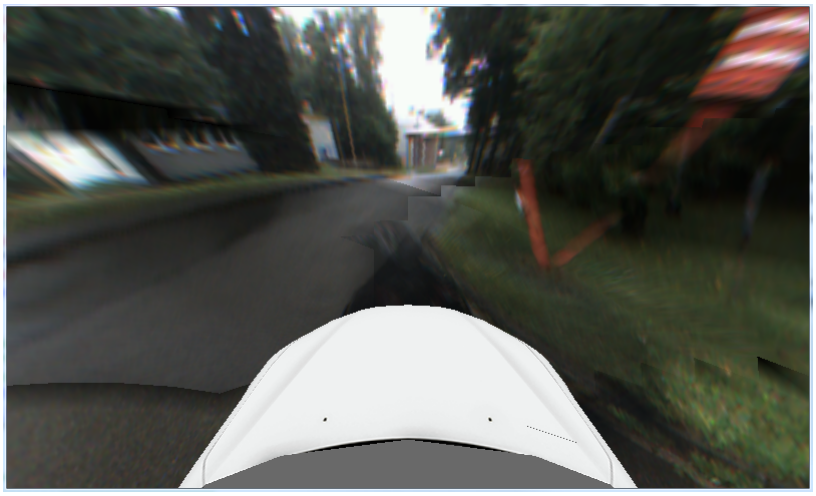


Figure 1.4.2: Ghost artefacts as viewed on Head Unit/Simulator of the Surround-View System - Image provided by Robert Bosch Car Multimedia GmbH

Assuming that we have reliable good correspondences in the overlap area (in the context of this thesis, these have been marked manually), this thesis discusses a possible unconventional approach, that uses deformation of the projection surface in order to reduce ghost artefacts, and the respective challenges to solve the following research question: "Find a suitable method, using known, good and reliable feature correspondences, such that, the objects in the overlap region do not disappear and ghost artefacts in the close proximity of the car are visibly reduced."

The thesis is organized as follows: Section 2 discusses related work and challenges that explain why image stitching cannot be implemented. Section 3 discusses the algorithm used for reducing the Ghost Artefacts by deforming the projection surface, the challenges faced and solutions arrived at. Section 4 discusses the experiments conducted, their results and observations. Section 5 concludes the thesis along with discussion on open issues and future work.

2

RELATED WORK

The problem statement as described in Section 1.4 seems to be a problem with Image Stitching. Secondly, as explained in the Section 1.2, since the images are projected assuming a flat surface, the problem could also be solved by modifying the projection surface to be placed at the correct depth of the obstacle in the vertical direction (like a screen at correct depth). For arriving at a solution to the said problem, the literature was searched in these two specific domains that are related to the problem, namely Image Stitching and Mesh Deformation. In the first sub-section, work related to Image Stitching is discussed and in the second sub-section, those related to Mesh Deformation is discussed. In the end, the approach used to solve the problem is briefly introduced.

2.1 IMAGE STITCHING

Image Stitching is a domain of Image Processing, where two images with a certain amount of overlap (both images have some common content) are "stitched" to form a composite image with a wider angle. A brief overview of image stitching is given in [Sze06].Also, the problem of *ghost artefacts* is well addressed in the image stitching domain.

The simplest form of image stitching is, to align the images and make a smooth transition from one image to the other along the border. [RLE+05] discusses such a simple and effective method for image stitching and blending with applications in medical imaging. In this method, multiple images are aligned using the normalized cross-correlation co-efficient in a small window as shown in Figure 2.1.1. The normalised cross correlation coefficient is as defined in Equation 2.1.1 [RLE+05].

$$\text{cross-correlation} = \frac{\sum_{x=0}^{L-1}\sum_{y=0}^{K-1}(w(x,y)-\bar{w})(f(x+i,y+j)-\bar{f}(i,j))}{\sqrt{\sum_{x=0}^{L-1}\sum_{y=0}^{K-1}(w(x,y)-\bar{w})^2}\sqrt{\sum_{x=0}^{L-1}\sum_{y=0}^{K-1}(f(x+i,y+j)-\bar{f}(i,j))^2}} \tag{2.1.1}$$

where, $w(x, y)$ represents a pixel value of the image to be placed, $\bar{w}$ is the mean value of all pixels included in the box area, $f(x + i, y + j)$ represents a pixel value of the composite image inside the box area, $\bar{f}(i, j)$ is the mean value of all pixels of the composite image within the box area and parameters K, L represent the box dimensions in number of pixels included. Image stitching is achieved by sliding the image

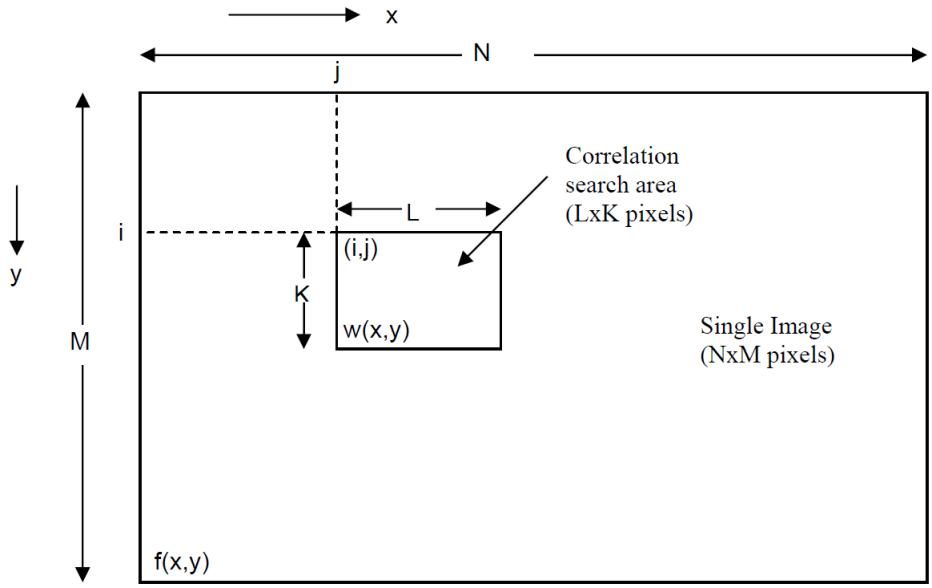


Figure 2.1.1: The general normalized cross correlation procedure is performed within a small search area - Image taken from [RLE+05]

over the composite (since this yields better results than using the reference image) so as to achieve the best cross-correlation [RLE+05]. After stitching, the seam artifacts are removed using alpha blending (thrice in case of color, by separating each color channel). The algorithm is simple and very fast in case the overlap areas are known (as in our case). The window's size (the window in which the cross-correlation coefficient is calculated) could be varied depending on the number of features available. The algorithm would fail when no correlation is found either due to a lack of texture or plane surfaces in the overlap region resulting in no features. Also, changes in intensities would be very vividly visible.

For a more intuitive appearance, the above method could be applied in the derivative domain. [ZLPW06] explains exactly this process. The derivative images are calculated using the input images. The stitching is done by minimizing a cost function in the

gradient domain. This results in a smooth transition between various input images. It is very suitable for combining different perspectives of the same object. As a result of working in the gradient domain, it is less susceptible to changes in the intensities of the images. However, it assumes a planar scene and there is an additional cost of computing the gradient images. The assumption of planar scenes is violated in our scenario.

One way of automating the alignment and stitching process is explained in [BL07]. It introduces automatic alignment and stitching of multiple unordered images under ideal imaging conditions. The algorithm uses SIFT [Low99] to extract features from all unordered images. It then finds the k nearest neighbors for each feature using a k-d tree [Ben75]. For each image, the following is done:

1. Select m candidate images that have the most feature matches to this image.
2. Find geometrically consistent feature matches using RANSAC [FB81] to solve the homography between pairs of images, and
3. Verify image matches using a probabilistic model (cf. [BL07]).

It then finds the connected components of image matches. For each connected component, the algorithm performs bundle adjustment (cf. [BL07]) to solve for the rotation angles and focal lengths of all cameras. The panorama is finally rendered using multi-band blending (cf. [BL07]). The method is suitable for automatic stitching of unordered images. The system is robust to camera zooms, orientations and changes in illumination. It however is not robust to camera motions. Since, it uses the ideal camera model; it does not take care of lens distortion or parallax. It uses SIFT that is computationally intensive. Not using SIFT nullifies the advantages stated above and sacrifices the robustness of the system, as a result.

All the methods till now do not take the error due to parallax into account. [QC07] introduces a novel method of synthesizing mosaic from non-planar scenes with parallax using a sparse set of translated cameras. The algorithm assumes identical high-definition translated cameras in an environment with controlled illumination, ignoring effects of lens distortion, vignetting, exposure and color differences. The method compensates parallax using a set of interpolated virtual images in between the original images. The final mosaic is created by cutting successive strips from each of these virtual interpolated images. The method gives good results of non-planar scenes with parallax. However, it is applicable only to common baseline cameras

with translation in only one axis. Also, the algorithm is shown to work well under controlled illumination. The effect of different illuminations (common with outdoor scenes) is not known.

In contrast, [DC04] considers generating an image mosaic for generic 3D scenes under general camera motions. In case of parallax, while stitching two images, an intermediate image is constructed, which has an overlapping area of the target image and features the non-overlapping areas of both images. The fundamental matrix is then used and the parallax is estimated between the target image - intermediate image and between the target image - reference image pairs. The mosaic is constructed by one of the following two methods:

1. an approximated mapping in 2D between the target and intermediate images, and

2. approximating the scene as a piecewise planar scene.

The second method was shown to yield better results. However, this method assumes a common baseline and a planar scene, both of which are not true in our case.

[SS07] describes the construction of panoramic image mosaic from image sequences without projecting them explicitly to a common surface. It uses two motion models namely homographies and, 3D rotations and zooms. A patch-based alignment algorithm is developed for registration of images. The focal length is then estimated and the error of misregistration is distributed evenly. A global alignment is then done by having new point correspondences and minimizing a cost function in screen coordinates. Finally, a deghosting step is performed to remove small misregistrations. The method assumes planar scenes and is a good one for automatic stitching of multiple images. It handles parallax using global and local adjustment. But, it is computationally intensive and is useful for offline stitching of images.

A huge challenge in image stitching is handling occlusions. [LLM+11] discusses a method to stitch images, which is flexible to handle parallax, has good extrapolation and occlusion handling properties. The method uses a sparse set of corner features with associated SIFT descriptors. A global affine is computed from SIFT correspondences using RANSAC. An affine stitching field is computed by repeatedly minimizing a cost function until convergence. Lastly, Poisson blending with optimal seam finding algorithm of [CE07] is used. The method results in good interpolation in regions with features. However, it resorts to the global affine in absence of features and

the extrapolation is poor as noted by [ZCBS13]. Also, it is computationally intensive as the affine field is iteratively computed.

To stitch natural landscapes, [GKB11] comes up with an idea of using two homographies per image to stitch and blend the images using a non-linear warping. It uses SIFT to detect and match feature points. The scene is then divided into two parts namely the ground plane and distant plane using Llyod's algorithm [Lloo6] for clustering. Further, RANSAC is used to robustly estimate two homographies, one each for the ground plane and the distant plane. A weighted combination of both homographies is done for every pixel. Then, seam cutting and blending is performed. In the end, a global straightening is done by using the similarity transform [GKB11]. The algorithm is more accurate than the global homography when the scene is known to consist of two distinct depths. As noted in [GKB11], it fails when there are more than two distinct depths. Also, the clustering algorithm is computationally intensive and time consuming.

[ZCBS13] introduces an innovative method of image stitching by using warps that aim to be globally projective, yet allow non-projective deviations locally to account for violations to the assumptions of the imaging conditions. It uses DLT [Haro3] as a base method. Additionally, a weight is added for every point correspondence in calculating the homography matrix. The weight of every point correspondence is calculated as a Gaussian which has more effect in the neighborhood of the point. Assume the homography between two images is given by [ZCBS13]:

$$\tilde{x}'_* = H_* \tilde{x}_* \tag{2.1.2}$$

where, H_* is estimated from the weighted problem,

$$h_* = \arg\min_h \sum_{i=1}^{N} \|w_*^i a_i h\| \tag{2.1.3}$$

subject to $\|h\| = 1$. The weights $\{w_*^i\}_{i=1}^N$ vary according to x_* according to the following:

$$w_*^i = exp(-\|x_* - x_i\|^2/\sigma^2). \tag{2.1.4}$$

Here, σ is a scale parameter and it can be seen that the weight is calculated as a Gaussian. The image is split into a grid and a homography is computed for each cell of this grid. It uses SIFT for feature correspondences and RANSAC to remove outliers.

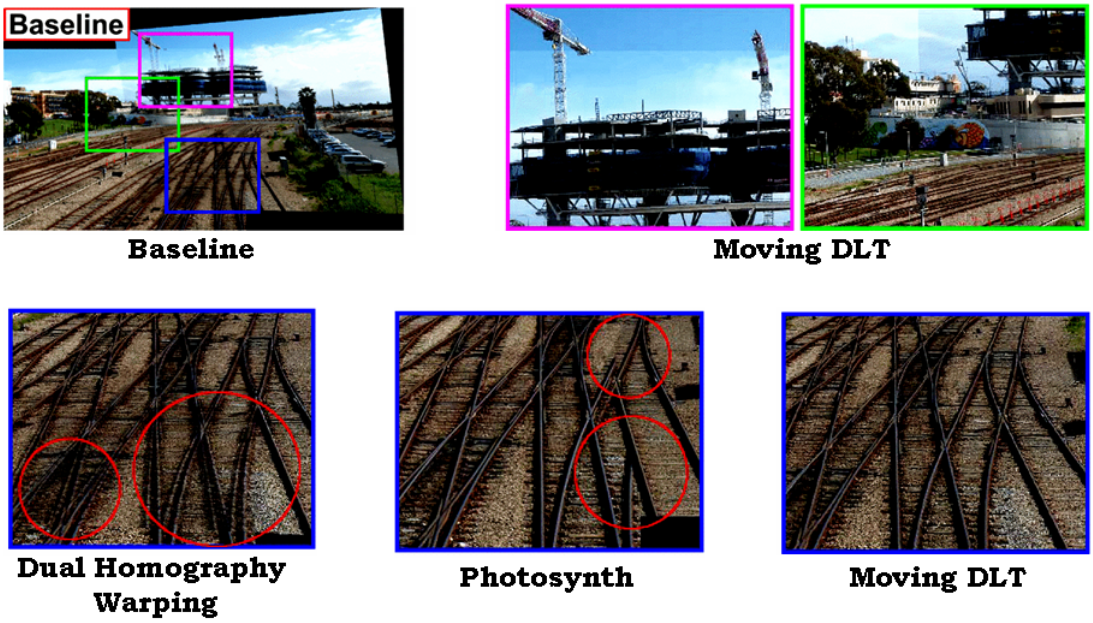


Figure 2.1.2: Results of Moving DLT [ZCBS13] along with Photosynth and Dual Homography Warping [GKB11] - Images taken from [ZCBS13]

Figure 2.1.2 shows some results of this algorithm compared with the popular stitching software Photosynth and with [GKB11] (Dual Homography Warping). As can be seen from Figure 2.1.2, the algorithm is extremely accurate in terms of image stitching quality and generic scenes can be effectively stitched irrespective of the varying depths in the scene. However, it is computationally intensive and assumes planar scenes.

As shown in the discussion section, our Surround-View setup throws certain unique challenges, that cannot be addressed by Image Stitching. Nevertheless, the available literature was studied to understand the current trend in this field and to understand if there exists a solution to deal with the current problem on hand.

2.2 MESH DEFORMATION

Meshes (triangle or polygonal) are the basic building blocks in computer graphics. In our scenario, the visualization of the images is done using a triangular mesh. The images are projected to this mesh so as to form a 3D visualization. One approach to solve the problem of *ghost artefacts* is to deform the mesh, which acts as a projection surface,

to the position of the obstacle so that the images of the two cameras are projected very close to each other. In order to try such an approach, the available literature was searched for finding out appropriate methods for deforming the projection surface. There are a lot of papers on free-form modeling and many methods on preserving fine scale details which are not required for our scenario. We require a simple and efficient method to deform a smooth mesh with no fine scale details.

The process of modeling and mathematics involved in constructing deformable curves and surfaces are explained in great detail in [CG91]. It introduces the concepts of curve energy and surface energy. The paper first details the complete system for continuous curves and surfaces and then goes on to apply the same principles to triangle meshes using Barycentric coordinates. It shows a method to build surfaces using the *ShapeWright paradigm* [CG91], where a user first draws some characteristic lines, then skins it, and finally sculpts the shape in accordance to external forces and geometric constraints. Finite elements were used to solve for the systems of equations in the discrete domain. This provides a sound mathematical understanding of the deformation process along with results of the forces that can be applied on the surfaces.

A nice survey on the available methods in mesh deformation is presented in [BS08]. It describes the basics of deformation, along with solutions to certain common challenges encountered during implementation. The paper surveys methods that generally formulate the mesh deformation as a "global variational optimization problem that addresses the differential properties of the edited surface" [BS08]. It outlines the similarities and differences between various methods for deformation and preservation of details. It helps as a good guideline for selecting a particular method depending on the scenario and problem at hand.

To enable one to deform meshes at will, [BK04] suggests a simple but effective method using “constraint shape optimization [BK04]”. The paper describes a modeling metaphor where the user can potentially set any basis function using which the mesh needs to be modified. The method minimizes an energy functional that penalizes stretching and bending while satisfying the arbitrary boundary conditions. The advantage is that it allows the user to take arbitrary boundary conditions and these are taken into account resulting into an optimal solution that is known to have a certain smoothness. This is why this approach is called “boundary constraint modeling (BCM)" [BK04]. The intension of the algorithm is to offer complete flexibility to the user to define his own basis functions which are tailored to result in an intended modification. In this modeling metaphor, the user decides a certain region of the

surface as the *support*. Then the user chooses a *manipulator object* (also called *handle*), which is a subset of the support. This means that when the handle moves, the support moves along with it. The user now decides the final position of the handle and the remaining part, that is the region of support minus the handle, needs to smoothly interpolate and bend according to the translation, rotation and scaling of the handle region. The smoothness of the output can be determined mathematically. This method can be easily adapted to discrete meshes as well. Mathematically, the continuous energy functional of a surface S is [BK04]:

$$E_k(S) = \int F_k(S_{u\ldots u}, S_{u\ldots uv}, \ldots., S_{v\ldots v}) \tag{2.2.1}$$

where the expressions S_* represent the partial derivatives of order k with respect to the surface parameterization $S : \Omega \to R^3$. This energy functional needs to be minimized. To solve equation 2.2.1, one uses variational calculus to get the corresponding Euler-Lagrange equation that characterizes the minimizers of 2.2.1. For most common quadratic functionals, the resulting differential equation is:

$$\begin{aligned} \Delta^k S(x) &= 0, x \in \Omega \setminus \delta\Omega \\ \Delta^j S(x) &= b_j(x), x \in \delta\Omega, j < k \end{aligned} \tag{2.2.2}$$

where, Δ is the Laplacian operator and b_j are the boundary conditions of order $j < k$. Discretizing this surface, for using the method on triangular meshes, implies discretizing the Laplacian operator and applying the algorithm. [BK04] describes a way of converting this problem for discrete meshes to solving a sparse linear system for the position of the free region (the region of support minus the handle). The main advantage of this method is its simplicity, arbitrary use of basis functions, arbitrary boundary conditions and ease of implementation. These advantages make it an ideal candidate for our particular case. It however, does not preserve orientations of the fine scale details and is not necessary in our case, as the surface we intend to deform is smooth and does not have any detail.

There are a lot of methods that are used to preserve the orientation of fine scale details, as intuitive as possible. [LSCO+04] advocates using differential coordinates as the means to preserve the fine scale details of the surface. To compensate for not being rotational invariant, the authors rotate the differential coordinates in an approximated local frame. This helps to edit complicated meshes while preserving the shape of fine details in their intuitively natural orientation. The algorithm for preserving local details, first applies a rough transformation to the mesh that represents the low frequency

detail of the surface. Then, it approximates the local rotations and each differential coordinate is rotated by this rotation. Finally the system of the rotated differential coordinates is solved to reconstruct the edited surface. While this method is very intuitive for fine scale details, our mesh does not contain any such detail and is just a smooth surface. Hence, the additional burden of transforming into the differential domain is not justified.

[BSPG06] also introduces another method for preserving fine scale detail while deforming meshes. It combines minimization of the variational bending energy and solving a Poisson system of differential coordinates. It involves separating the surface into a low frequency base surface and high frequency details. The deformation to the base surface is done in the gradient domain. The base surface is modified as per [BK04]; however, other methods may also be used. Then, the deformation transfer is done for the details, which preserves the gradient field of the original surface in the deformed mesh. [ZHS+05] describes deformation of large meshes using the volumetric graph Laplacian [ZHS+05] that helps avoid self-intersections during deformation and make it as intuitive as possible. The method involves constructing two volumetric graphs, an inside volumetric graph that resists changes to volume and an outside volumetric graph that prevents local self-intersections during deformations. It minimizes a weighted energy functional that also preserves the volume of the mesh being deformed. This kind of deformation is very intuitive for 3D characters. Both the methods [BSPG06] and [ZHS+05] are intuitive and advantageous for meshes with fine scale detail. But, both of them are also computationally expensive. While these methods are very intuitive in appearance, they are computationally prohibitive for our case.

Mesh deformations have also been widely studied in the morphing industry. [Ale03] uses the differential coordinate representation for mesh morphing and deformation. It uses the same advantages of differential coordinates as described in [LSCO+04] for applying local operations to meshes. The paper in particular introduces the Laplacian coordinates, which are treated as a transformation that is invariant to translations and are shown to be well suited for the modeling task. They are also made invariant to affine transformations by adding a constraint. The main advantage of the Laplacian coordinates is that they are not dependent on the transformations of the shape (that is being morphed) and can tolerate degeneracies in mesh. However, they are affected by scaling and rotations of the shape.

2.3 DISCUSSION

All papers in Section 2.1 describe good methods for image stitching. However, they assume planar scenes (where the change of depth in the scene is negligible when compared to the distance of the scene from the camera) along with pin-hole camera optics. When there are varying depths in the scene, [SS07], [LLM+11] and [GKB11] show visible artifacts as explained in [ZCBS13]. However, the method described in [ZCBS13] seems very promising. None of the methods describe image stitching when the images are from a wide angle fish-eye optic camera. Also, all the methods discussed assume ample computation power and are not capable of running in real time. This is also true for methods discussed in Section 2.2 when the vertices for deformation change with every frame. Further, due to certain challenges as discussed below, we do not use image stitching to solve the problem of *ghost artefacts*.

Scenario 1: Different ordering of poles in images
Assume the scenario as shown in Figure 2.3.1. Assume there are three poles as shown. Neglecting the effects of distortion (fish-eye optics) for the moment, the left camera would see the scene as shown in Figure 2.3.2 and the front camera would see the scene as shown in Figure 2.3.3. In this case, since the order of the poles seen in both the cameras is different, image stitching cannot be used.

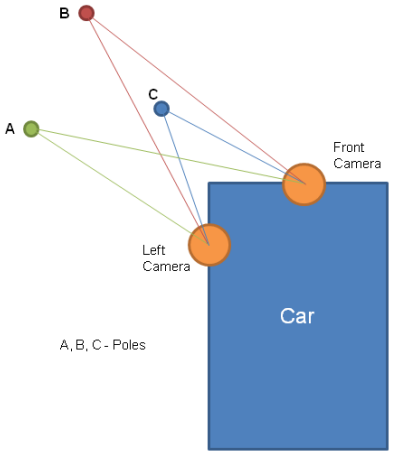


Figure 2.3.1: Scenario 1

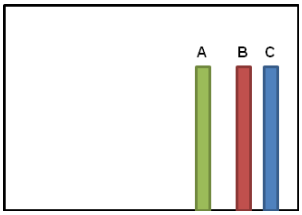


Figure 2.3.2: Image of Left Camera

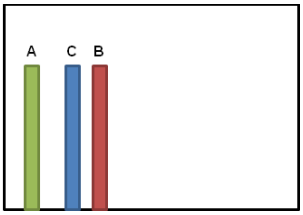


Figure 2.3.3: Image of Front Camera

Scenario 2: Occluded poles
Assume the scenario shown in Figure 2.3.4. Assume the height of pole B is less than that of A. Hence, pole B is completely hidden by pole A in the image of the left camera as seen in Figure 2.3.5. But, all three poles are seen in the image from the front camera as shown in Figure 2.3.6. Since the information in the area to be stitched is not consistent, image stitching cannot be used.

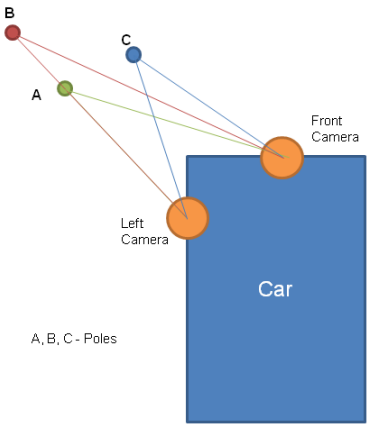


Figure 2.3.4: Scenario 2

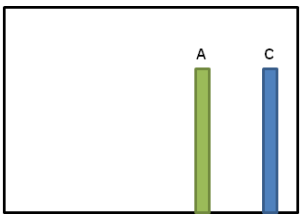


Figure 2.3.5: Image of Left Camera

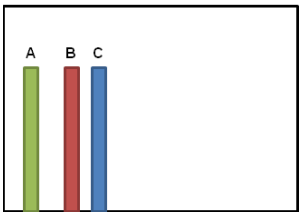


Figure 2.3.6: Image of Front Camera

Both the scenarios are very common and are generally encountered in parking situations, where this system is expected to provide assistance. As a result, it was decided to pursue the deformation of the projection surface as against image stitching. Since the surface to be deformed is smooth and without any fine scale detail, and that it needs to be run on a low end processor, it was decided to adapt the method as explained in [BK04].

3

ELIMINATING GHOST ARTEFACTS USING MESH DEFORMATION

Deformation of the projection surface has never been used to solve the problem of ghost artefacts. There have been many approaches in the image stitching domain, but, none in the mesh deformation domain to solve this particular problem. This section explains the method of deforming the projection surface as applied to eliminating ghost artefacts.

3.1 GENERAL APPROACH

As was explained in Section 1, during visualization, the projection on the regions which are visible in only one camera is as one sees the scene. However, in the overlap regions, the flat-surface assumption, used for projecting the images, is violated. Hence, the images from the adjacent cameras do not match and give the illusion of two poles (ghost artefacts - See Figure 3.1.1 for an example). The idea is to deform the

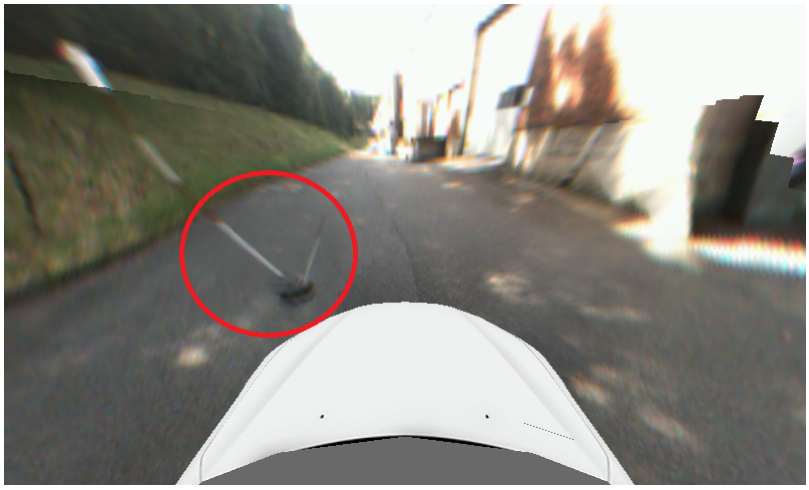


Figure 3.1.1: Visualization on the head-unit/simulator, of a parking scenario with single pole in the overlap region giving rise to ghost artefacts - Image provided by Robert Bosch Car Multimedia GmbH

projection surface so that it is at the depth of the obstacle (pole in this case) and then, the images from the adjacent cameras should coincide, thus eliminating the ghost artefact. Although, the phenomenon of ghost artefacts occurs with all obstacles in the close proximity of the car, an example of a thin pole is used in this thesis.

In the following, it is assumed that calibrated cameras with intrinsic parameter matrix, extrinsic parameter matrix and the distortion function is available for each of the four cameras. It is also assumed that reliable good correspondences are available between adjacent images (marked manually using a tool). This assumption is made to understand the ground truth of the method and to see if it is effective with known good correspondences. Later, it can be adapted to any feature matching algorithm. The task of eliminating ghost is divided into two parts namely, calculating depth from known feature correspondences and deforming the projection surface.

3.1.1 *Calculation of World Coordinates from Reliable Feature Correspondences*

Since the correspondences between adjacent images are known, these are used along with the camera parameters, and triangulated to find the depth of the corresponding point in the world. Since the cameras that are used are wide angle with fish-eye optics, the transformation from the image to the world is non-trivial. This is due to the non-linear nature of the distortion function used to model the camera. The algorithm used to arrive at the depth of obstacle from the point correspondences is as shown in Figure 3.1.2.

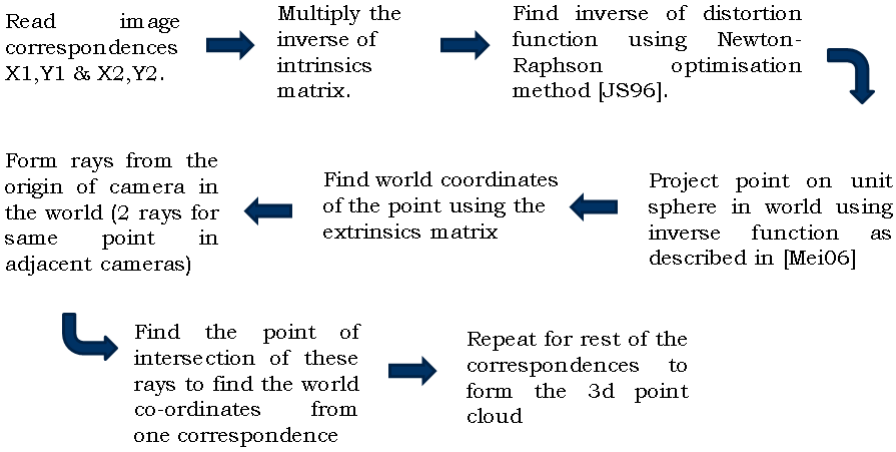


Figure 3.1.2: Algorithm for calculation of world coordinates from image correspondences

The camera model as described in [Mei06] is used as reference, as the same is used for calibration. As shown in Figure 3.1.2, first the image correspondences are read. Then, these are transformed to the camera's sensor coordinate system using the inverse of the camera intrinsic matrix as per Equation 3.1.1.

$$\begin{bmatrix} X_s \\ Y_s \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & O_x \\ 0 & f_y & O_y \\ 0 & 0 & 1 \end{bmatrix}^{(-1)} * \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{3.1.1}$$

where, $[X, Y, 1]'$ are the pixel coordinates on the image and $[X_s, Y_s, 1]'$ are the corresponding coordinates in the sensor coordinate system.

Next the distortion must be removed. This is done using the inverse of the distortion function. The distortion function modeled, is continuous and is differentiable twice (see Equation 3.1.2). Hence, its inverse is calculated by using the iterative Newton's method [JS96], assuming that the solution lies in between $[-1, -1]^T$ and $[1, 1]^T$. This is because the distorted pixel coordinates of the corners of the image, on multiplication with inverse of the camera intrinsic matrix lie between $[-1, -1]^T$ and $[1, 1]^T$, the undistorted pixels must also lie within the same range in the worst case.

Assume the function of distortion to be of the form,

$$\begin{aligned} x_{dis} &= f(x_{undis}, x_{undis}^2, x_{undis} * y_{undis}) \\ y_{dis} &= f(y_{undis}, y_{undis}^2, x_{undis} * y_{undis}) \end{aligned} \tag{3.1.2}$$

where, (x_{dis}, y_{dis}) are pixel coordinates of the distorted image, (x_{undis}, y_{undis}) are the pixel coordinates of the undistorted image and $f(.)$ is a mapping between the two pairs. On rearranging terms, Equations 3.1.2 can be written as,

$$\begin{aligned} x_{dis} - f(x_{undis}, x_{undis}^2, x_{undis} * y_{undis}) &= 0 \\ y_{dis} - f(y_{undis}, y_{undis}^2, x_{undis} * y_{undis}) &= 0 \end{aligned} \tag{3.1.3}$$

Assuming an initial solution as $X_0 = [0, 0]^T$, the system 3.1.3 can be solved iteratively for a given tolerance on error, as follows:

1. Calculate the function values of f at the guessed value $X_i = X_0$ to give $G = [g_1, g_2]^T$ by evaluating 3.1.3.
2. Taking only first order partial derivatives into account for simplicity, the Jacobian matrix is constructed as,

$$J = \begin{bmatrix} \frac{\partial g_1}{\partial x_{undis}} & \frac{\partial g_1}{\partial y_{undis}} \\ \frac{\partial g_2}{\partial x_{undis}} & \frac{\partial g_2}{\partial y_{undis}} \end{bmatrix} \text{ evaluated at } X_i. \tag{3.1.4}$$

3. Solve the linear system $-G = J * dx$, where $dx = [x_{undis}, y_{undis}]^T$, for dx.
4. Update the guess by dx, yielding $X_{i+1} = X_i + dx$, where i is the iteration count.
5. Continue until the squared norm of $G = [g_1, g_2]^T$ evaluated at the new value X_{i+1} is less than a required tolerance on error.

The result of the above procedure is a point that is transformed from the distorted image coordinates to the undistorted image coordinates in the camera's sensor coordinate system.

Next, using the inverse function of the camera model as described in [Mei06], the points are transformed from the plane image to the unit sphere using Equation 3.1.5. Here, the transformation is from the two-dimensional image plane to the three dimensional world in the camera's coordinate system.

$$h^{(-1)} \begin{bmatrix} X_u \\ Y_u \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\xi + \sqrt{1+(1-\xi^2)(x_u^2+y_u^2)}}{x_u^2+y_u^2+1} x_u \\ \frac{\xi + \sqrt{1+(1-\xi^2)(x_u^2+y_u^2)}}{x_u^2+y_u^2+1} y_u \\ \frac{\xi + \sqrt{1+(1-\xi^2)(x_u^2+y_u^2)}}{x_u^2+y_u^2+1} - \xi \end{bmatrix} = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{3.1.5}$$

where, $[X_u, Y_u, 1]'$ are the undistorted pixel coordinates in the sensor coordinate system of the camera and $[X_c, Y_c, Z_c]'$ are the world coordinates in the camera's coordinate system for the corresponding pixel $[X_u, Y_u, 1]'$.

Finally, the point is transformed into the world coordinate system using the inverse

of the camera extrinsic matrix. This is done for each pixel that is part of an image correspondence using Equation 3.1.6.

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = R^{(-1)}_{(3x3)} * \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} + T_{(3x1)} \tag{3.1.6}$$

where, $R_{(3x3)}$ is the (3x3) orthonormal rotational matrix and $T_{(3x1)}$ is the (3x1) translational vector.

Then, rays are constructed using the world coordinates of pixels in one image from the origin of the respective camera. For a point correspondence, there are two rays

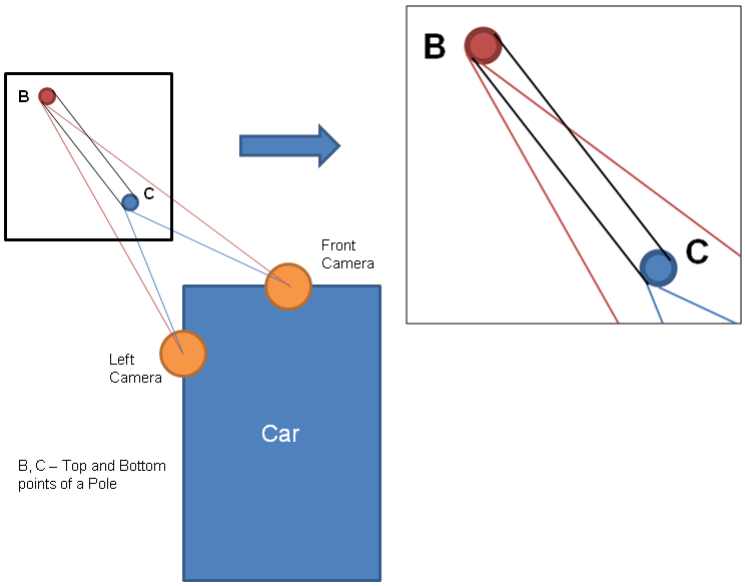


Figure 3.1.3: System of skew rays using two cameras for two points on the pole (top and bottom) as obstacle

from adjacent cameras, that intersect. This intersection point is the world coordinate of the point correspondence. On the edges of the obstacle (for example, the pole), the adjacent cameras never see the same point in the world due to different perspectives, although there exists feature correspondence between these two points in the image. As a result, the two rays never intersect. In fact, they always form a pair of skew lines

as depicted in Figure 3.1.3. The mid-point on the line of the shortest distance between these two skew lines (which is necessarily perpendicular to both lines) is calculated as follows and is used as the point of intersection.
Consider the two rays of a particular point correspondence to be:

$$\begin{aligned} L1 : r_1 &= p_1 + t * v_1 \\ L2 : r_2 &= p_2 + s * v_2 \end{aligned} \tag{3.1.7}$$

where, v_1 and v_2 are the direction vectors of the rays, p_1 and p_2 are two points on the respective rays, and t and s are scalars. To find the shortest distance between these two lines, it suffices to find scalars t and s that minimize the squared norm $(r_1 - r_2)$. Substituting for r_1 and r_2 from 3.1.7 into $\|r_1 - r_2\|^2$, we get:

$$[v_1, -v_2] * [t, s]^T - [p2 - p1] = 0 \tag{3.1.8}$$

The coordinates of the midpoint of this line of shortest distance is given by,

$$(p_1 + t * v_1 + p_2 + s * v_2)/2 \tag{3.1.9}$$

This is repeated for every point correspondence to result in a 3D point cloud in the world. This point cloud indicates the location of the nearest obstacle in the world coordinates (assuming the nearest obstacle had feature correspondences).

3.1.2 *Deformation of Projection Surface to eliminate Ghost Artefacts*

Once the location of the nearest obstacle is decided, the next task is to deform the projection surface (the bowl) so that no ghost artefacts are visible. To achieve this, the method as described in [BK04] and [BS08] is adapted ([BS08] has solved the problem with geometry of meshes, it has never been used to solve the problem of eliminating ghost artefacts by deforming projection surfaces), as discussed in Section 2.

The method described in [BS08] explains a way of deformation as minimizing the thin shell energy. The mathematical model described, resists the changes to the fundamental forms of the surface giving an intuitive deformation. Before discussing it further, the following definitions of surface regions are necessary to understand the method:

- Fixed Region - Region of the mesh that is unaffected by the deformation and retains the original shape.

- Handle Region - Region of the mesh for which we know the position, or the displacement from the original position, that is to be maintained after the deformation.
- Free Region - The Region in between the Handle and Fixed Regions for which the deformation needs to be intuitively carried out. The positions or displacements of the mesh part for this surface needs to be calculated so as to result in an intuitive deformation.

Let us first assume a continuous surface. Let the surface S be deformed to surface S'. Then the change of fundamental forms results into [BSo8]:

$$E_{(Shell)}(S') = \int_{\Omega} k_s \|I' - I\|_F^2 + k_b \|II' - II\|_F^2 du dv \tag{3.1.10}$$

where, $E_{(Shell)}(S')$ is the shell energy of the modified surface, $\Omega \subseteq S$ is the surface undergoing deformation, $I(u,v)$, $II(u,v) \in R^{2x2}$ represent the fundamental forms of surface S, $I'(u,v), II'(u,v) \in R^{2x2}$ represent the fundamental forms of surface S' and $\|*\|_F$ represents the Frobenius norm. The first fundamental form denoted by $I(u,v)$ allows one to calculate curvature and metric properties of the surface (length and area). The second fundamental form denoted by $II(u,v)$ along with the first fundamental form defines the principal curvatures of the surface. The stiffness parameters k_s and k_b are used to vary the resistance to stretching and bending [BSo8]. This system needs to be minimized to yield an intuitive deformation. However, Equation 3.1.10 is a non-linear optimization problem and is computationally very expensive for real-time applications. Hence, to simplify, the change of first and second fundamental forms is replaced by the first and second order partial derivatives. This problem can be solved efficiently using variational calculus so as to yield the following *Euler-Lagrange Partial Differential Equation* [BSo8].

$$-k_s \Delta d + k_b \Delta^2 d = 0 \tag{3.1.11}$$

Equation 3.1.11 is analogous to the minimization problem of Equation 3.1.10, where Δ and Δ^2 are the Laplacian and bi-Laplacian operators respectively and d is the displacement function (d can be absolute positions of the handle on S' or displacement of the vertices from their initial position on S).

Since Ω is defined to be a subset of the original surface S, and additionally choosing d on S itself, reduces Equation 3.1.11 to the following [BSo8]:

$$-k_s \Delta_s d + k_b \Delta_s^2 d = 0 \tag{3.1.12}$$

where, the Laplacian operator Δ reduces to the Laplace-Beltrami operator Δ_s.

To discretize the above setup to a triangle mesh, two approaches are available. The Finite Element Method (FEM) and the Finite Differences. As per [BSo8], FEM gives better approximations, however has some restrictions on the initial surface. In comparison, [BSo8] suggests use of finite differences as it is simpler to use. This involves discretizing only the Laplace-Beltrami operator of Equation 3.1.12. Assume that a piecewise scalar function $f : S \to R$ is defined on the initial surface S, the discrete Laplace-Beltrami for a vertex v_i can be written as [BSo8]:

$$\Delta_s f(v_i) = w_i \sum_{v_j \in N_1(v_i)} w_{ij}(f(v_j) - f(v_i)) \tag{3.1.13}$$

where, $v_j \in N_1(v_i)$ are the connected one-ring neighbours of v_i (cf. Figure 3.1.4). The discretization depends on the normalization weights w_i and the edge weights $w_{ij} = w_{ji}$ (see Section 3.1.3 for details).

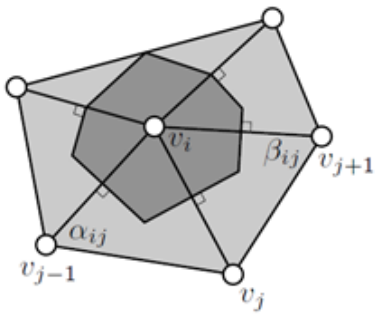


Figure 3.1.4: Discretization Weights, One ring neighborhood of v_i, one of the neighbors v_j, the dark gray area indicates the *Voronoi Area* - Image taken from [BSo8]

Using Equation 3.1.13, the Laplace-Beltrami operator can be written in the following way as [BSo8]:

$$\begin{bmatrix} \Delta_s f(v_1) \\ . \\ . \\ . \\ \Delta_s f(v_n) \end{bmatrix} = M^{-1} L_s \begin{bmatrix} f(v_1) \\ . \\ . \\ . \\ f(v_n) \end{bmatrix} \tag{3.1.14}$$

where, M is a diagonal matrix of normalization weights with, $M_{ii} = \frac{1}{w_i} = A_i$, the Voronoi area (discussed in Section 3.1.3) and L_s is a symmetric matrix defined as follows [BS08]:

$$(L_s)_{ij} = \begin{cases} \sum_{v_k \in N_1(v_i)} w_{ik}, & i = j, \\ w_{ij}, & v_j \in N_1(v_i), \\ 0, & \text{Otherwise} \end{cases} \tag{3.1.15}$$

The Equation 3.1.12 now becomes a sparse n x n linear system, where, n is the total number of mesh vertices:

$$(-k_s L + k_b L^2)d = 0 \tag{3.1.16}$$

The boundary conditions (the constraints of the fixed and handle regions) are incorporated into Equation 3.1.16 and the corresponding non-zero right hand side equation system is solved.

3.1.3 *Decision on Parameters*

The above section describes the general approach that has been followed. To adapt this approach to our case, certain values need to be decided for the various parameters that are used. The choices and the justification of the same are discussed in this sub-section.

Discretization weights

There are many variants of discretizing the Laplace-Beltrami operator. The defacto standard is the cotangent discretization as described in [DMSB99], [MDSA03] and [PP93]. This is described by the following definition for weights:

$$w_i = \frac{1}{A_i}, w_{ij} = \frac{1}{2}(\cot\alpha_{ij} + \cot\beta_{ij}) \tag{3.1.17}$$

where, α_{ij} and β_{ij} are the angles on opposite sides of the edge (v_i, v_j), and A_i is the voronoi area (see Figure 3.1.4) as described in [MDSA03]. Also, more often, the *uniform Laplacian* as used in [KCVS98], [LSCO+04], [SCOL+04] and [SZGP05] is commonly employed owing to simpler computations. The definition of weights used are:

$$w_{ij} = 1, w_i = \frac{1}{\sum_j w_{ij}} \tag{3.1.18}$$

Since this discretization does not take the geometry or position of vertices in the neighborhood into account, it is an approximate method and works well only for regular meshes, which is true in our case. To decide which discretization should be used, a small experiment was performed.

A regular grid of 50 X 50 was constructed. An elliptical handle region was selected. At first a deformation of 5 (10% of one side) units was chosen for both the methods. The method with a uniform Laplacian gave intuitive results, whereas, this was not the case with cotangent weights. To check when would the method with a uniform Laplacian fail, the deformation was increased upto 100 units and the deformation was still intuitive, mainly because of the regularity of mesh. The results of deformation of 100 units with a uniform Laplacian and that of 5 units with a cotangent discretization are as shown in Figures 3.1.5 and 3.1.6. The blue vertices are the handle, the magenta vertices (and black vertices not included in the equation system for reducing the number of computations) are the fixed vertices and the red vertices are the free vertices.

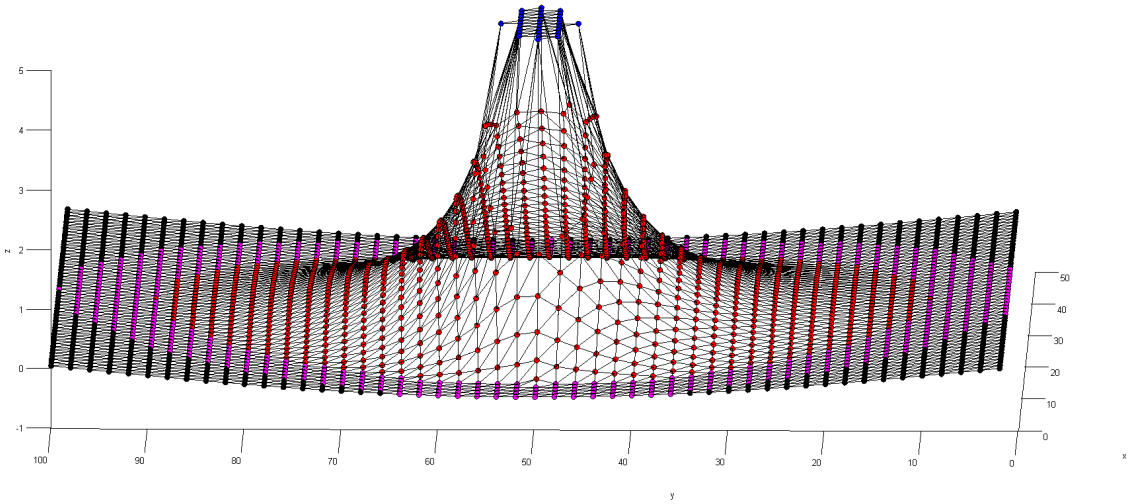


Figure 3.1.5: An example of Cotangent Discretization on an experimental regular mesh of 50 x 50

As one can see from Figures 3.1.5 and 3.1.6, the cotangent discretization does not perform as well as the one of uniform Laplacian. It was also observed that the cotangent discretization worked similar to the one with uniform Laplacian weights upto only 0.5

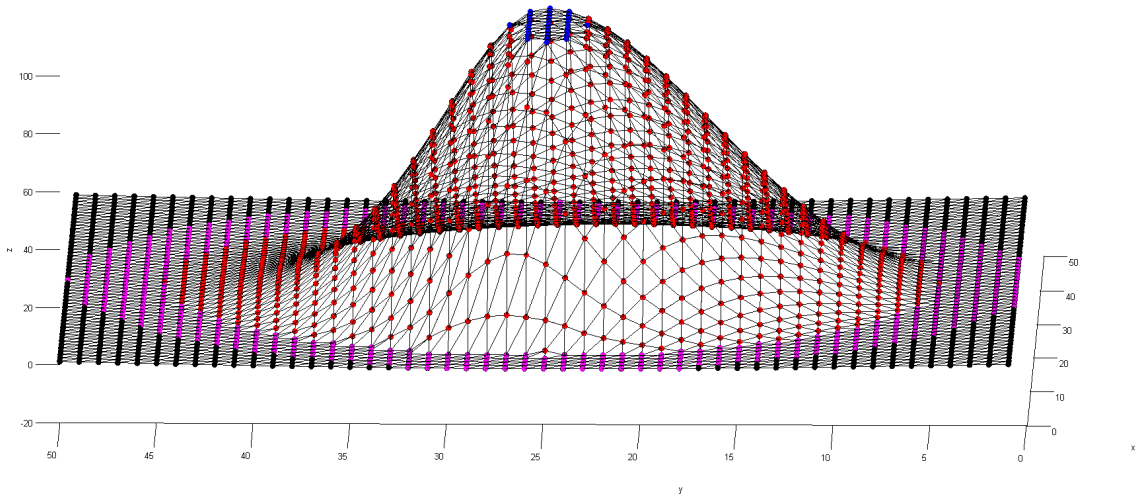


Figure 3.1.6: An example of Uniform Laplacian Discretization on an experimental regular mesh of 50 x 50

units. Hence, 10 iterations each with increasing height of handle were made with the deformed mesh as input to the next iteration in the case of cotangent discretization. The result is as shown in Figure 3.1.7. As can be seen, this produces a similar result as the deformation by Uniform Laplacian weights.

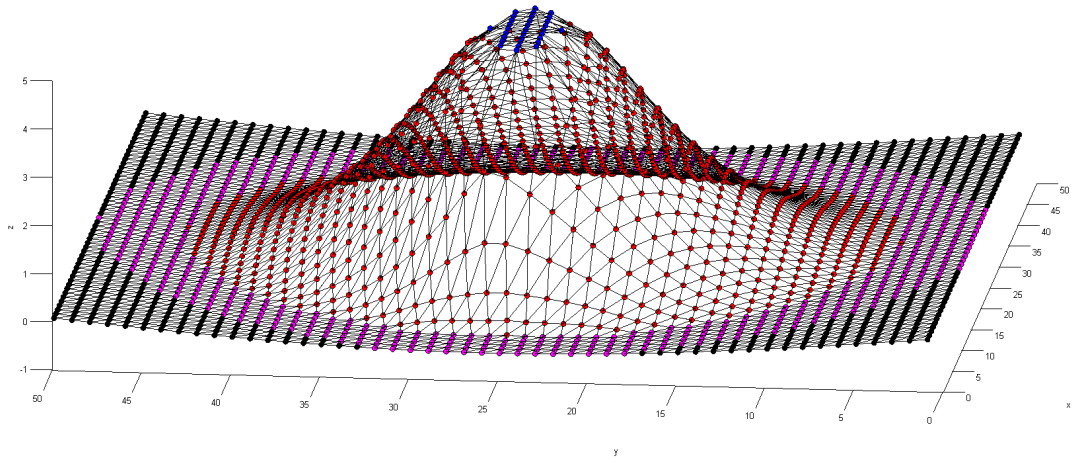


Figure 3.1.7: Same example of Cotangent Discretization, but with 10 iterations of 0.5 units increase in deformation with every iteration

Also, the execution times for Uniform Laplacian was 30 seconds and that for cotangent discretization weights was 64 seconds/iteration. Looking at these results and the fact that our mesh is a uniform grid, it was decided to use the Uniform Laplacian weights.

Stretching and bending coefficients

As per [BS08], if $k_b = 0$, then the continuity of surface would be C_0. This is highly undesirable, especially when the surface is used for texturing. This is because the textures would get stretched when continuity changes abruptly. This has been also confirmed on the same regular grid and the result can be seen in Figure 3.1.8. Values

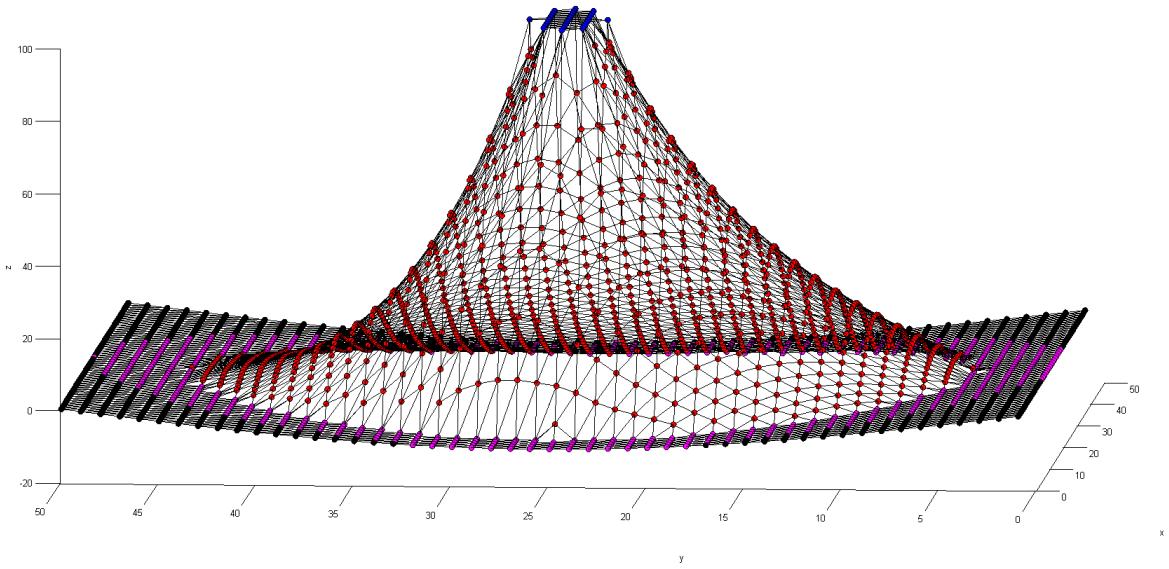


Figure 3.1.8: $k_s = 1, k_b = 0$ Note the loss of continuity between the transition from free region to handle region

of 0, 1, 10, 100 and 1000 were tried on the regular grid and our case scenario. No visible difference was observed except when $k_s = 0$ or when $k_b = 0$ for the regular mesh. In the actual mesh, a difference was observed for cases with $k_s = 0$, and for $k_s = 1$, $k_b = 1000$. Hence, values of $k_s = 1$ and $k_b = 10$, as suggested in [BS08], are used.

Handling Constraints

There are two approaches to include the boundary constraints, namely hard constraints and soft constraints. In hard constraints, the known variables are shifted to the right hand side by deleting the respective rows and columns of the system matrix. In soft constraints, the known variables are added as additional equations to the system and the resulting overdetermined system is solved.

Consider the example shown in Equation 3.1.19.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & j \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} m1 \\ m2 \\ m3 \end{bmatrix} \tag{3.1.19}$$

Assume that the value of y is known. When hard constraints are implemented, the equation system 3.1.19 would become the following:

$$\begin{bmatrix} a & c \\ g & j \end{bmatrix} * \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} m1 - b * y \\ m3 - h * y \end{bmatrix} \tag{3.1.20}$$

When soft constraints are implemented, the equation system 3.1.19 would become the following:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & j \\ 0 & \lambda & 0 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} m1 \\ m2 \\ m3 \\ \lambda * y \end{bmatrix} \tag{3.1.21}$$

where, λ is a constant that can be varied depending on how much the constraint should be weighted and satisfied. As explained in [BS08], hard constraints are preferred due to better condition number of system matrix and better numerical stability as compared to use of soft constraints. Hence, hard constraints are implemented.

Handle, Fixed and Free Regions

The handle, fixed and free regions are selected based on the implementation method and the visualization results on the simulator. Hence, these were decided by experiments based on visualization results and are discussed in Section 4.3. Detailed description of the methods are described in Section 3.2

3.2 SELECTION OF FREE, HANDLE AND FIXED REGIONS FOR THE DEFORMATION ALGORITHM

The method for the selection of free, handle and fixed regions is discussed here. There were two methods that were promising. The first one being a naive approach and the second one extended the first.

3.2.1 *Deformation Method without including the Ground Plane*

The 3D point cloud generated from the image correspondences form the basis of the handle. The idea is, all vertices of the mesh that are in the same radial direction as the world coordinates of the nearest obstacle are treated as the handle vertices. And the extent of their spread marks the handle sector (see Figure 3.2.1 for an illustration). The closest world coordinate of the correspondence in the radial direction to the center of the bowl, marks the position of the handle. The complete handle region is made to have this position. The maximum height of the point cloud decides the height of the handle region. Parameters, $\alpha_{\mathtt{free}}$ and $\alpha_{\mathtt{fixed}}$ were used for angle of free sector on both sides of the handle sector, and for the angle of fixed sector on both sides of the free sector respectively.

In this method, the ground plane of the bowl was not included in the deformation. Also, the fixed and free sectors extended for a certain fixed height of the bowl. When these were extended to cover the complete height of the bowl, certain undesirable black areas were seen, which is explained in the Section 3.3. The results of this method are as shown in Section 4.5. These results were very promising but, had certain artefacts on the ground surface, which was expected, as the ground plane was not included in the deformation. The results also proved that the direction was correct. And hence, this paved the way for the deformation method including the ground plane.

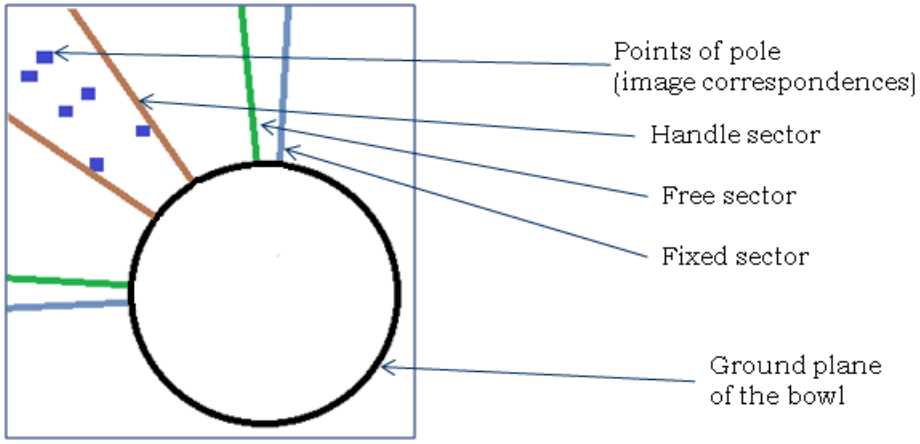


Figure 3.2.1: Illustration of spread of free, fixed and handle vertices - Deformation Method without including the ground plane

3.2.2 *Deformation Method including the Ground Plane*

Here, the idea of deformation method without including the ground plane is carried forward. Here however, the ground plane is included in the deformation. Once the 3D point cloud is calculated, the depth of the handle region is arrived at in the same way as with the previous method. Now, all the vertices beyond the handle depth on the flat surface are interpolated to the vertical line in the gap between the flat surface and the first vertex on the curve of the wall (see Figure 3.2.2 for an illustration). The free and fixed regions are also decided in the same way as in the previous method. However additionally, parameters are used to decide the height of the free region, and the fixed region extends to the height of the bowl. The results of this method are discussed in the Section 4.6. This method yields promising results but also throws certain challenges which are discussed in the next section.

3.3 CHALLENGES AND IMPLEMENTATION DETAILS

In the following, a couple of challenges that were faced during the course of implementation of the methods described above, are discussed. These deal with certain practicalities of the setup and results show improvement in visualization after dealing with the challenges.

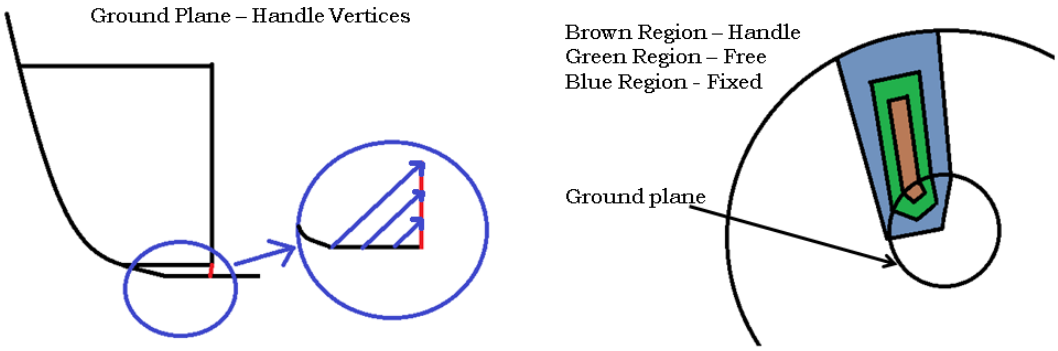


Figure 3.2.2: Spread of free, fixed and handle vertices - Deformation Method including the ground plane, the left figure illustrates interpolation of flat surface to the vertical plane for the handle region, the right figure illustrates the use of ground plane in the deformation algorithm

3.3.1 *Failure of Deformation in certain frames*

When the algorithm was run on a sequence of frames, in some frames, it was observed that no deformation occurred, even though image correspondences data was present and world coordinates were calculated. The problem was with respect to the resolution of the bowl in the horizontal direction. A small pole of $8-10$ cm in diameter and 100 cm in height would subtend an angle of 0.5 degrees for the handle region when it is very close to the car. This was less than the resolution of the bowl. Also, in case there were no correspondences near the top of the pole, it would not be included as handle in the deformation process and would be marked as free region.

To resolve this, a default handle angle and default height was chosen. The default handle angle was decided to be a little more than the resolution of bowl. In case spread of the 3D point cloud was more than this default value, the calculated value would be used, else default value would be in effect. Similarly, the default height was chosen depending on the visualization on the simulator. If the height as determined by the 3D point cloud was beyond this default height, the calculated value would be used, else the default height would be in effect.

3.3.2 *Black Areas in Frames after deformation is accomplished*

When the algorithm was run on a sequence of frames, some frames contained a black patch in the deformed region instead of image data. This was because, during deformation, some of the deformed area would go outside the visible frustum of the camera. Figure 3.3.1 shows the visible frustum of the setup. For simplicity, only the planes that limit the visible portion (top planes) are pictured. The bottom planes intersect inside the car and are not interesting here. Once the area is outside the

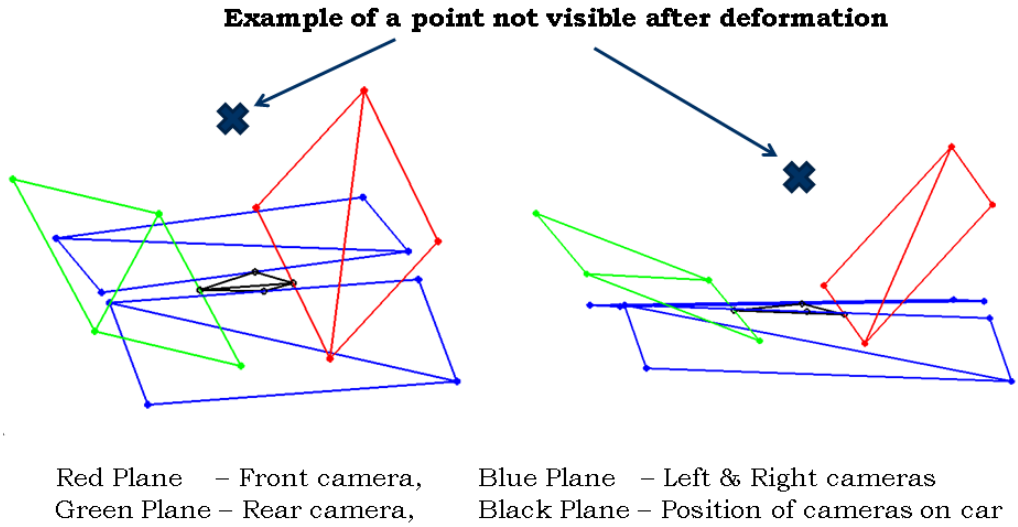


Figure 3.3.1: Visible frustum of cameras as mounted on the car, the combined volume below the planes is the visible frustum. Everything above is not visible

visible frustum of all the cameras, it cannot be textured and hence would be rendered transparent (with alpha value = 0). Since, the background color used was black, this resulted in a black region inside the frame. Two solutions were tried for the said problem namely, constrained deformation and deformation constraining the free region to handle before deformation, which are discussed in the following.

Constrained Deformation

Assume that the visibility is constrained by only two cameras, as only two cameras contribute to the region in the overlap area of the bowl. These constraints can be framed as equations of a plane with the form:

$$\begin{aligned} a_1x + b_1y + c_1z + d_1 &= 0 \\ a_2x + b_2y + c_2z + d_2 &= 0 \end{aligned} \tag{3.3.1}$$

Equation 3.1.16 with the hard constraints implementation would be as follows:

$$(-k_sL' + k_bL'^2)d = b \tag{3.3.2}$$

where, b is non-zero due to shifting of known variables to the right hand side and L' is the same matrix as L with the rows and columns of known variables (constraints) deleted. To implement the visibility constraint, the Equations 3.3.1 and 3.3.2 need to be solved together. However, there is no direct solver available to solve such a constrained optimisation. Hence, the following solution was worked out.

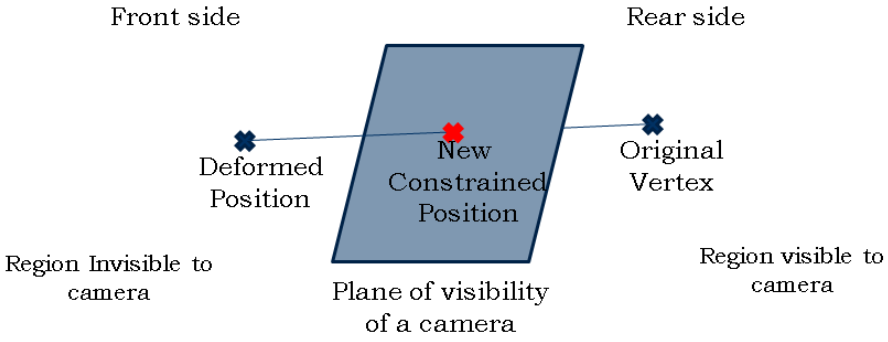


Figure 3.3.2: Constrained Deformation - The vertex moving away from the visible frustum of camera is constrained to the visible frustum using the intersection of the plane of frustum and the ray joining the original vertex and its deformed position

The situation is explained with a single camera for simplicity as shown in Figure 3.3.2. Assume a vertex is present on the visible region of the camera. Consider that, as a result of the deformation, it is shifted to a region not visible to the camera. The position of this vertex is now forced to be at the intersection of the plane of the camera frustum and the line joining the original vertex to its deformed position. When the vertex is not visible in both cameras, the intersection with the least distance to the

deformed vertex is chosen. With this, the constraint of continuity (C_1 as dictated by [BSo8]) is lost, and the textures appear to be stretched and blurred, but the black regions are eliminated. The results are shared in Section 4.6.

Deformation Constraining the Free Region to the Handle before deformation

Here, before performing the deformation, all the vertices in the free region in the same radial direction as the handle were made to take positions similar to handle (at the same distance as handle vertices with original Z coordinates). Then, the constraint of Constrained Deformation was applied to this new handle region alone. With this as handle, the equation system was solved to give an intuitive deformation that is bound by the visible frustum, automatically. This is because upon deformation, none of the parts of mesh go further than the handle. This method gives far less region of C_0 continuity than the Constrained Deformation. This is because a very small portion of the free region gets constrained to be flat (the same sector as the handle region), whereas, in Constrained Deformation, all the vertices of free region violating the constraints of frustum had to be constrained. This resulted in a larger patch of surface to be flat on a C_1 surface.

To summarize, the complete algorithm is as follows:

1. Calculate world coordinates from feature correspondences between images.
2. Decide on the handle, free and fixed regions.
3. If method is constrained deformation,
   - Solve the equation system.
   - Apply the constraints of camera frustum.
   - Replace the free vertices with the solution.
   - Reform the bowl and put the image content on it.

   else
   - Apply constraint of camera frustum to free region in the same radial direction as the handle region.
   - Use these constrained vertices as new handle.
   - Solve the equation system.
   - Replace the free vertices with the solution.
   - Reform the bowl and put the image content on it.

4

EXPERIMENTS, RESULTS AND OBSERVATIONS

This section is dedicated to explanations of the various experiments conducted, the results and discussions there off, in order to understand the strengths and limitations of the algorithms used.

4.1 DATASET USED

Sequence	Total Number of Frames	Number of Frames Used	Total Time of Sequence	Kind of Obstacle
Real World Sequence 1	174	27 and 15	18s	Pole in Front-Left overlap region and Back-Left overlap region
Real World Sequence 2	208	22	21s	Pole in Front-Right overlap region
Real World Sequence 3	31	31	3s	Pole in different overlap regions with 10 correspondences each
Real World Sequence 4	169	16	17s	Crate in Back-Left overlap region with 6 correspondences each
Synthetic Sequence 1	180	22	10s	Pole in Front-Left overlap region
Synthetic Sequence 2	180	22	10s	Pole in Front-Right overlap region

Table 4.1.1: Dataset used for all experiments

Data forms an integral part of all experiments. For a systematic evaluation of any algorithm, the data is the most important part and needs to be accurate enough. For all the experiments, four real world sequences and two synthetic sequences were used. The images of the real world sequences were captured using the surround view system mounted on the Opel Insignia car, by driving it around the Bosch campus. The synthetic sequences were generated using MATLAB® and the camera model (intrinsic matrix, extrinsic matrix and the distortion function) as described in Section 4.4. The Table 4.1.1 gives more details on the dataset used.

4.2 QUANTITATIVE MEASUREMENT OF HUMAN PERCEPTION OF VISION - THE SSIM INDEX

For making a quantitative assessment of the visualizations, an algorithm as described in [WBSS04] was used. [WBSS04] introduces a quantitative measure for measuring human perception of vision, namely the Structural Similarity Index (SSIM Index). The algorithm uses a quality assessment method based on degradation of structural information. Structural information is all the information in a scene that does not depend on illumination. In other words, on an image, it is all the information (including edges, corners and their dependencies) that are not dependent on constrast and luminance on the image. The SSIM Index gives a more realistic picture of human perception and is rated a better measure than Peak Signal-to-Noise Ratio (PSNR) and Mean Squared Error (MSE) [WBSS04]. The algorithm measures the similarity between two images based on the structural information and gives an Index value between 0 and 1. An SSIM Index of 0 means no similarity and an SSIM Index of 1 means identical images in terms of structure.

4.3 DECISION ON PARAMETERS OF THE FIXED, HANDLE AND FREE REGIONS

The parameters used for the angles of handle, free and fixed regions, the heights of these regions and the distance of these regions on the flat surface (for details, refer to Section 3.2.2), were finalized using the visualization as an end result. The parameters were modified with multiple experiments and finalized so as to give the best possible visualization to the user.

Different values were tried for the angles and heights of the free, handle and fixed regions for the methods described in Section 3.2.1 and 3.2.2. From the visualizations, it was clear and required, to use a huge angle (α_{free}) for the free region (of the order of

tens of degrees) and a small default angle (less than 10 degrees) for the handle region (α_{handle}). For the fixed region, the results were compared between using the rest of the mesh as fixed region and using a small angle of fixed region on both sides. No difference was observed in both the solutions, however, using rest of the mesh for solving the linear system took longer to solve. Also, using the rest of the mesh makes the system more overdetermined, due to an increased number of constraints. With the above two facts, it was decided to use a very small angle (less than or equal to 10 degrees) for the fixed region (α_{fixed}).

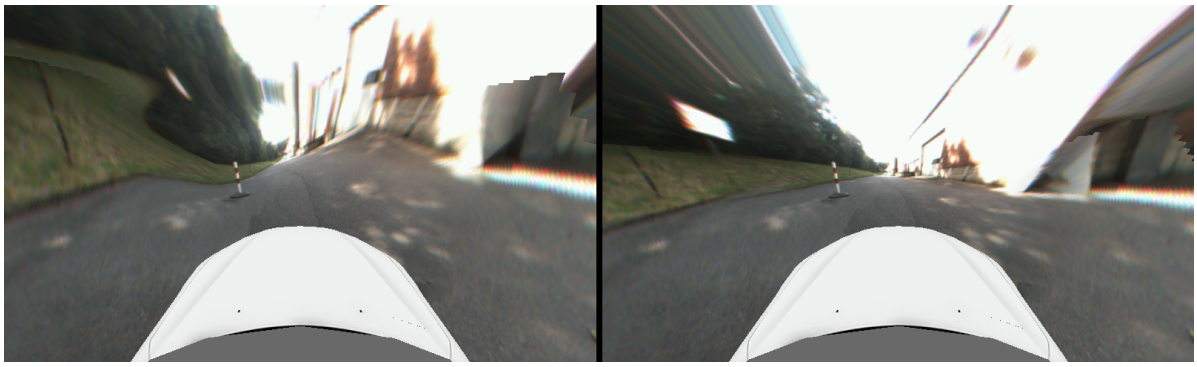


Figure 4.3.1: Visualizations with different parameter selections for free, handle and fixed regions - Left image shows visualization with $\alpha_{free} = 30^o$, $\alpha_{fixed} = 5^o$, $Height_{handle} = 1.5m$, $Height_{free} = 4.5m$ and $Height_{fixed} = 1m$, Right image shows visualization with optimized values, $\alpha_{free} = 70^o$, $\alpha_{fixed} = 10^o$, $Height_{handle} = 4.5m$, $Height_{free} = 12.5m$ and $Height_{fixed} = 4m$ - Image provided by Robert Bosch Car Multimedia GmbH

For heights of different regions, it was observed that, the handle region needs a substantial default height to help result in a good visualization (in our scenario $Height_{handle} = 4.5m$). An adequate height (12.5m above the handle) and distance on flat ground (1m) was given for free region in agreement with the huge angles provided to have a smooth and a symmetrical deformation and also to result in a good visualization. Similarly, a comparatively smaller height (4m) and distance (1m) was required for the fixed region. The values arrived at, for the various heights and angles have been concluded by looking only at the resulting visualizations on the simulator. A visualization of the result using the smaller angles with smaller heights ($\alpha_{free} = 30^o$, $\alpha_{fixed} = 5^o$, $Height_{handle} = 1.5m$, $Height_{free} = 4.5m$ and $Height_{fixed} = 1m$ as an example) and optimized angles with higher heights ($\alpha_{free} = 70^o$, $\alpha_{fixed} = 10^o$, $Height_{handle} = 4.5m$, $Height_{free} = 12.5m$ and $Height_{fixed} = 4m$) is as shown in

Figure 4.3.1. As can be seen from the Figure 4.3.1, the left image looks definitely less pleasing than the right. Hence, the parameters used for generating the visualization on the right have been used in all the further experiments.

4.4 ACCURACY OF WORLD COORDINATES

To understand how good the calculation of the world coordinates from image correspondences is, synthetic images were generated. A simple example of an obstacle is a pole, which can be assumed to be approximated by a cylinder of a certain diameter and height and having horizontal stripes so as to generate enough points for point correspondences. It was decided to generate an artificial 3D pole. In the perspective of the camera, this cylinder can be assumed to be a rectangle on a plane perpendicular to the camera's viewing direction. Therefore, individual point clouds were constructed (one for each camera), given the coordinates of center of base, radius of base, height of the cylinder and number of stripes on the cylinder, as shown in Figures 4.4.1 and 4.4.2. Point clouds were generated instead of lines, as this facilitated the conversion of 3D

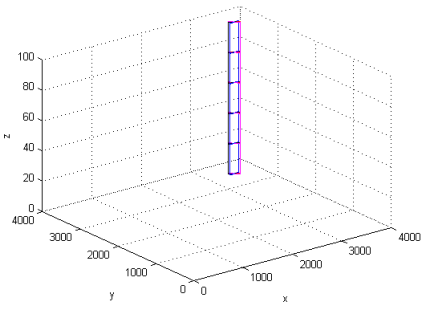


Figure 4.4.1: 3D point cloud of a pole as seen from all cameras

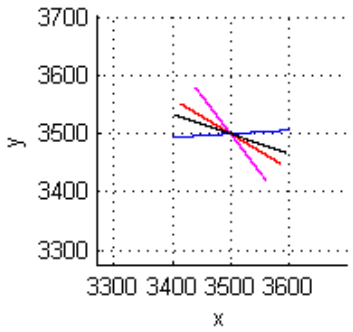
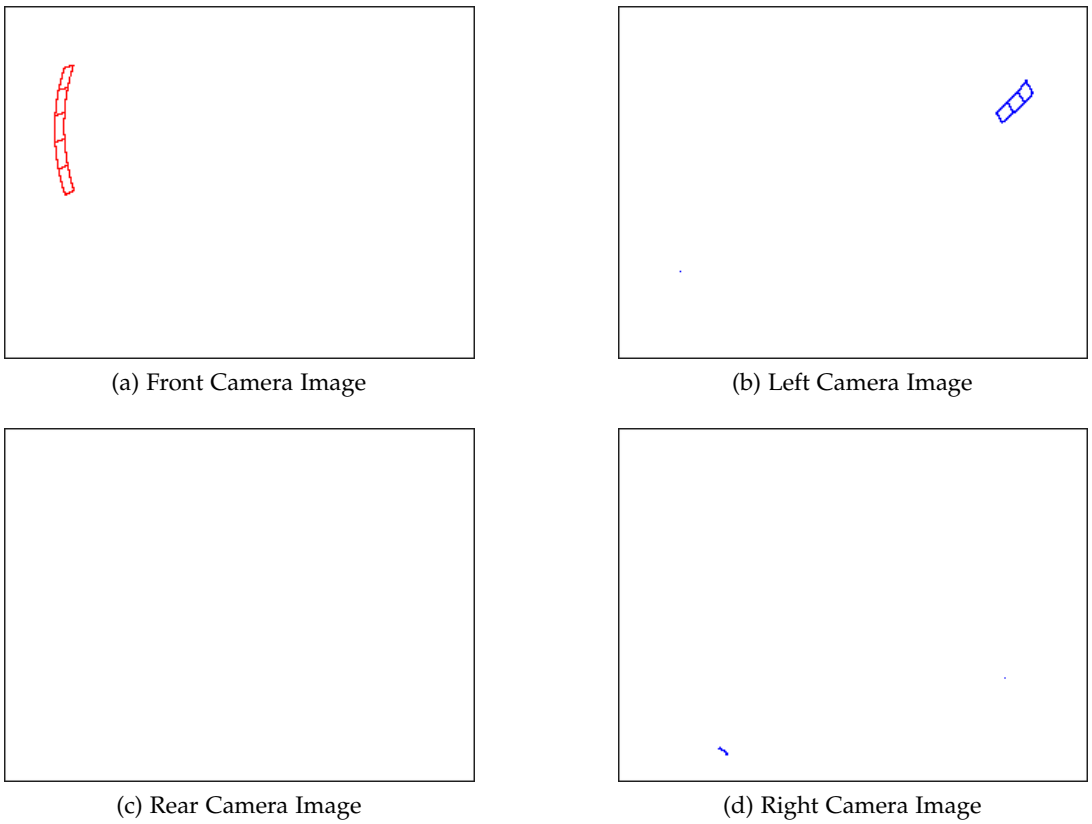


Figure 4.4.2: Top view of the point clouds generated, red for front camera, blue for left camera, magenta for rear camera and black for right camera

World to 2D Images. It is easier to convert points from the world to the image using the camera model (the intrinsics matrix, the extrinsics matrix and the distortion functions) than lines, as lines will have to be finally converted to points, or only end points of line would be available and the region in between would need to be interpolated. These point clouds along with the camera model were used to create the images.

In the world coordinate system, the origin is the center of the rear axle on the car. The positive x-axis is directed towards the front and the positive y-axis is directed towards the left of the car. For the example shown in Figure 4.4.1, the point cloud is visible in the left and front cameras (since both x- and y-coordinates are positive) and is not visible in the right and rear cameras (as the point cloud is behind the cameras, as dictated by the viewing direction of the cameras). The images generated using this method also proves this fact, and are as shown in Figures 4.4.3 (a), (b), (c) and (d). The



(a) Front Camera Image

(b) Left Camera Image

(c) Rear Camera Image

(d) Right Camera Image

Figure 4.4.3: Images generated from the synthetic point clouds using the camera model (intrinsic parameters, extrinsic parameters and distortion function)

image of the right camera as shown in Figure 4.4.3 (d) also shows the pole, however the viewing angle is limited using the alpha blending explained earlier. Hence, all points in the world can be utmost viewed by a maximum of two cameras.

Two synthetic sequences, Synthetic sequence 1 and Synthetic sequence 2 were generated using this method. One simulated the car driving by the pole on the left side and another on the right side. Correspondences were selected on these images using a tool, as discussed in Section 1. The calculation of world coordinates was carried out using the algorithm described in Section 3.1.1. This was compared with the actual coordinates of the 3D point cloud. The Table 4.4.1 is one example of the outcome of the algorithm on point correspondences of one frame of the synthetic images. The Table 4.4.2 is the ground truth, where the actual coordinates of points from the point cloud were used and the midpoint rule of skew lines (see Section 3.1.1 for details) was used to calculate the world coordinates. Table 4.4.3 shows the % error of using the algorithm with reference to the point cloud.

X	Y	Z
4919.99	2622.86	16.2298
5046.21	2437.17	15.7548
4888.85	2595.84	414.52
5088.78	2482.86	427.472
4873.20	2583.72	783.325
5064.80	2462.95	827.607
4858.96	2570.50	1218.07
5029.87	2428.16	1198.88

Table 4.4.1: Result of Algorithm

X	Y	Z
5069.19	2421.81	0.57220
4925.70	2573.59	1.13250
5074.30	2426.41	399.268
4925.70	2573.59	401.533
5074.30	2426.41	799.668
4925.85	2573.44	801.931
5074.30	2426.41	1200.07
4926.00	2573.30	1202.33

Table 4.4.2: Ground Truth

X	Y	Z
−2.9432	8.3016	2736.31
2.4466	−5.3008	1291.15
−3.6547	6.9827	3.8200
3.3108	−3.5254	6.4600
−3.9631	6.4832	−2.0437
2.8208	−4.2935	3.2018
−4.2437	5.9384	1.5000
2.1086	−5.6402	−0.2869

Table 4.4.3: % Error

Similar tables were created for all the frames with correspondences. As can be seen, there is a huge % error in the first two rows for the Z coordinate. This is due to the points being on the ground surface having a small position error. But, owing to the fact that the ground truth is close to 0, for the percentage, it causes a division by a very small value, making the % error very high. The error statistics of X, Y and Z coordinates are as detailed in Table 4.4.4. 136 points were used in total and it was confirmed that the absolute error in the Z coordinates of ground point was much less than the other errors and these were deleted before the statistics were calculated. As can be seen from Table 4.4.4, the Z coordinate still had higher mean % error and much higher standard deviation. This is due to the use of the mid-point rule of skew

lines and that the point correspondences were always on the edge of the pole, which explains the higher error.

	X	Y	Z
Mean	0.008729	−0.35494	16.7811
Standard Deviation	1.8948	5.3499	10.9820

Table 4.4.4: % Error Statistics in calculation of World coordinates

The error in calculation of world Coordinates is mainly on account of low resolution of the images (320 x 240) and due to pixel accuracy of image correspondences (sub-pixel accuracy would improve the estimation). Hence, with an actual feature detection and matching algorithm (with sub-pixel accuracy) and with improved resolution of images, the estimation of world coordinates would improve.

4.5 RESULTS OF DEFORMATION METHOD WITHOUT USING GROUND PLANE

Figure 4.5.1 shows the comparison of the visualizations, as seen on the simulator. It compares the non-deformed view with the deformed view for the method discussed in Section 3.2.1. As can be seen from the figure, there is an improvement in the

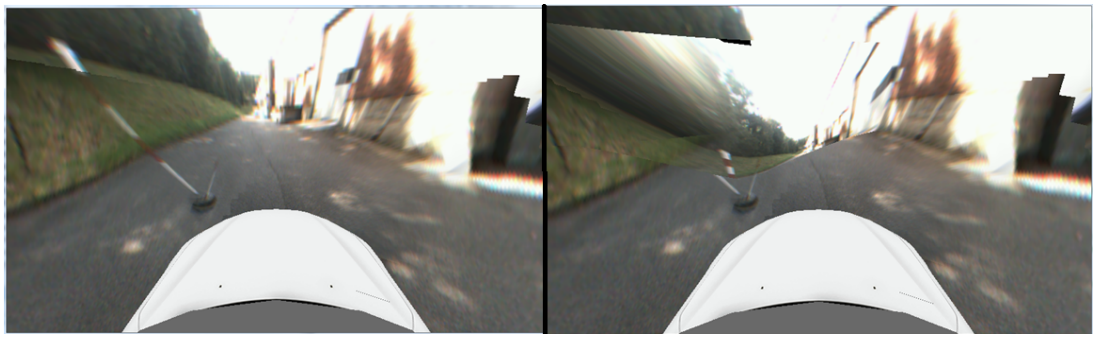


Figure 4.5.1: Deformation without using ground plane (The left image is the undeformed View and the right image is the deformed view) - Image provided by Robert Bosch Car Multimedia GmbH

visualization for the surface above the ground (the region of the pole). The ghost artefacts above the ground are eliminated. However, on the ground surface, the ghost artefacts are still visible. This is due to the fact that the ground plane was not

considered for the deformation. However, the improvement in visualization above the ground proves that this method if used along with ground plane can help eliminate the ghost artefacts completely.

4.6 RESULTS OF DEFORMATION METHOD INCLUDING THE GROUND PLANE

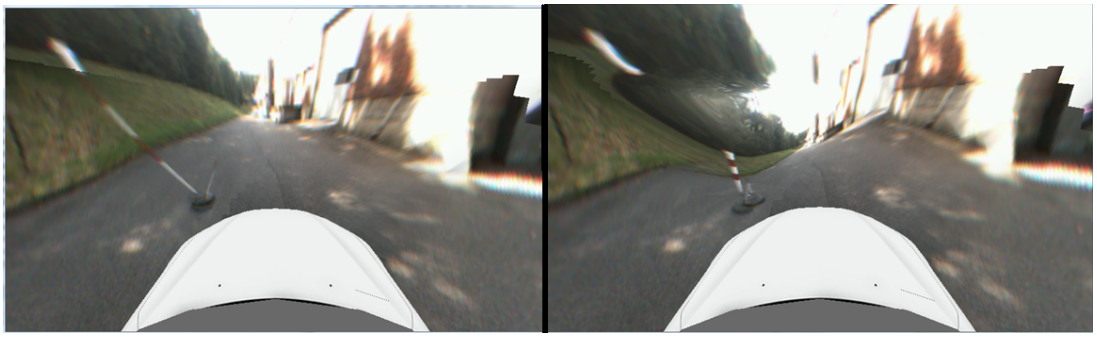


Figure 4.6.1: Deformation using ground plane (The left image is the undeformed view and the right image is the deformed view) - Front View - Image provided by Robert Bosch Car Multimedia GmbH

Figure 4.6.2: Deformation using ground plane (The left image is the undeformed View and the right image is the deformed view) - Left View - Image provided by Robert Bosch Car Multimedia GmbH

Figure 4.6.1 shows the comparison of the visualizations, as seen on the simulator. It compares the non-deformed view with the deformed view for the method as discussed in Section 3.2.2. As can be seen from the figure, there is a great improvement in visualization and the ghost artefacts are now parallel. This can be seen more clearly

in Figure 4.6.2 which shows the view from the left side. These parallel ghosts appear due to the small error in world coordinates (owing to the different perspectives of camera in addition to the reasons stated in Section 4.4). When the algorithm was run on multiple frames of different sequences, in some of the frames, black regions were observed, as shown in Figure 4.6.3. This is due to the fact that some part of the deformed surface is not within the visible frustum of cameras as discussed in Section 3.3.2.

As described in Section 3.3.2, a solution of using Constrained Deformation was arrived at. Figure 4.6.4 shows the implementation of the same example frame with black region shown in Figure 4.6.3, with the solution of Constrained Deformation implemented. As can be seen from the Figure 4.6.4, there is visible blurring or stretching of the image, where there were black regions. This is due to the fact that, the basic algorithm results in a C_1 continuous surface and resists the changes to fundamental forms (lengths of curves, areas etc.) [BS08]. This is lost by using the Constrained Deformation. Hence, the textures appear to be stretched.

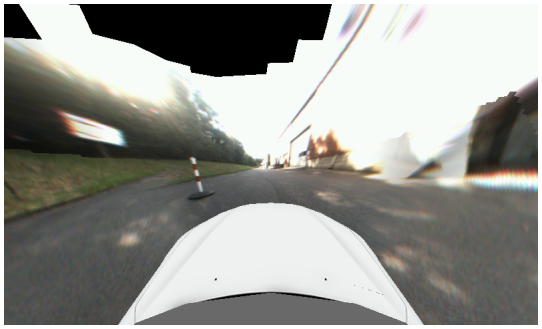


Figure 4.6.3: Black Regions during Deformation due to the mesh region going out of the visible frustum of cameras during deformation - Image provided by Robert Bosch Car Multimedia GmbH

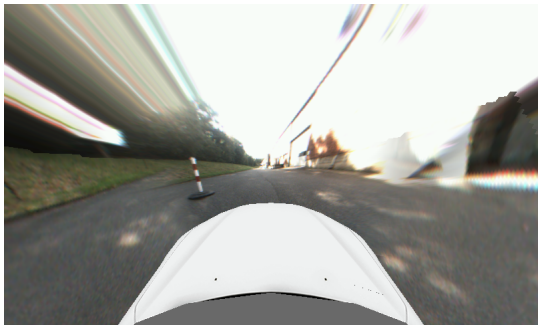


Figure 4.6.4: Solution of black regions in Deformation - Constrained Deformation - Image provided by Robert Bosch Car Multimedia GmbH

Figures 4.6.5, 4.6.6, 4.6.7 and 4.6.8 show frames of different sequences implemented using the method of Constrained Deformation described in 3.3.2, including the implementation on the synthetic image sequence. As can be seen, in Figures 4.6.5 and 4.6.6

Figure 4.6.5: Constrained Deformation including ground plane (obstacle in Back-Right Region) - Image provided by Robert Bosch Car Multimedia GmbH

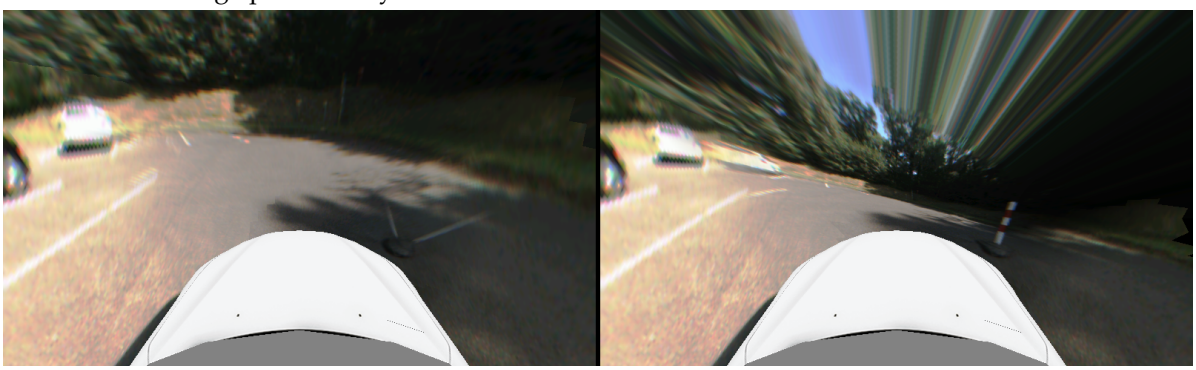


Figure 4.6.6: Constrained Deformation including ground plane (obstacle in Front-Right Region) - Image provided by Robert Bosch Car Multimedia GmbH

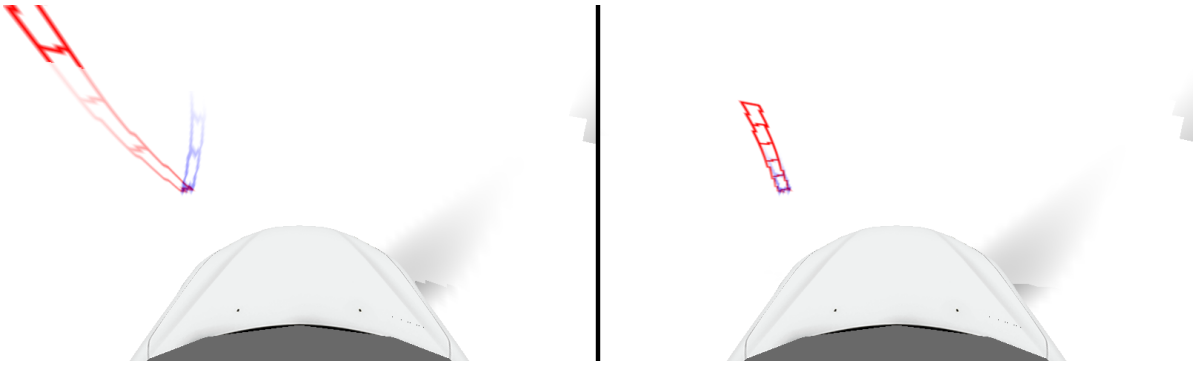


Figure 4.6.7: Constrained Deformation including ground plane (obstacle in Front-Left Region - Synthetic Images) - Image provided by Robert Bosch Car Multimedia GmbH

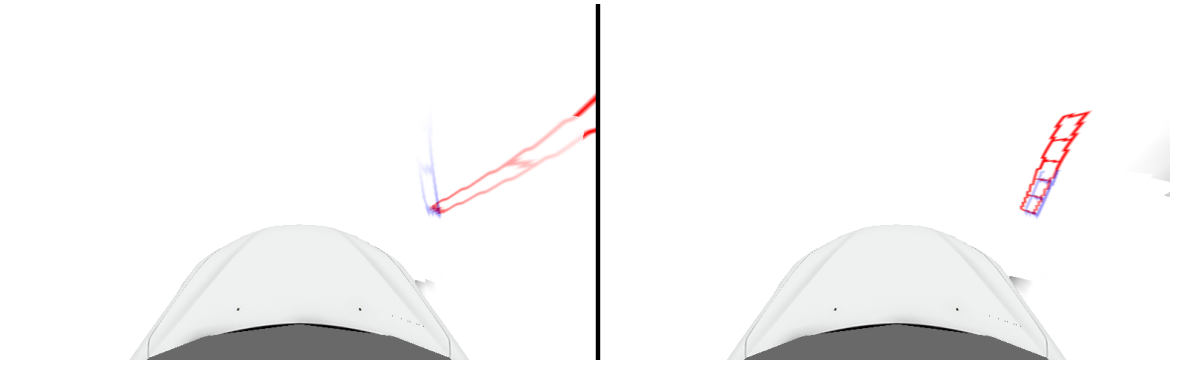


Figure 4.6.8: Constrained Deformation including ground plane (obstacle in Front-Right Region - Synthetic Images) - Image provided by Robert Bosch Car Multimedia GmbH

Figure 4.6.9: Free as handle Deformation including ground plane (obstacle in Back-Right Region) - Image provided by Robert Bosch Car Multimedia GmbH

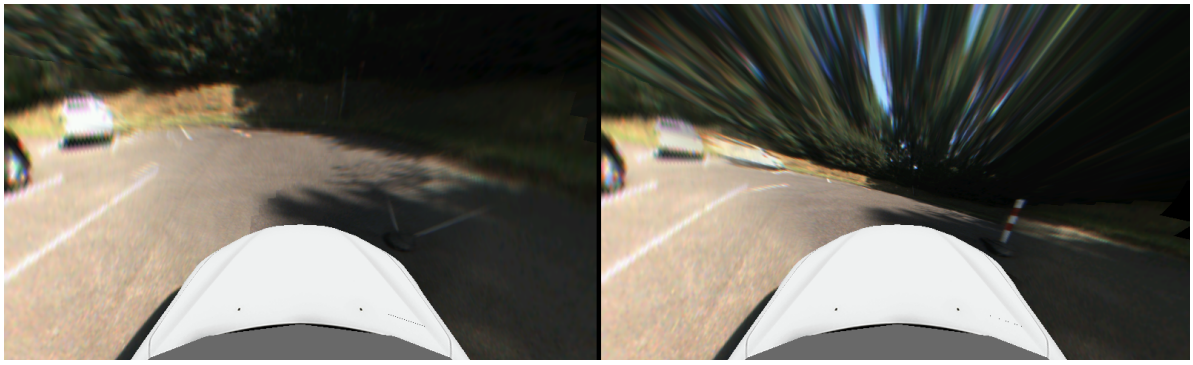


Figure 4.6.10: Free as handle Deformation including ground plane (obstacle in Front-Right Region) - Image provided by Robert Bosch Car Multimedia GmbH

the scene seems rotated a little. This is due to the fact that the deformation is across a very wide angle. Since the the deformation is intuitive, it tries to maintain a smooth surface and tends to be directed towards the handle. This effect gives a feeling that the scene is rotated. Also in Figure 4.6.6, three cars are seen on the left side instead of two cars, in the deformed view. This is the same problem dealt with in this thesis. Since the deformation is not done on this side, the car comes into the overlap region at a farther distance and the two perspectives of the different cameras are projected and blended, giving rise to three cars instead of two.

Using the method of constraining the free region to the handle before deformation as described in 3.3.2, the frames described in Figures 4.6.5 and 4.6.6 look as shown in Figures 4.6.9 and 4.6.10. The synthetic frames exhibited no difference owing to the fact that no textures other than the pole are present in the images. It can be seen that

the amount of stretching of textures and blurring is visibly same or increased. This is due to the fact that, although the solution tries to preserve the lengths and areas of surface, it cannot preserve it completely. Hence, the textures still appear to be stretched.

Both the challenges discussed above (scene rotation and appearance of three cars instead of two) can be eliminated when deformation is done at all the four overlap regions, depending on the image correspondences simultaneously. Additionally a least squares curve (preferably an ellipse) can be fit first with respect to the depth in four or eight directions while forming the initial bowl, so that the amount of deformation with respect to the frame can be minimized, resulting in fewer artefacts after deformation.

4.7 NOISE ON PIXEL COORDINATES

The algorithm was run with image correspondences selected manually using a tool. As explained in Section 3.1, this was done to understand the ground truth.

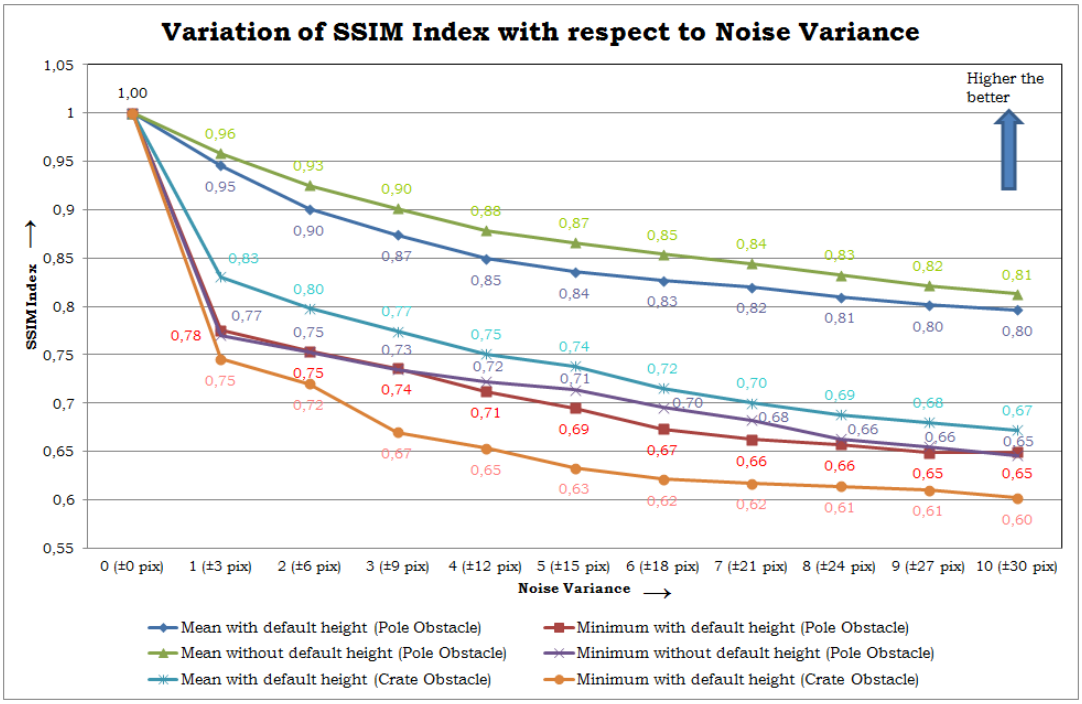


Figure 4.7.1: Variation of SSIM Index with respect to Noise Variance

In actual practice this would not be feasible, as the algorithm has to run in real time and an automatic feature detection and matching algorithm would be used. One of the problems associated with such an algorithm is that it is bound to have some error associated with it and is not perfect. To understand the behavior of the algorithm when the feature matching algorithm is implemented, an experiment was carried out.

In this experiment, random noise from a Gaussian distribution with 0 mean and varying variance was added to each of the pixel coordinates of the image correspondences in both the images and the algorithm was run on the 64 Frames of Real World Sequence 1 and Real World Sequence 2. To understand the effect of size of obstacle, 16 frames of the sequence, Real World Sequence 4 was used. The quality of the images so formed on the visualization of simulator needed to be studied. This was done qualitatively by getting the different sequences rated from different individuals and is described in Section 4.10. The method of Constrained Deformation as described in 3.3.2 is used for the experiment. The SSIM Index as described in Section 4.2 is used as a quality measure. Figure 4.7.1 shows the variation of the SSIM Index with respect to

Figure 4.7.2: Effect of Noise on Visualization - Left Image is a frame without noise, Right Image is same frame with a random noise of ± 10 variance with the lowest SSIM Index of 0.65 - Image provided by Robert Bosch Car Multimedia GmbH

change in the variance of the 0 mean Gaussian noise. The curves with default height were plotted when the algorithm was run using the default height of the handle region ($\mathtt{Height}_{\mathtt{handle}}$). The curves without default height were plotted with the algorithm run without any default height of the handle region. In this case, the height of the handle region was dictated by the maximum Z-coordinate of all the world coordinates obtained from the feature correspondences as described in Section 3.2.1. As can be seen from Figure 4.7.1, the mean and minimum SSIM Index reduces with increase

in the variance of the noise (which is expected). However for the Pole Obstacle, the decrease is not very sharp and is gradual with increase in noise variance. This is evident from the Figure 4.7.2 which shows a comparison of a frame without noise and the same frame with noise of 10 times the variance (equivalent of ±30 pixels), with the lowest SSIM Index of all the frames. As can be seen, the pole seems more bent at the bottom and then rises up. Also, there is not much difference in using the default height of the handle for improving the visualization.

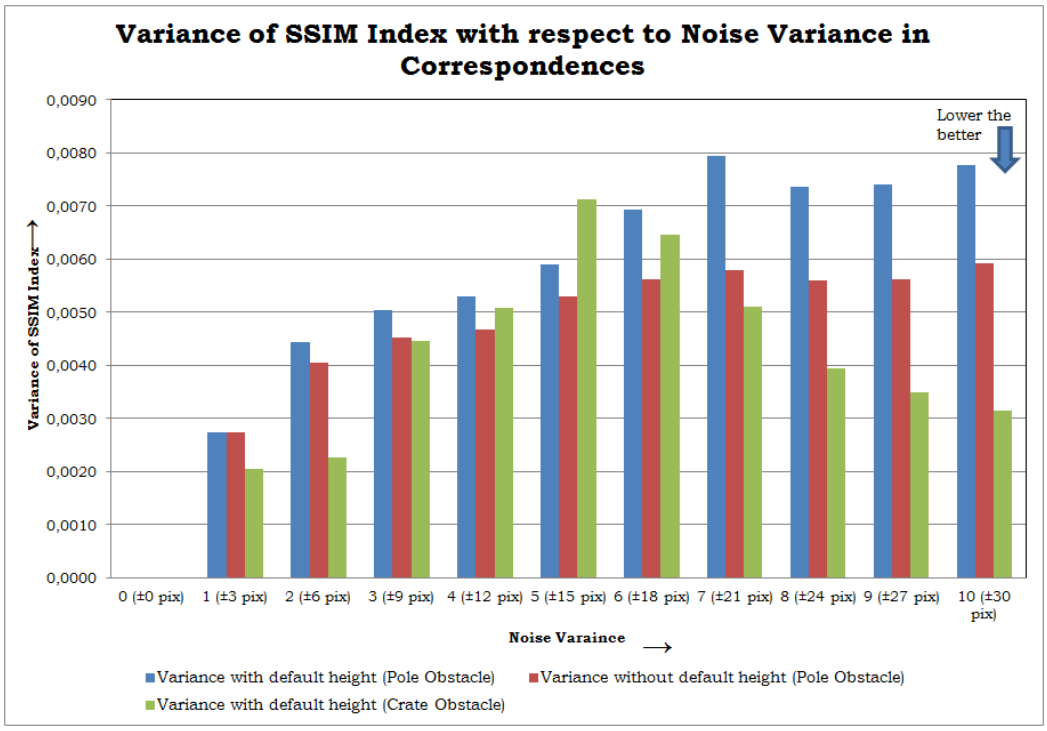


Figure 4.7.3: Variance of SSIM Index with respect to Noise Variance

But, the reduction in SSIM Index for a sequence with a bigger obstacle (a crate in this case) is sharper. This is because of the fact that the default angle of the handle is no longer in effect as the actual sector of handle (which is bigger than the default value) is used and varies with noise. Figure 4.7.3 shows the variance of SSIM Index with respect to change in noise variance. For the pole obstacle, the variance of SSIM Index shows an increasing trend with increase in variance of noise. Also, not using the default height, always has less variance with respect to the pole obstacle using the

default height. Also, the mean of the SSIM Index is always higher (as seen in Figure 4.7.1). This suggests that not using the default height should be beneficial. This is true only for individual frames. However, when run as a sequence, not using default height exhibits lot of jumping back and forth. This is supported by the fact that users rated not using default height lower than the algorithm using default height (see Section 4.10 for details). For the crate obstacle, the variance of SSIM Index initially increases and later on decreases, with increase in noise variance, and is almost always less than the pole obstacle using default height. However, the mean of SSIM Index for the crate obstacle is substantially lower than that for the pole obstacle using default height.

The Figures 4.7.1, 4.7.2 and 4.7.3 suggest that the algorithm is quite robust to noise for smaller obstacles. For larger obstacles, certain artefacts (similar to bending of crate) are introduced even at low noise levels. However, theoretically the behavior can be very different. This is because of the constrained deformation. The solution to the basic equation system (without constrained deformation) is always similar and may be a little shifted due to noise. The solution upto this part is robust to noise, in terms of the visualizations on simulator. However, owing to the constraints, the vertices may get laid on either plane of the frustum of the two cameras, due to which the visualisation may be greatly affected, resulting in extremely poor SSIM Index for the same noise level. This is especially true of the case where the obstacle is at a farther distance than the flat ground surface. However, such a phenomenon was not observed during the experiments.

4.8 MISMATCH OF IMAGE CORRESPONDENCES

The previous section discussed one of the problems of feature detection and matching algorithms. This section focusses on a second problem. Sometimes feature detection and matching algorithm detects and matches features that are not identical, that is, there are mismatches among detected features. To test on the robustness of our algorithm to feature mismatches, a second experiment was carried out.

In this experiment, the pixel values of one image were kept constant. The correspondences of this with the second image was changed in such a way that, one of the pixel values was removed at random from the list and another from the rest available would be repeated in its place. To understand the experiment, consider the following example. In Table 4.8.1, $x_{1,n}$, $y_{1,n}$ are the pixel coordinates of some world points in Image 1. $x_{2,n}$ and $y_{2,n}$ are the pixel coordinates of these same world points in

X_{Image1}	Y_{Image1}	X_{Image2}	Y_{Image2}
$x_{1,1}$	$y_{1,1}$	$x_{2,1}$	$y_{2,1}$
$x_{1,2}$	$y_{1,2}$	$x_{2,2}$	$y_{2,2}$
$x_{1,3}$	$y_{1,3}$	$x_{2,3}$	$y_{2,3}$
$x_{1,4}$	$y_{1,4}$	$x_{2,4}$	$y_{2,4}$
$x_{1,5}$	$y_{1,5}$	$x_{2,5}$	$y_{2,5}$

Table 4.8.1: Original Image Correspondences

X_{Image1}	Y_{Image1}	X_{Image2}	Y_{Image2}
$x_{1,1}$	$y_{1,1}$	$x_{2,1}$	$y_{2,1}$
$x_{1,2}$	$y_{1,2}$	$x_{2,2}$	$y_{2,2}$
$x_{1,3}$	$y_{1,3}$	$x_{2,1}$	$y_{2,1}$
$x_{1,4}$	$y_{1,4}$	$x_{2,4}$	$y_{2,4}$
$x_{1,5}$	$y_{1,5}$	$x_{2,5}$	$y_{2,5}$

Table 4.8.2: An example of Correspondences after mismatch

Image 2, thus making these rows, point correspondences between the two images. Assume there are five image correspondences to begin with. Now consider having one mismatch among these at random, say the third. Now, the third row pixel values of the right image are removed and one among the remaining four is selected at random in its place. The result is as shown in Table 4.8.2, where, the pixel coordinates of the first

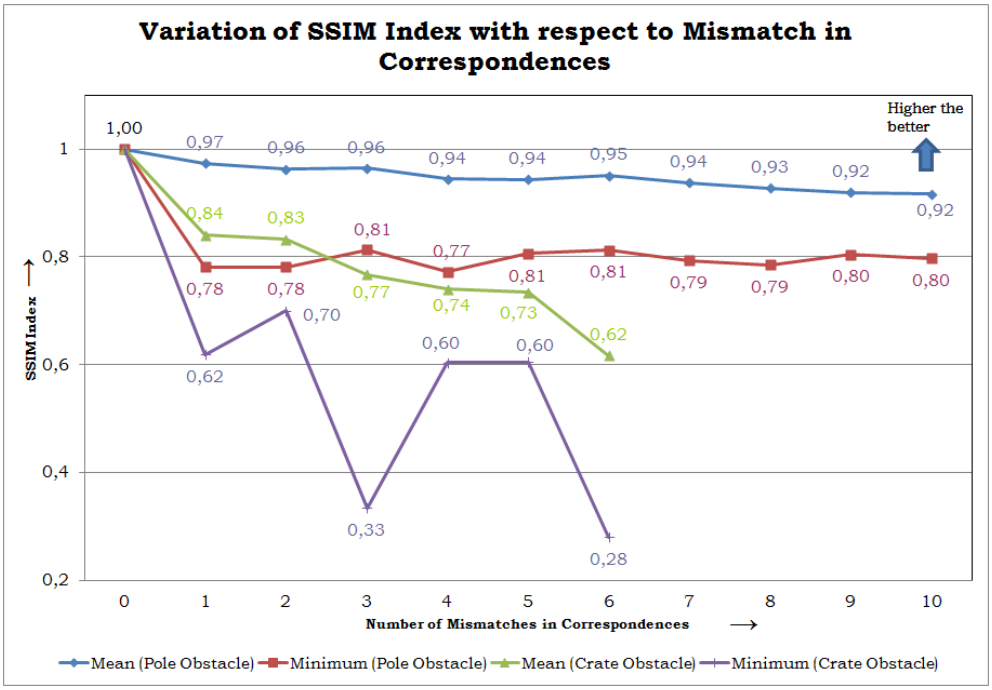


Figure 4.8.1: Variation of SSIM Index with respect to Number of Mismatches in Feature Correspondences

correspondence on the second image are also the coordinates of the corresponding pixel for the third correspondence. This process is repeated with more mismatches and the algorithm is run each time. The images of visualization are saved and are evaluated using SSIM Index, as described in Section 4.2, with respect to the visualizations with no mismatches, on the simulator. The experiment was carried out on a set of 31 frames of Real World Sequence 3 that have 10 image correspondences. To understand the effect of size of obstacle, 16 frames of Real World sequence 4 was also used. The method of Constrained Deformation as described in 3.3.2 is used for the experiment.

Figure 4.8.1 shows the variation of SSIM Index with respect to the number of mismatches between the feature correspondences. As can be seen, the SSIM Index drops a little with more mismatches for the Pole Obstacle. However, there is not significant difference between the visualization of Frame with no mismatches and the frame with all correspondences being mismatched. This is evident from the Figure 4.8.2 where a Frame with all its feature correspondences mismatched is compared with the same frame having no mismatches in correspondences for the Pole Obstacle. Such behavior can be reasoned due to the fact that all the correspondences are on the pole and the width of the pole is too small to have a negative impact on the visualization.

To further understand about this, Real World Sequence 4 with a huge obstacle (a Crate) was subjected to the same experiment. This is indicated by the curves Mean (Crate obstacle) and Minimum (Crate obstacle) on Figure 4.8.1. As can be seen from these curves, the SSIM Index drops sharply and there is a visible effect of mismatches

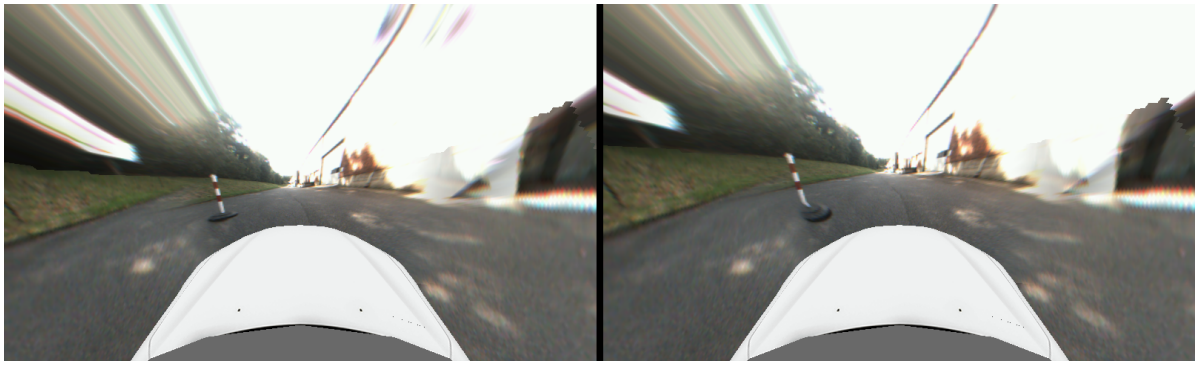


Figure 4.8.2: Effect of Mismatches on Visualization - Left Image is a Frame without mismatches, Right Image is the same Frame with all 10 correspondences mismatched randomly having lowest SSIM Index of 0.80 - Image provided by Robert Bosch Car Multimedia GmbH

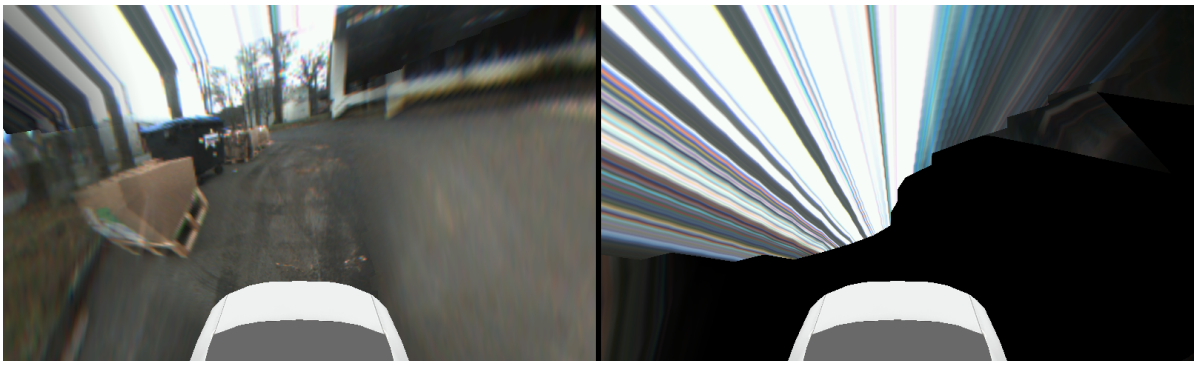


Figure 4.8.3: Effect of Mismatches on Visualization - Left Image is a Frame without mismatches, Right Image is the same Frame with all 6 correspondences mismatched randomly having lowest SSIM Index of 0.28 - Image provided by Robert Bosch Car Multimedia GmbH

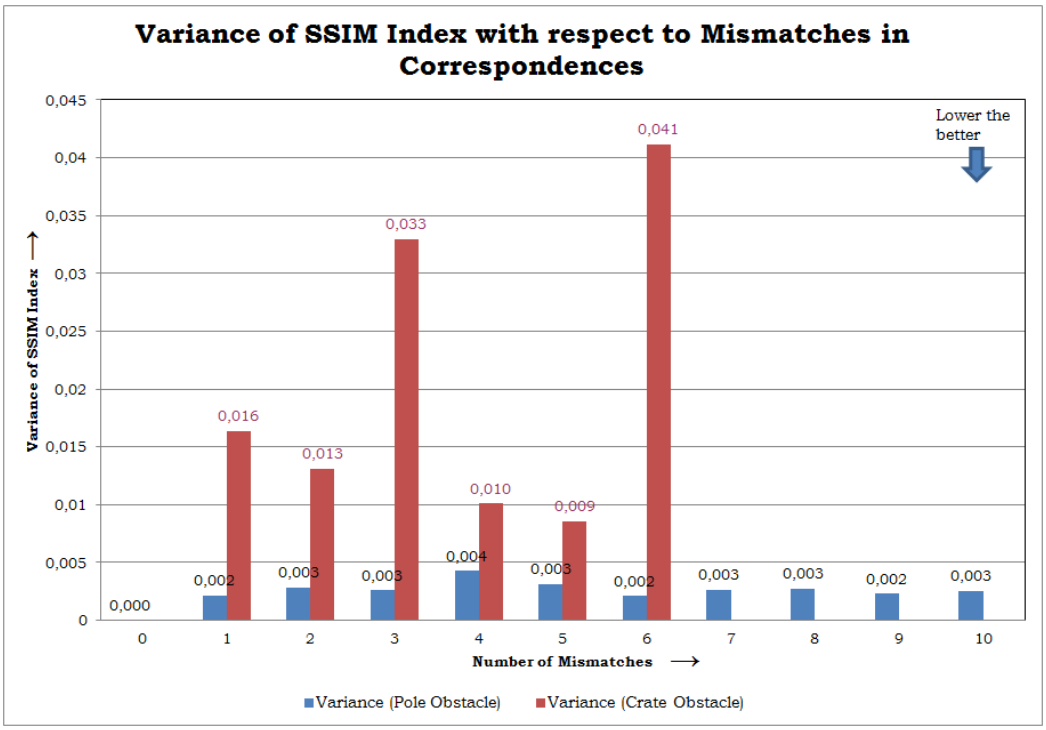


Figure 4.8.4: Variance of SSIM Index with respect to Number of Mismatches in Feature Correspondences

on the visualizations. This is also evident from Figure 4.8.3 where the frame without mismatches is compared with the same frame with all correspondences mismatched with the lowest SSIM Index of all frames. In this visualization, because of the mismatches, the location of the handle is inside the car. Due to this, the complete set of vertices forming the deformation goes outside the visible frustum and is behind the cameras, thereby rendering all those regions black. Also, there is no trend for the crate obstacle, in the minimum curve for SSIM Index. This is because of the random nature of generating mismatches.

This is also supported by the fact that the variance of SSIM Index for the crate obstacle is always substantially larger than that for the pole obstacle as evident from Figure 4.8.4. Hence, it can be concluded that the mismatches of feature correspondences, depends hugely on the size of the obstacle. Hence, it can be concluded that the algorithm is quite robust to small obstacles, but, fails with bigger obstacles.

4.9 NUMBER OF FEATURE CORRESPONDENCES

Another problem of feature detection and matching algorithm is detection of very few features in the images or no features in the worst case. An experiment was planned to understand the robustness of the algorithm to such an effect. The set of 31 Frames used in the previous experiment is used as base. To understand the effect of size of obstacle, the sequence of 16 frames of Real World sequence 4 was also used. Every time a certain amount (ranging from 1 to 10 for the 31 frames and from 1 to 6 for the 16 frames) of the pixel correspondences is removed at random and the algorithm is run, until there are no more correspondences available.

Each time the SSIM Index, as described in Section 4.2, is calculated between the deformed visualization on the simulator with 10 correspondences and with the remaining visualizations obtained by removing the correspondences. The method of Constrained Deformation as described in 3.3.2 is used for the experiment. The same is plotted as a measure of number of correspondences.

Figure 4.9.1 shows the effect of the number of correspondences on the SSIM Index. As can be seen, there is negligible impact of, the number of correspondences, on the SSIM Index for the Pole Obstacle. Only when all correspondences are lost (no features are detected), the SSIM Index falls sharply. This is due to the fact that the mesh takes the undeformed shape in the absence of correspondences and the

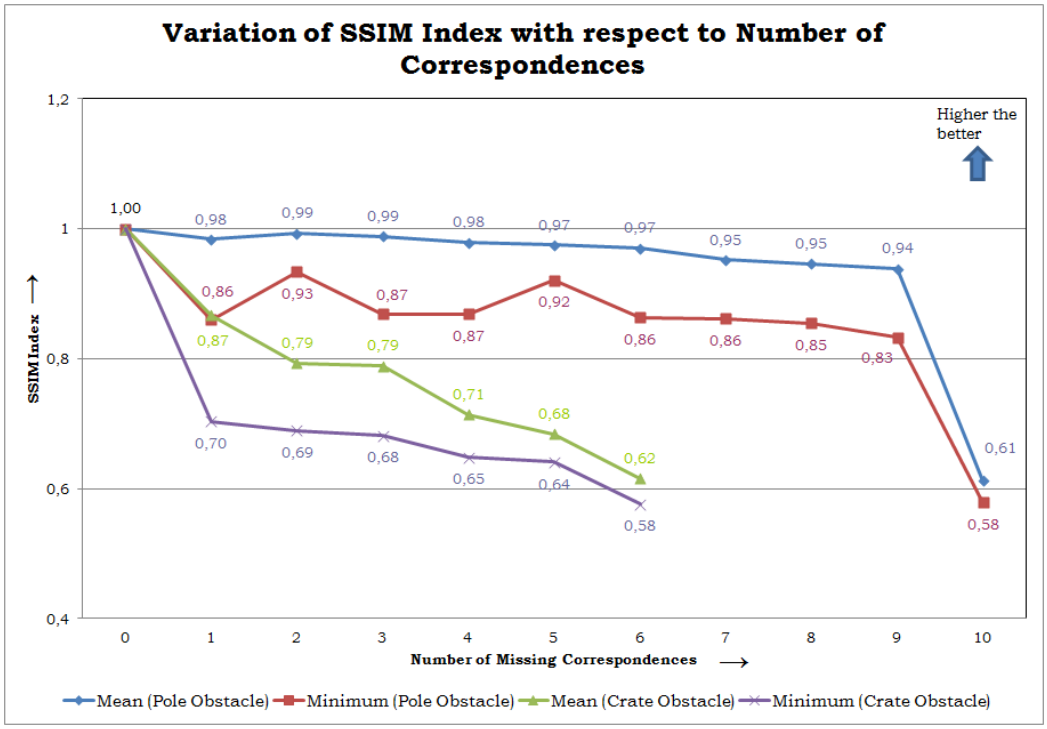


Figure 4.9.1: Variation of SSIM Index with respect to Number of Feature Correspondences

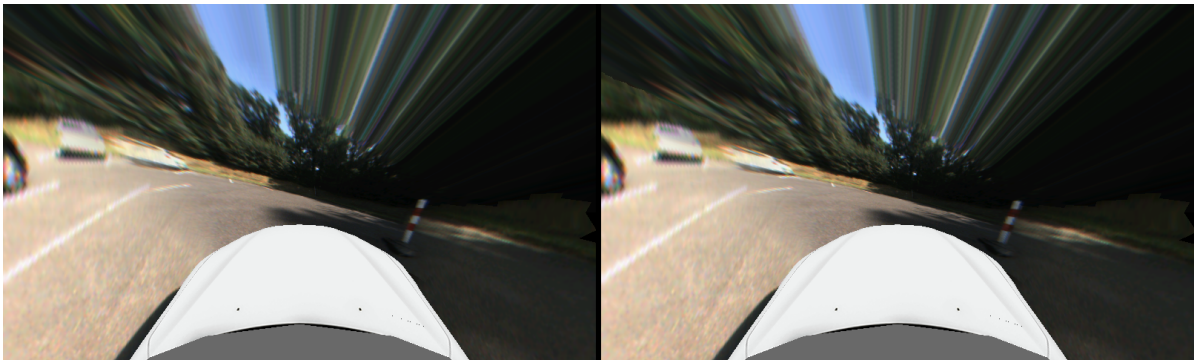


Figure 4.9.2: Effect of Number of Correspondences on Visualization - Left Image is a Frame with all 10 Correspondences, Right Image is the same Frame with only 1 correspondence at random having lowest SSIM Index of 0.83 for the Pole Obstacle - Image provided by Robert Bosch Car Multimedia GmbH

visualization is definitely a lot different from that of a deformed mesh. The negligible

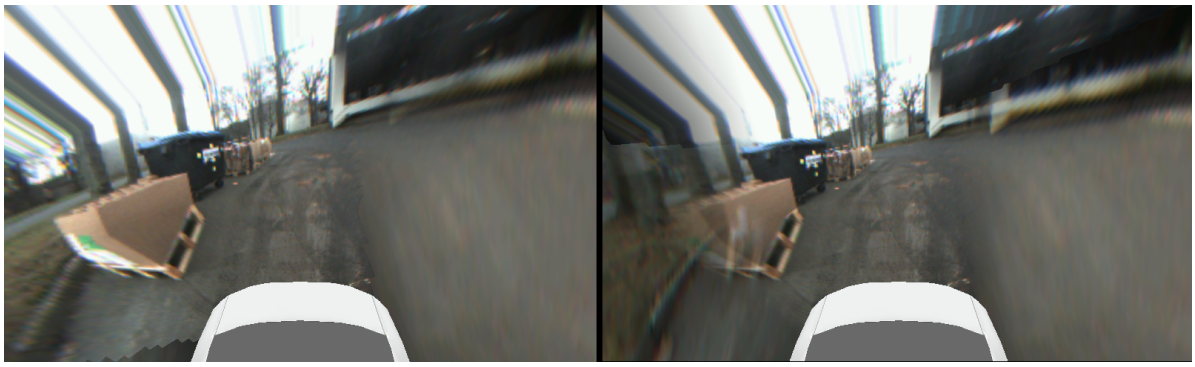


Figure 4.9.3: Effect of Number of Correspondences on Visualization - Left Image is a Frame with all 6 Correspondences, Right Image is the same Frame with only 1 correspondence at random having lowest SSIM Index of 0.64 for the Crate Obstacle - Image provided by Robert Bosch Car Multimedia GmbH

difference in visualizations, for a frame with one correspondence (with the lowest SSIM Index), is compared with the same frame having ten correspondences, and is as shown in Figure 4.9.2.

For a bigger obstacle (a Crate), the SSIM Index falls a little more sharply. This is due to the fact that in absence of some of the correspondences, the angle of the handle sector (which is greater than the default angle) is in effect and is varying. This is evident from the Figure 4.9.3. Figure 4.9.4 shows the variance of SSIM Index with respect to number of correspondences. The variance of SSIM Index is always substantially larger for the crate obstacle than the pole obstacle. Only for 6 missing correspondences, the variance of SSIM Index for crate obstacle is less than that of pole obstacle. It is however equal to the variance of SSIM Index for the pole obstacle with 10 missing correspondences. This is because, in both these cases (crate obstacle with 6 missing correspondences and pole obstacle with 10 missing correspondences), the bowl returns to undeformed state and is compared with that of the deformed visualization.

Using Figures 4.9.1, 4.9.2 and Figure 4.9.4, it can be stated that the algorithm is robust against the number of correspondences as long as there is at least one reliable correspondence detected, for the Pole Obstacle. For larger obstacles, however, the algorithm is not so robust against number of correspondences.

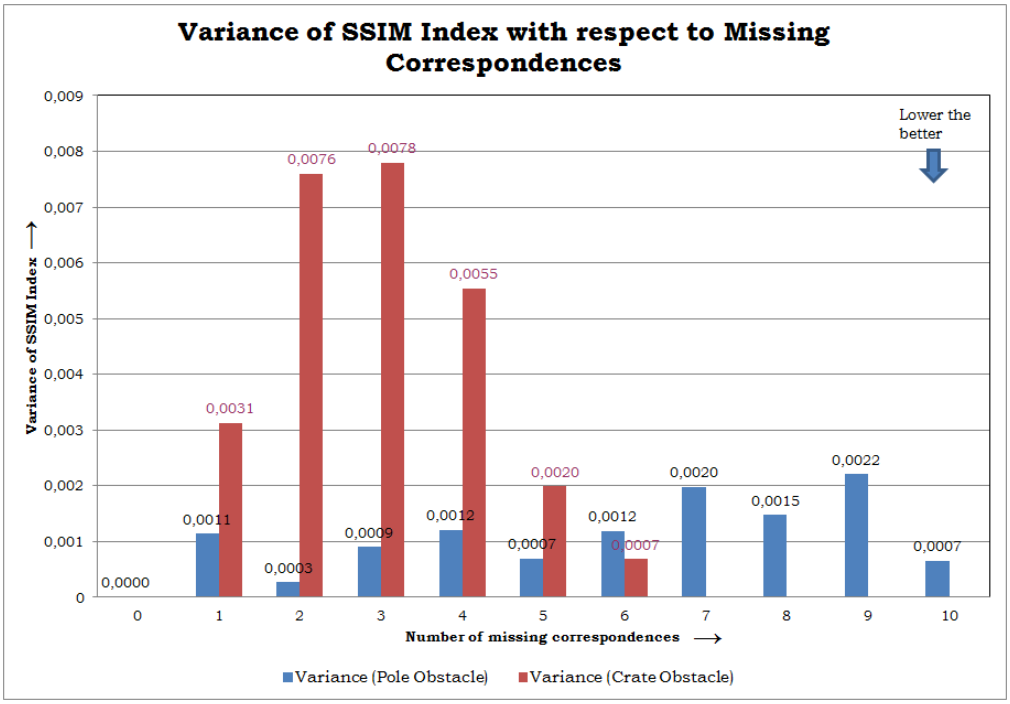


Figure 4.9.4: Variance of SSIM Index with respect to Number of Feature Correspondences

4.10 QUALITATIVE EVALUATION

To understand if the algorithm implemented makes a difference to the visualizations of the pole, there are no tools that can replace the human perception of vision. Hence, videos of three different sequences were made, where each video had three options for comparison and rating. The options were:

1. The deformation algorithm along with constrained deformation as described in Section 3.3.2.
2. Same as option 1 above, but without using the default height for the handle region.
3. Same as option 1 above, but with noise of 0 mean and 10 times the variance (equivalent of ± 30 pixels) added to the point correspondences at random. This option is provided to verify the agreement/disagreement with results of Quantitative analysis.

Also, the uninteresting portions of the image (areas other than the pole) were darkened to ensure focus on visualization of the pole alone and not of the surrounding regions. These were shown to a set of 10 individuals of which 8 were experts in Computer Vision/Computer Graphics. The users were to rate each of the three options relatively from a rating of 1 to 10. 1 was to mean a really poor visualization and 10 was to mean a very good visualization. The ratings were then normalized to have values between 0 and 1, 0 for poor and 1 for good visualization as per the user. The ratings received and the observations there off, is discussed in the following.

Average Ratings	Deformation	Deformation without Default Height	Deformation with Noise
Sequence 1	**0.79**	0.52	0.34
Sequence 2	**0.79**	0.68	0.27
Sequence 3	**0.63**	0.41	0.24

Table 4.10.1: User Evaluation - Ratings received are normalized to be between 0 and 1, 0 for poor visualization and 1 for good visualization

The average ratings for each of the three sequences for the various options are as summarized in Table 4.10.1. As can be seen from the Table 4.10.1, the option with the deformation using default height for the handle region has the highest rating. Next is the deformation without using the default height for the handle region and the option of deformation with noise was rated as worst. This does not agree completely with the quantitative results as seen in Section 4.7 above. Although, in quantitative analysis, the SSIM Index with default height had always a lower score than the one without default height, users thought otherwise. This is because of the fact that in video sequences, the history of previous frames matters and a sequence with frequent jumps between frames is considered disturbing and is rated lower.

4.11 TIMING OF THE ALGORITHM

All experiments were done on the Simulator under Microsoft Windows® platform and MATLAB® (The MathWorks Inc.) was used for solution to the sparse linear system. For checking if the algorithm can be run on the target system, a skeleton form of the algorithm was developed with the mesh and deformation functions alone and without any images, to keep it simple. The code was tested on a Linux machine (in

VirtualBox) on a desktop PC for confirming compatibility on the target. The target system is a i.MX 6 DualLite Processor-Dual Core, 3D Graphics, HD Video, Multimedia, ARM® Cortex®-A9 Core. The CPU can utmost run at 1GHz. The OS running on the target system is a UBUNTU based Linux modified by Bosch for internal purposes. For making the algorithm independent of MATLAB®, three different methods were used for solving the sparse linear system of equations, namely, SparseLU [Li05], Multifrontal LU[DD97], [Dav04b] and BiCGSTAB[Vor92].

SparseLU is Sparse supernodal LU factorization into a lower triangular matrix L and an upper triangular matrix U with partial pivoting, along with a permutation matrix P which is pre-multiplied to the product of L and U. The resulting triangular system after factorization is solved through forward and back substitution. This can then be used along with an appropriate ordering to solve the linear system. The column or row pre-ordering helps reducing the fill-in. "Fill-in is the introduction of new non-zero enteries in L and U whose corresponding entries in A are zero" [Dav04b]. (Sparse LU solver with COLAMD which implements Supernodal LU, of Eigen 3.2.2, was used [GJ+10]). COLAMD stands for COLumn Approximate Minimum Degree pre-ordering algorithm. It is one of the methods of column pre-ordering used commonly to reduce fill-in.

In case of Multifrontal LU, the algorithm uses a preordering [Dav04a] using a modified version of COLAMD that reduces the fill-in using partial pivoting, and then uses the multifrontal method [Dav04a]. The ordering method has different strategies depending on the pattern of the system matrix so as to give the minimal fill-in. The frontal method is a variation of the Guass elimination that avoids a large number of operations on zero terms [KNB87]. The method involves working only on a small subset of the equations. This small set is called the front and the operations on the front are done using dense matrix operations that are very efficient on the CPU. During a sparse matrix operation, the fronts are only stored in memory and operated on. The multifrontal method exploits this, by allowing multiple fronts to occur at a time [DR83] and hence, arriving at the solution of the system faster.

Lastly, BiCGSTAB uses the iterative variant of Bi-Conjugate Gradient (BiCG) method and has faster and smoother convergence than the orignial BiCG[Vor92]. In this method, pre-conditioners are used to help converge faster. (Diagonal pre conditioner of Eigen 3.2.2, was used [GJ+10]). All the methods can be additionally run on GPU. However, this suppresses the shaders, which means, the display becomes inactive when GPU is used for mathematical processing. Since, our application has a user

interface that needs updating in real time, this option was not explored. All the four regions of overlap were tried and successfully implemented. The time taken by the algorithms and further discussions on compatibility are presented in the following.

Table 4.11.1 shows the time taken by the various methods for solving the sparse linear system as described in Section 4.11. These timings are an average of 10 trials. These were run on a powerful 2.67 GHz processor. The target is a low end ARM processor with a maximum frequency of 1 GHz.

Method	Sparse LU		BiCGSTAB		Multifrontal LU	
	Mean	Variance	Mean	Variance	Mean	Variance
Handle 1 Deformation	41.9164	0.4513	141.532	1.8534	**27.7872**	0.3388
Handle 1 Back to undeformed Bowl	0.0065	$1.48e-07$	0.0066	$4.66e-07$	0.0064	$1.56e-07$
Handle 2 Deformation	42.7429	0.3566	139.988	1.4996	**28.1425**	0.2744
Handle 2 Back to undeformed Bowl	0.0066	$1.61e-07$	0.0069	$1.58e-07$	0.0067	$4.82e-07$
Handle 3 Deformation	47.1704	0.6299	165.297	0.3742	**31.5149**	0.0826
Handle 3 Back to undeformed Bowl	0.0063	$1.04e-07$	0.0065	$6.97e-08$	0.0066	$2.54e-07$
Handle 4 Deformation	47.0052	0.3802	160.756	3.8940	**30.7542**	0.0198
Handle 4 Back to undeformed Bowl	0.0064	$1.56e-07$	0.0064	$5.79e-08$	0.0067	$1.45e-07$

Table 4.11.1: Time taken by different methods (in seconds)

The four regions of overlap are named Handle1 for Front-Left region, Handle2 for Back-Left region, Handle3 for Back-Right region and Handle4 for Front-Right region. From Table 4.11.1, it can be seen that Multifrontal LU methods takes the least time

and the iterative BiCGSTAB takes the highest. Incidentally, Multifrontal LU is also used in MATLAB®. However, the algorithm takes longer than MATLAB® to solve the system. Also, since the variance of all the methods is extremely low, it shows they are all consistent in performance. It is also clear that, all these methods in this form cannot be directly implemented on the target.

As an alternative, a 3D surface can be worked out based on the angles of free, fixed and handle regions. A lookup table can be made to approximate this continuous 3D surface so as to maintain the properties of solution to the sparse linear system. Different lookup tables can be stored for different depths depending on the amount of memory and the values in between could be interpolated using a certain scheme. This can be executed during startup and during runtime, only positions of vertices are calculated using values picked from the lookup table. This method is less costly than solving the sparse linear system. The time taken would be a little more than the time taken for removing the deformation and forming the original bowl in the Table 4.11.1 above, which would then make it compatible for real-time execution.

5 CONCLUSION AND FUTURE WORK

The thesis adapted an unconventional method of deforming meshes, for solving the problem of ghost artefacts. The idea used was, to deform the projection surface resulting in visible reduction of ghost artefacts. It was an attempt to understand the ground truth using this approach. For thin obstacles for example, the pole, the proposed algorithm is found to be quite robust against noise in feature correspondences, mismatches among correspondences and scarce number of correspondences, and produces good visualizations of the obstacle in the overlap region. For bigger obstacles, the algorithm is not as robust, as explained in Section 4, however yields good results with reliable feature correspondences on the obstacle. Also in the current form, it is not real-time capable. Therefore, the algorithm in this form cannot be implemented on the low end target system.

This thesis shows there is tremendous potential in the idea of using deformation of projection surfaces to eliminate ghost artefacts, and more research needs to be carried out in this direction, in order to come up with a solution that works well with all obstacles, with all the limitations of feature matching algorithms used and also works on the low end target system. It throws up a lot of research questions that needs to be answered before one has a good, real-time visualization not depending on the obstacle size or on reliability of feature detection and matching algorithms.

With respect to cameras, improved resolution with no compression should be used, so as to have advantage of better resolution. Compression may be used in case less bandwidth is available. This would improve visualization and also the accuracy of world coordinates calculation for deformation. Also, the visible frustum of cameras needs to be increased to aid in better visualization. This can be done either by choosing cameras with more than 180^{o} field of view in both directions, or by rotating the left and right cameras to look more upwards (similar to the front and rear cameras see Figure 3.3.1 for details), so that more volume is available and the visualization would be better after deformation. Ideal condition would be one where only pure deformation is required and the visible frustum does not constrain the deformed portion of the mesh.

For restricting the amount of deformation, one could use temporal correspondences (correspondences between images of the same camera in successive time steps) to have a sense of depth in different directions and fit a spline. This would then reduce the amount of deformation for individual obstacles and improve the visualization. Also, the depth could be discretized in steps and various lookup tables could be used to interpolate amongst these depths. Only the system memory would be a limiting factor on the number of lookup tables. With these factors, the algorithm would also be real time capable. The bias to the orientation of big obstacles needs to be studied. This is because, on big obstacles the depth varies on different parts of obstacle and may pose a problem to the visualization after deformation.

The feature matching algorithms are used to match correspondences between images, ultimately resulting into depth of the various points on the scene. To counter the unreliability of low end feature detection and matching algorithms that must run on extremely less processing power, one could use a different approach to do the same. For example, the LSDSLAM algorithm of [ESC14] could be used to directly find the depth of various points in the scene. It does not use any feature correspondence algorithm and relies on using image information alone to find depth of the scene and is known to run real time. Each pixel in the image is assigned a depth value. The information on the new image along with the history of depth maps for each pixel in the image is used to arrive at the depth map of current image. The depth data from the algorithm can be directly used to deform the bowl.

All the questions and possibilities stated above are taken up for future research and development, so that it finally results in a solution close to the ideal scenario in the real world.

BIBLIOGRAPHY

[Adm08] Administration, National Highway Traffic S.: *Fatalities and injuries in motor vehicle backing crashes*. http://www-nrd.nhtsa.dot.gov/Pubs/811144.pdf. Version: 2008

[Ale03] Alexa, Marc: *Differential coordinates for local mesh morphing and deformation*. Springer, 2003. http://dx.doi.org/14.02.2003. http://dx.doi.org/14.02.2003

[Ben75] Bentley, Jon L.: Multidimensional Binary Search Trees Used for Associative Searching. In: *Commun. ACM* 18 (1975), September, Nr. 9, 509–517. http://dx.doi.org/10.1145/361002.361007. – DOI 10.1145/361002.361007. – ISSN 0001–0782

[BK04] Botsch, Mario ; Kobbelt, Leif: An intuitive framework for real-time freeform modeling. In: (MERL), Joe Marks Mitsubishi Electric Research L. (Hrsg.): *ACM SIGGRAPH 2004 Papers*, ACM, August 2004, S. 630–634

[BL07] Brown, Matthew ; Lowe, David G.: Automatic panoramic image stitching using invariant features. In: *International Journal of Computer Vision* Bd. 74, 2007, S. 59–73

[BS08] Botsch, Mario ; Sorkine, Olga: On linear variational surface deformation methods. In: *IEEE Transactions on Visualization and Computer Graphics* Bd. 14, IEEE, February 2008, S. 213–230

[BSPG06] Botsch, Mario ; Sumner, Robert W. ; Pauly, Mark ; Gross, Markus: Deformation transfer for detail-preserving surface editing. In: *Vision, Modelling & Visualization*, IOS Press, AKA, November 2006, S. 357–364

[CE07] Chan, L. H. ; Efros, A. A.: Automatic generation of an infinite panorama. 2007. – Forschungsbericht

[CG91] Celniker, George ; Gossard, Dave: Deformable curve and surface finite-elements for free-form-shape design. In: *SIGGRAPH '91 Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, ACM New York, NY, USA, July 1991, S. 257–266

[Dav04a] Davis, Timothy A.: Algorithm 832: UMFPACK V4.3—an Unsymmetric-pattern Multifrontal Method. In: *ACM Trans. Math. Softw.* 30 (2004), Juni, Nr. 2, 196–199. http://dx.doi.org/10.1145/992200.992206. – DOI 10.1145/992200.992206. – ISSN 0098–3500

[Dav04b] Davis, Timothy A.: A Column Pre-ordering Strategy for the Unsymmetric-pattern Multifrontal Method. In: *ACM Trans. Math. Softw.* 30 (2004), Juni, Nr. 2, 165–195. http://dx.doi.org/10.1145/992200.992205. – DOI 10.1145/992200.992205. – ISSN 0098–3500

[DC04] Dornaika, F. ; Chung, R.: Mosaicking iimage with parallax. In: *Signal Processing: Image Communication* Bd. 19, 2004, S. 771–786

[DD97] Davis, Timothy A. ; Duff, Iain S.: An unsymmetric-pattern multifrontal method for sparse LU factorization. In: *SIAM Journal on Matrix Analysis and Applications* 18 (1997), Nr. 1. http://dx.doi.org/10.1137/S0895479894246905. – DOI 10.1137/S0895479894246905

[DMSB99] Desbrun, Mathieu ; Meyer, Mark ; Schröder, Peter ; Barr, Alan H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: *Computer Graphics (SIGGRAPH '99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques)*, ACM Press/Addison-Wesley Publishing Co. New York, NY, USA Copyright 1999, 1999. – ISBN 0–201–48560–5, S. 317–324

[DR83] Duff, I. S. ; Reid, J. K.: The Multifrontal Solution of Indefinite Sparse Symmetric Linear. In: *ACM Trans. Math. Softw.* 9 (1983), September, Nr. 3, 302–325. http://dx.doi.org/10.1145/356044.356047. – DOI 10.1145/356044.356047. – ISSN 0098–3500

[Ede01] Edelsbrunner, Herbert: *Geometry and Topology for Mesh Generation*. Cambridge University Press, 2001 http://dx.doi.org/10.1017/CBO9780511530067. – ISBN 9780511530067. – Cambridge Books Online

[ESC14] Engel, J. ; Schöps, T. ; Cremers, D.: LSD-SLAM: Large-Scale Direct Monocular SLAM. In: *European Conference on Computer Vision (ECCV)*, 2014

[FB81] Fischler, Martin A. ; Bolles, Robert C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In: *Commun. ACM* 24 (1981), Juni,

Nr. 6, 381–395. http://dx.doi.org/10.1145/358669.358692. – DOI 10.1145/358669.358692. – ISSN 0001–0782

[GJ+10] Guennebaud, Gaël ; Jacob, Benoît u. a.: *Eigen v3*. http://eigen.tuxfamily.org, 2010

[GKB11] Gao, Junhong ; Kim, Seon J. ; Brown, Michael S.: Constructing image panoramas using dual-Homography warping. In: *Computer Vision and Pattern Recognition*, IEEE, June 2011, S. 49–56

[Har03] Hartley, Richard: *Multiple view geometry in computer vision*. Cambridge, UK New York : Cambridge University Press, 2003. – ISBN 9780521540513

[JS96] Jr., J. E. D. ; Schnabel, Robert B.: *Numerical Methods for Unconstrained Optimisation and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1996

[KCVS98] Kobbelt, Leif ; Campagna, Swen ; Vorsatz, Jens ; Seidel, Hans-Peter: Interactive Multi-resolution Modeling on Arbitrary Meshes. In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA : ACM, 1998 (SIGGRAPH '98). – ISBN 0–89791–999–8, 105–114

[KNB87] Kardestuncer, H. ; Norrie, D.H. ; Brezzi, F.: *Finite element handbook*. McGraw-Hill, 1987 (McGraw-Hill reference books of interest: Handbooks). http://books.google.de/books?id=fLoRAQAAMAAJ. – ISBN 9780070333055

[Li05] Li, Xiaoye S.: An Overview of SuperLU: Algorithms, Implementation, and User Interface. 31 (2005), September, Nr. 3, S. 302–325

[LLM+11] Lin, Wen-Yan ; Liu, Siying ; Matsushita, Yasuyuki ; Ng, Tian-Tsong ; Cheong, Loong-Fah: Smoothly varying affine stitching. In: *Computer Vision and Pattern Recognition*, IEEE, June 2011, S. 342–352

[Llo06] Lloyd, S.: Least Squares Quantization in PCM. In: *IEEE Trans. Inf. Theor.* 28 (2006), September, Nr. 2, 129–137. http://dx.doi.org/10.1109/TIT.1982.1056489. – DOI 10.1109/TIT.1982.1056489. – ISSN 0018–9448

[Low99] Lowe, David: Object recognition from local scale-invariant features. In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference* Bd. 2, IEEE, 20 Sep 1999-27 Sep 1999 1999, S. 1150 – 1157 vol.2

[LSCO+04] Lipman, Yaron ; Sorkine, Olga ; Cohen-Or, Daniel ; Levin, David ; Rössl, Christian ; Seidel, Hans-Peter: Differential Coordinates for Interactive Mesh Editing. In: *SMI '04 Proceedings of the Shape Modeling International 2004*, IEEE Computer Society Washington, DC, USA, June 2004, S. 181–190

[MDSA03] Meyer, Mark ; Desbrun, Mathieu ; Schröder, Peter ; AlanH.Barr: Discrete differential-geometry operators for triangulated 2-manifolds. Version: 2003. http://dx.doi.org/10.1007/978-3-662-05105-4_2. In: Hege, Hans-Christian (Hrsg.) ; Polthier, Konrad (Hrsg.): *Visualization and Mathematics III*. Springer Berlin Heidelberg, 2003 (Mathematics and Visualization). – DOI 10.1007/978–3–662–05105–4_2. – ISBN 978–3–642–05682–6, 35-57

[Mei06] Mei, Christopher: *Projection Model*. http://www.robots.ox.ac.uk/~cmei/Toolbox.html. Version: 2006

[PP93] Pinkall, Ulrich ; Polthier, Konrad: Computing discrete minimal surfaces and their conjugates. In: *Experiment. Math.* 2 (1993), Nr. 1, 15–36. http://projecteuclid.org/euclid.em/1062620735

[QC07] Qi, Zhi ; Cooperstock, Jeremy R.: Overcoming parallax and sampling density issues in image mosaicing of non-planar scenes. In: *BMVC*, 2007, S. –1–1

[RLE+05] Rankov, Vladan ; Locke, Rosalind J. ; Edens, Richard J. ; Barber, Paul R. ; Vojnovic, Borivoj: An algorithm for image stitching and blending. In: *Proceedings of the SPIE* Bd. 5701, 2005, S. 190–199

[SCOL+04] Sorkine, Olga ; Cohen-Or, Daniel ; Lipman, Yaron ; Alexa, Marc ; Christian Rössl, C. ; Seidel, Peter-Hans: Laplacian surface editing. In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. New York, NY, USA : ACM, 2004 (SGP '04). – ISBN 3–905673–13–4, 175–184

[SG09] Shreiner, Dave ; Group, The Khronos OpenGL ARB W.: *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1*. 7th. Addison-Wesley Professional, 2009. – ISBN 0321552628, 9780321552624

[SS07] Shum, Heung-Yeung ; Szeliski, Richard: Construction of panoramic image mosaics with global and local alignment. In: *Internation Journal of Computer Vision* 36 (2007), February, Nr. 2, S. 101–130

[Sze06] Szeliski, Richard: Image alignment and stitching: A tutorial / Microsoft Corporation. Version: December 2006. http://dx.doi.org/10.1561/0600000009. 2006 (MSR-TR-2004-92). – Forschungsbericht

[Sze10] Szeliski, Richard: *Computer Vision: Algorithms and Applications*. 1st. New York, NY, USA : Springer-Verlag New York, Inc., 2010. – ISBN 1848829345, 9781848829343

[SZGP05] Sumner, Robert W. ; Zwicker, Matthias ; Gotsman, Craig ; Popović, Jovan: Mesh-based inverse kinematics. In: *ACM SIGGRAPH 2005 Papers*. New York, NY, USA : ACM, 2005 (SIGGRAPH '05), 488–495

[Tra10] Transportation, US D.: *U.S. DOT Proposes Rear View Visibility Rule to Protect Kids and the Elderly*. http://www.nhtsa.gov/PR/NHTSA-17-10. Version: 2010

[Vor92] Vorst, H. A. d.: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. In: *SIAM Journal on Scientific and Statistical Computing* 13 (1992), Nr. 2. http://dx.doi.org/10.1137/0913035. – DOI 10.1137/0913035

[WBSS04] Wang, Zhou ; Bovik, Alan C. ; Sheikh, Hamid R. ; Simoncelli, Eero P.: Image quality assessment: from error visibility to structural similarity. In: *Image Processing, IEEE Transactions on (Volume:13 , Issue: 4)* Bd. 13, IEEE, 2004. – ISBN 1057–7149 (ISSN), S. 600–612

[ZCBS13] Zaragoza, Julio ; Chin, Tat-Jun ; Brown, Michael S. ; Suter, David: As-projective-as-possible image stitching with moving DLT. In: *Computer Vision and Pattern Recognition*, IEEE, June 2013, S. 2339–2346

[ZHS+05] Zhou, Kun ; Huang, Jin ; Snyder, John ; Liu, Xinguo ; Bao, Hujun ; Guo, Baining ; Shum, Heung-Yeung: Large mesh deformation using the volumetric graph laplacian. In: *SIGGRAPH '05 ACM SIGGRAPH 2005 Papers*, ACM New York, NY, USA, July 2005, S. 496–503

[ZLPW06] Zomet, Assaf ; Levin, Anat ; Peleg, Shmuel ; Weiss, Yair: Seamless image stitching by minimising false edges. In: *IEEE Transactions on Image Processing* 15 (2006), S. 969–977. http://dx.doi.org/10.1109/TIP.2005.863958. – DOI 10.1109/TIP.2005.863958

Eidesstattliche Versicherung

_____ _____
Name, Vorname Matrikel-Nr.

Ich versichere hiermit an Eides statt, dass ich die vorliegende Masterarbeit mit dem Titel

---

---

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

_____ _____
Ort, Datum Unterschrift

Belehrung:

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -)

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird gfls. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin“) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

_____ _____
Ort, Datum Unterschrift