# AUTOMATIC 2D TO STEREOSCOPIC VIDEO CONVERSION FOR 3DTV

Xichen Zhou

A thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

July 2017

# Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Xichen Zhou**

Entitled: **Automatic 2D to Stereoscopic Video Conversion for 3DTV**

and submitted in partial fulfillment of the requirements for the degree of

**Master of Computer Science**

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Chair

_____ Examiner

_____ Examiner

_____ Supervisor

Approved _____
Chair of Department or Graduate Program Director

_____ 20 _____ _____

Rama Bhat, Ph.D.,ing., FEIC, FCSME, FASME, Interim Dean
Faculty of Engineering and Computer Science

# Abstract

Automatic 2D to Stereoscopic Video Conversion for 3DTV

Xichen Zhou

In this thesis we address the problem of automatically converting a video filmed with a single camera to stereoscopic content tailored for viewing using 3D TVs. We present two techniques: (a) a non-parametric approach which does not require extensive training and produces good results for simple rigid scenes and, (b) a deep learning approach able to handle dynamic changes in the scene. The proposed solutions both include two stages: depth generation and rendering. For the first stage, for the non-parametric approach we utilize an energy-based optimization, and for the deep learning approach a multi-scale convolutional neural network to address the complex problem of depth estimation from a single image. Depth maps are generated based on the input RGB images. We reformulate and simplify the process of generating the virtual camera's depth map and present how this can be used to render an anaglyph image. Anaglyph stereo was used for demonstration only because of the easy and wide availability of red/cyan glasses however, this does not limit the applicability of the proposed technique to other stereo forms. Finally, we have extensively tested the proposed approaches and present the results.

# Acknowledgments

Firstly I would like to thanks for my supervisors' professional support during this project, I would not be able to finish it without it. Also to my family for their encouragement and love when I had difficulties. Finally, to all the members of theICT Lab and 3dGraphics Lab whom I worked with for the good memories.

# Contents

**List of Figures** vii

**List of Tables** ix

**1 Introduction** 1

**2 Literature Review** 3

2.1 3D video Production . . . 3

2.2 Stereo Rendering . . . 5

2.2.1 Stereo rendering by perspective transform . . . 6

2.2.2 Depth Based Rendering . . . 7

2.2.3 Cue fusion based rendering . . . 7

2.3 Depth Cues . . . 8

2.3.1 Depth from Parallax . . . 8

2.3.2 Depth from Linear Perspective . . . 9

2.3.3 Depth from Occlusion . . . 9

2.3.4 Other depth cues . . . 9

2.4 Depth Map Estimation . . . 9

2.4.1 Parametric based method . . . 10

2.4.2 Non parametric based method . . . 11

2.4.3 Model based approaches . . . 12

**3 Methodology** 14

3.1 Super-Matching Method . . . 14

3.1.1 Superpixel Representation . . . 15

3.1.2 Descriptors for Superpixels . . . 16

3.1.3 Superpixel Matching . . . 17

3.2 CNN Approach for Depth Prediction . . . 20

3.2.1 Scale 1 . . . 23

3.2.2 Scale 2 . . . 24

3.2.3 Scale 3 . . . 24

3.3 Virtual Camera Projection for rendering . . . 25

3.4 Stereo View Rendering . . . . . 28

**4 Implementation and Results** **31**

4.1 Implementation . . . . . 31

4.2 Results . . . . . 32

4.3 Evaluation . . . . . 33

**5 Conclusion** **35**

5.1 Future Works . . . . . 35

**Bibliography** **36**

# List of Figures


1 Left image is a pocket stereoscope with original test image. Used by military to examine stereoscopic pairs of aerial photographs. Right image is a view of Boston, c. 1860; an early stereoscopic card for viewing a scene from nature. . . . . 3

2 A sample Red-Cyan Anaglyph image taken at Concordia's EV building. To composite an anaglyph image, one can take one photo, then shift the camera along a direction for a short distance, then take another picture, and fuse them together . . . . . 4

3 Stereo rendering through 3D geometries, image from [pag] . . . . . 5

4 Homography Matrix vs Fundamental Matrix, homography matrix(left) map pixel to pixel and fundamental matrix(right) map pixels to a line. Images are from OpenMVG project . . . . . 6

5 Here are 10 frames of anaglyph generated for a sample video from fundamental matrix for consecutive frames using the approach which is discussed in this section, the images are from the paper of their work [LRKL13] . . . . . 7

6 Make3D model for 3D reconstruction, here you can see two superpixels plane connected together, ray $R$ is project on the planes and $d_1$ $d_2$ are the distance, image is from Make3D work [SSN09a] . . . . . 10

7 superpixel matching pipeline . . . . . 14

8 SLIC searching scheme . . . . . 15

9 Superpixel representation, the image above is broken into non overlapped superpixels, a typical size image usually has a few hundred to one thousand superpixels. . . . . 15

10 Sector map of superpixels, the image on the left in the superpixel contour, the right is the visualization of sector map. The sector map is coded in $[0 - 15]$ . . . . . 17

11 Superpixel graph . . . . . 18

12 Two different set of matches, left image is the one has the superpixels to find matches for, the middle image have matches far away from each other, the right image have matches closer to each other . . . . . 19

13 The optimizing process, image (a) is the matches before step, image (b) is the matches after optimizing step. Notice the **RED** line in (a), after moving the match, we can see that energy of geometry distances of matches decreases . . . . . 21

14 Pipeline with the neural network components . . . . . 22

15 The diagram of the Convolutional Neural Network architecture we are using The first scale is represented in white, the second scale is in red and the last scale is blue . . . 23
16 Output of the neural network: the image of the top row is the orignal RGB image; the two images on the second row are the low resolution depth and normal map; the last row are the finer resolution depth and normal map. Note that depth and normal map are post-processed with montage technique to be more colourful . . . . . 24
17 Original RGB image from Depth-Transfer Dataset . . . . . 27
18 Depth map captured from Kinect . . . . . 27
19 A rendered left-view of depth map, note that there is a small disparity between this left view and original view . . . . . 27
20 Camera Setup illustration . . . . . 29
21 Components of the projects . . . . . 31
22 The graph matching algorithm . . . . . 32
23 Single image Depth Transfer result(without smoothing), the right two grey scale images represent the depth . . . . . 33
24 The original sample image . . . . . 34
25 rendered anaglyph without in-painting . . . . . 34
26 Rendered anaglyph with too large focal lens settings that causes discomfort . . . . . 34
27 rendered anaglyph after in-painting . . . . . 34
28 rendered anaglyph using new rendering formula . . . . . 34

# List of Tables

1 Evaluation, per-pixel error for re-generated depth map . . . . . 33

# Chapter 1

# Introduction

It has been more than a century since the first video of human history; the earliest surviving film recording is from 1888 and it shows traffic crossing the Leeds bridge in England. Since then a enormous amount of videos have been recorded, including amateur recordings as well as professional films, the majority of which were filmed using a single camera.

Recently the enormous progress in the field of virtual reality and particularly the release of affordable VR headsets e.g. Google cardboard, Facebook Occulus, HTC Vive, Samsung Gear has made it possible for the general public to directly experience and interact with 3D content. Stereo video cameras are starting to become popular for recording 3D videos and many recent films have already used them. However, videos captured with a single camera cannot be easily converted to 3D. There are commercial films which were originally shot in 2D converted into 3D films later on as an post processing step. For example, IMAX provides such service to the film industry. The process takes massive human effort and a limited number of human-in-the-loop software [evil twin, etc] are written in order to aid this specific task. Nevertheless, converting a 2D video into 3D requires extensive human interaction which makes the process time-consuming and expensive. An attempt in automating this process was made by Google's Youtube with the "3D converter" which briefly appeared as new functionality but has now been removed likely because of the poor performance.

The method to produce stereo image or content is called stereoscopy, it has years of history. In the next chapter, we will briefly present the capturing, displaying, rendering techniques which exist so far. Then rendering part is the major focus in the thesis. It can be treated as a computer vision problem as an application of depth perception from images. A handful of other solutions such as [SSN09b] [RVCK16a] [LSH14] have been proposed to extract depth from images and videos. Essentially these methods are for retrieving depth information from monocluar or binocular images. We will spend some explaining those cues and related studies utilizing those cues on chapter 2 as well.

With the study of previous works, in this thesis we address the complex problem of automatically converting a video filmed with a single camera to stereoscopic content tailored for viewing in a 3DTV environment. Similar to other techniques we create a plausible depth interpretation of the scene and we focus on generating stereoscopic videos which does not cause depth perception inconsistencies

to the viewer. In the thesis two different methods will be discussed. The first one is an energy minimization optimization based approach on superpixels [ASS+12a]. The method takes rgb images as input, breaking them into superpixels then finding the superpixel with similar appearance features inside the database and label the superpixel with different depth. However this optimization based method has its inherited limitation as it is bind to the database, so we adopted the second method, as opposed to first, it employs a Convolutional Neural Network, which, given an RGB image generates a depth map. The rendering part comes next. We used two different stereo setup to render RGB images with the depth map to generate the virtual view for the second camera. Using the two views and the disparities between the pixels, an anaglyph image is then rendered. Common artifacts due to rendering from depth are addressed using in-painting. The proposed technique has been tested on several videos and the results are reported.

# Chapter 2

# Literature Review

This chapter, literature review goes as in following order. Firstly we discuss the brief history of 3D video, followed by a short review on existing 2D to 3D converting techniques. Then we will focus on depth based methods and the depth cues available in 2D images and videos, and finally state-of-the-art depth extraction methods.

## 2.1 3D video Production

The so called 3D videos are the media content people view with a stereoscope. A stereoscope is essentially a pair of glasses. As shown in Figure 1, each lens passes through a image from different view port [Wik17]. It utilizes the parallax of human eyes to produce 3D depth perception. The techniques used in creating or enhancing such 3D videos is called stereoscopics. It was firstly introduced by Sir Charles Wheatstone[Wel78] in 1838 and he showed that depth effect appears in the brain when viewing two pictures stereoscopically.

Figure 1: Left image is a pocket stereoscope with original test image. Used by military to examine stereoscopic pairs of aerial photographs. Right image is a view of Boston, c. 1860; an early stereoscopic card for viewing a scene from nature.

Stereoscopics involves topics such as capturing and displaying and rendering. Capturing stereo photos usually requires two images captured by two parallel lenses. In special cases, if the scene to be captured is static, it can be achieved by moving the camera by certain offset along an axis. Capturing 3D video, however, requires synchronized cameras. Existing devices like ZED stereo cameras[Zed] or

more professional gear such as IMAX 3D cameras can capture two views synchronously. However, the cost of a such gear is usually high and it is not well accessible to public.

3D video displaying technology has evolved from stereoscope to many different new techniques. The principle is presenting a different view for each eye. Side-by-side displaying is one of the earliest method which presents the pair of images right on the glasses and each eye can only access one view, an sample image is figure 1.

Figure 2: A sample Red-Cyan Anaglyph image taken at Concordia's EV building. To composite an anaglyph image, one can take one photo, then shift the camera along a direction for a short distance, then take another picture, and fuse them together

Another type is free-viewing techniques, which blend two images into a single image using a fusing equation, called "Anaglyph". It is a simple blending function and we used it in the thesis. It is suitable for displaying large images or videos thus suitable for cinema experience. Although the 3D projected images are not 3D ready directly to the human eye, it can be "re-separated" through stereo glasses. There are different variants of fusing equation. For example, the **Red-Cyan** anaglyph follows the equation

$$R_{final} = R_{left}$$
$$G_{final} = G_{right}$$
$$B_{final} = B_{right}$$

It uses **Red** channel of the left view, **Green** and **Blue** channel of the right view thus the

left view only passes through left lens and right view passes the other. This method is called **Red-Cyan** anaglyph. It has good color perception of green and blue, not red, and it is currently the most popular one in use. Aside from that, other methods exist as well. Such as **Red-Blue**, **Anachrome**, **Trioscopic**, etc. Figure 2 is an example of of **Red-Cyan** anaglyph image.

## 2.2 Stereo Rendering

As mentioned above, 3D content can also be generated through rendering. It is the main topic of this thesis. The traditional way is through the graphics pipeline. One starts from 3D meshes and materials, using lighting model, ray tracing, shadow mapping, etc and finally render the 3D geometries into stereo views through two different camera matrices. Such technology is called single pass stereo rendering, which is supported by modern graphic cards. It vastly reduced the computation complexity thus catalysed the expansion of VR gaming. Figure 3 illustrates such graphics rendering technique

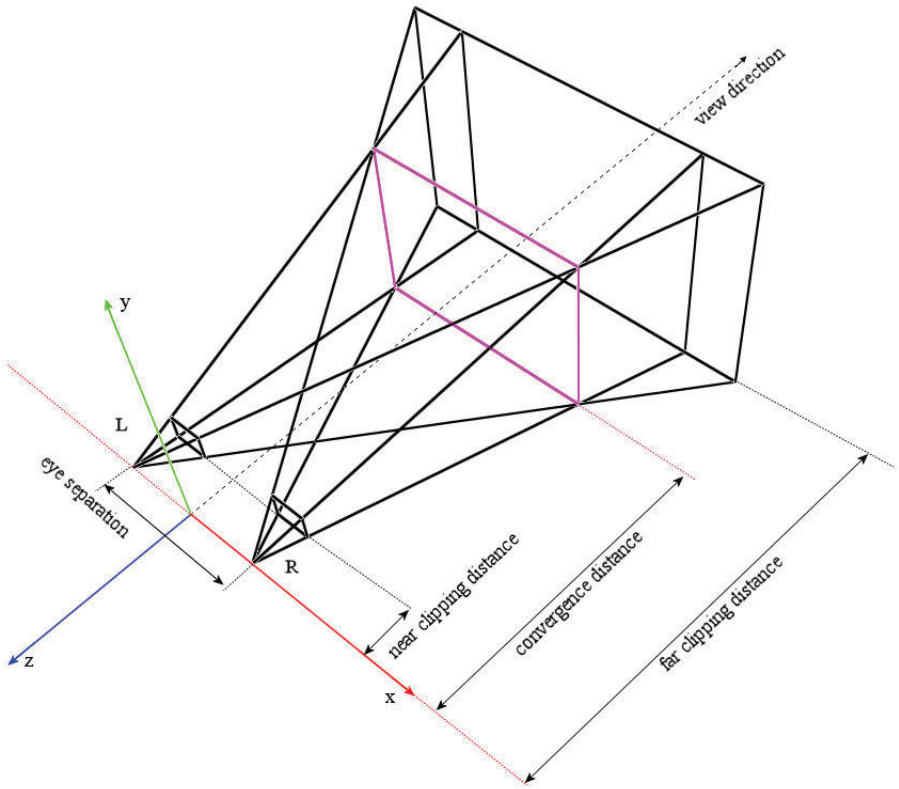



Figure 3: Stereo rendering through 3D geometries, image from [pag]

If the images or video are produced beforehand, one need to post-process such media into (pseudo) 3D, either manually or automatically. Post-processing a 2D video frame-by-frame requires massive human labour. Automating such processes is an open research area. In this section, we are reviewing several proposed 3DTV technologies, their highlights and limitations.

### 2.2.1 Stereo rendering by perspective transform

In the work of [LRKL13], a rendering techniques which based on fundamental matrix is proposed. It explores the research in using homography to rectify the image for generating stereoscopic 3D. Supported by their experiments, pseudo stereo effects is archived by rectifying two images into the same image plane. The method is effective for generating 3D anaglyph for personal photos, but it was not stable in generating 3D anaglyph from consecutive frames.

The inherited problem with this approach is that the homography is a special case for image space transform. This is because usually images from different views do not have pixel-wise mapping. Figure 4 is an illustration. In general cases, one can only find the fundamental matrix from one image to another as shown on the right. The special case is the mapping from one plane to another, where point-to-point correspondence can be recovered.

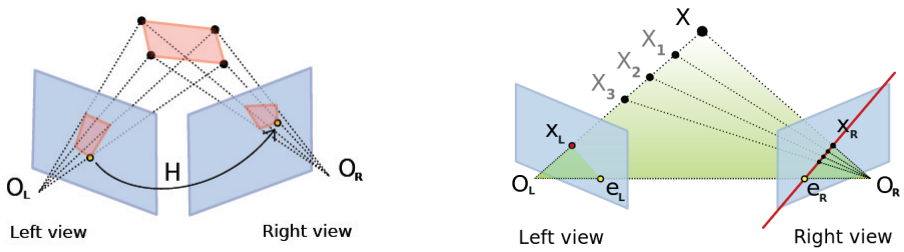



Figure 4: Homography Matrix vs Fundamental Matrix, homography matrix(left) map pixel to pixel and fundamental matrix(right) map pixels to a line. Images are from OpenMVG project

Visually it is still plausible even if we are finding the homography matrix for non-flat scenes, but in videos when we move the camera, at every frame, the homography matrix is changing as well, which causes noticeable non-smoothness. Another downside of homography mapping reported in the paper is the randomness of RANSAC algorithm, recently there are improvements in conquering this factor[KTS11].

In the method [LRKL13], the pipeline starts with (SfM) Structure from Motion. The poses of camera is vital to the approach. After breaking a video into frames and estimating camera poses, two of all frames from the video are selected and a fundamental matrix is estimated between the two. The fundamental matrix is then transferred to a homography matrix to reconstruct sythesis views for anaglyph 3D. The transferring technique is based on the following equation:

$$F_{ij} \approx [e_{ij}] \times H_{ij} \tag{1}$$

The $F$ is created from homography and epipolar line $e_{ij}$ between two frames. They assume the projective matrix for the first camera is $P = K[I|0]$ and the second is $P' = K[R|t]$. The second view is generated from $H = K[R_1 R_2 t]$. $R_1$, $R_2$ is from the SfM pose estimation and $t$ is from fundamental matrix. They normalized the $t$ to archive more stable result. An examples of their result is shown in figure 5

Although this work provides a simple, elegant solution to stereoscopic rendering, its limitation are also apparent.

Figure 5: Here are 10 frames of anaglyph generated for a sample video from fundamental matrix for consecutive frames using the approach which is discussed in this section, the images are from the paper of their work [LRKL13]

- Firstly, the solution needs a moving camera in the scene
- The scene has also be static

If the environment does not satisfy the two conditions, the result is undefined. For the first requirement, a recent work was done to improve on bundle adjustment [HIP+16] for small camera motion, such improvements may be used to relax the first condition.

### 2.2.2 Depth Based Rendering

The main stream of stereo rendering techniques is **DIBR**(Depth based stereo rendering)[Feh04]. It was introduced in "Advanced Three-Dimensional Television System Technologies"(ATTEST) project. In the thesis we derived the rendering formula on our own. The full deduction of the rendering formula is described in section 3.3.

Briefly, the work in [Feh04] concluded that one can simulate the depth of field effects in synthesized images with image-plus-depth stream without un-projecting the pixels from image space to 3D spaces. Camera positions(extrinsic matrix) and focal lens(intrinsic matrix) are not required. Since it is one of our major work, we will continue the discussion in section 3.3.

In our case, the depth map is not available with RGB image stream. We need depth perception techniques in order to render anaglyph 3D. In section 2.3, we will discuss various depth cues available in images as well videos and in section 2.4, we will present related depth estimation methods.

### 2.2.3 Cue fusion based rendering

The section discusses the work of [LKR+17], they proposed a system to infer binocular display from monocular video streams in 2016 based on fusing different depth cues together. Visually plausible results are generated at the end of processing.

Their pipeline consists of two main stages, a pre-processing step and a runtime step. In the beginning of the approach, the system extracts the priors of the video by classifying the video into different classes. The prior model is based on gaussian model with certain mean $\mu_0$ and variance

$\sigma_0$ for different scenes, different colours and different pixel locations in the images. For example, in forest scenes, lower green pixels have a medium depth. In scenes with water, blue pixels in the top part of the image have far depth and as a contrary, in the lower part of image, blue pixels have low depth.

The next step in the pipeline utilizes multiple depth cues in order to infer disparity and associate confidence. In the work, six depth cues are used. Aerial perspective, defocus, vanishing points, occlusion, motion and user input. However, they [LKR+17] did not mention how those depth cues are equationrized in the paper.

The last step of the approach is fusing different cues together and estimating the disparities. For every pixel $x$, a MLE(maximum likelihood estimation) method is used with the equation

$$\mu MLE(x) = \frac{1}{Z}\Sigma_{i=1}^{n_c}\left(\mu_i\beta_{i,c}, \sigma_i^{-2}(x)\right) \tag{2}$$

where $x$ is the disparity to estimate, $\mu_i$ are the cues, $\beta$ and $\sigma$ are the parameters and $Z$ is the normalizing partition function. The equation is an average of disparity means. If prior is included, the estimation becomes a Bayesian estimation and MAP(maximum a posterior) is used with the following equation

$$\mu MLE(x) = \frac{1}{Z}\Sigma_{i=1}^{n_c}\left(\mu_0(x)\beta_{0,c}\sigma_0^{-2}(x) + \mu_i\beta_{i,c}, \sigma_i^{-2}(x)\right) \tag{3}$$

This equation is similar to equation 2, difference is adding of priors $\mu_0$, $\beta_{0,c}$, $\sigma_0$.

## 2.3 Depth Cues

This section we discuss a few depth cues available in monocular images.

### 2.3.1 Depth from Parallax

Parallax forms when viewing a object from two different locations and it is stable. The depth cue is available as long as two different camera views are available. Parallax can be visualized when driving a car. When drivers or passengers look outside, the nearby objects moves quickly, where the far objects move relatively slower.

Capturing parallax is a well developed process as well. If binocular vision is available, for example, stereo cameras can be used. One can calibrate the relative location of the stereo camera, then match the feature points available in both images using the triangulation to determine the absolute depth. If binocular vision is not available, parallax can also be found by moving the camera. The Structure from Motion(SfM) method uses such technique to determine the camera location as well as determine the scene depth. The limitation in such cases, however, is the static scene requirement. SfM also requires user to try a few times to capture enough views. Bundle adjustment is a common step after SfM to reduce the projection error. Accurate Camera parameters can be used in 3D reconstructions.

### 2.3.2 Depth from Linear Perspective

Human eyes see the world in a perspective geometry, unlike its orthogonal geometry counterpart, it is such a space where parallel lines merge together at infinity. The only property kept is the "straightness". Perspective is not always available, an object that does not expand over a long range of depth may just not have perspective. In man-made scenes, perspective is easier to detect than natural scenes since the architectural scenes often contain many straight lines and shape angles. Those straight lines can be found by line accumulation algorithms. This process is called vanishing point detection. To address such problem, one must consider the two sub-problems: 1) How to cluster line segments going to a single vanishing point and 2) how to estimate an accurate vanishing point. Hough transform is one of such methods for finding line segments. An important application of vanishing point detection is auto driving. To keep the vehicle at the centre of the road, it should drive towards the vanishing point of the road. Note that even without perspective, depth detection can also be found using other techniques.

### 2.3.3 Depth from Occlusion

Occlusion happens when nearby objects occlude far object. and can be a strong cue for determining relative depth. It can be found by detecting T-junctions of edges and lines. Like the previous cue, occlusion may or may not be available. Moreover, occlusion often causes difficulty in 3D reconstructions[HWB12], because camera cannot capture the shape of objects occluded by others. For the same reason, occlusion also brings difficulties in rendering views.

### 2.3.4 Other depth cues

Besides parallax, perspective, occlusions, depth cues like defocus, similar size, texture can also be helpful in determining depth changes or absolute depth.

## 2.4 Depth Map Estimation

Depth perception is a basic human aptitude. Giving machines the same ability, however, involved a long history of efforts. Currently, techniques like Structured Light Scanning, stereo matching triangulation and structure from motion are already mature and they work well in their own specific conditions. For example, structure light scanning method projects patterns onto STATIC objects and then finds the matches based on patterns, it is basically a two-view-stereo method and the result is accurate enough for 3D mesh reconstruction. Structure from Motion(SfM), on the other hand, is a multiple-view stereo algorithm and it requires users to be patient, move the camera smoothly and sufficient camera motion is required. Stereo matching requires two synchronized cameras and matching two views frame by frame, the matching result may be noisy if the baseline of two cameras is not well set up. Both Structure from motion and Stereo matching can only produce sparse XYZ points clouds. These methods are binocular vision, and these techniques are mature because it utilized a very reliable depth cue, parallax.

Monocular depth perception, on the other hand, is still an open research problem. In recent years the attention is shifting from model based approaches towards data driven approaches. In this chapter, we will explore both of these approaches.

### 2.4.1 Parametric based method

Make3D [SSN09a] is a statistical approach to infer 3D scene structure from single still images based on Markov Random Field(MRF) to learn a set of "plane parameters". The result of the approach is an image texture which is mapped to the 3D mesh that inferred my this approach. Viewers can roughly see the 3D structure from a single image by changing the viewports.

To formalize the approach, we are introducing a few terms used in this method: **superpixels**, **co-planar structure**, **so-linearity**, **connected structure**. Superpixels $i$ (one type of oversegmenting method) represents a surface in an image, with a centre position $q$, normal vector $\alpha$ and depth value $d_i = 1/R_i^T$. By their assumptions, those superpixels may be connected together, may lie on a mutual plane, may be part of a huge linear structure. Image 6 is an illustration of their model to infer depth of an image.

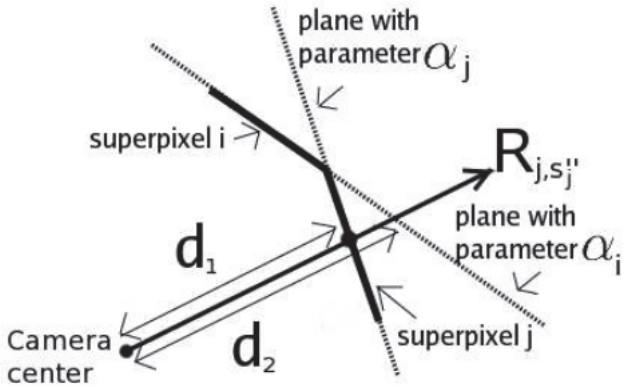



Figure 6: Make3D model for 3D reconstruction, here you can see two superpixels plane connected together, ray $R$ is project on the planes and $d_1$ $d_2$ are the distance, image is from Make3D work [SSN09a]

For the **Connected structure**, the goal is to infer a $y$ value on the edge image, $y_{ij} \in [0, 1]$ as soft value Instead of $\{0, 1\}$, because they believed in many cases, an edge does not correspond to an occlusion or folding. The model $y_{i,j}$ is based on a logistic response as $P(y_{i,j} = 1|\epsilon_{i,j}; \psi) = 1/(1 + \exp(-\psi^T \epsilon_{i,j}))$. Where $\epsilon_{i,j}$ is the feature, $\psi$ is the model parameters.

The main part of the project is the MRF on planar parameters. It consists of single superpixel parameters and plane parameters between two superpixels. It is formulated as following.

$$P(\alpha|X, \nu, y, R; \theta) = \frac{1}{Z} \prod_i f_1(\alpha_i|X_i, \nu_i, R_i; \theta) \prod_{i,j} f_2(\alpha_i, \alpha_j|y_{ij}, R_i, R_j) \tag{4}$$

The main purpose of the equation is to infer $\alpha$, the plane parameter, by the conditional variable such as features $X$; ray(normal) for the superpixel $nu$ that shoots from camera to the pixels on superpixel with the confidence of the depth prediction $\nu$ and the model parameter $\theta$. The ray $R$ and plane has the relationship that the dot product of the two is the depth. From this, it could formalize fractional error of depth map $d_{i,s_i}$ into $R_{i,s_i}^T \alpha(x^T \theta_r) - 1$

The model of $f_1$ is for minimizing the fractional error, to make the error in the range of $[0, 1]$, they used negative exponential function.

$$f_1(\alpha_i | X_i, \nu_i, R_i; \theta) = \exp\left( - \sum (R_{i,s_i}^T \alpha(x^T \theta_r) - 1) \right) \tag{5}$$

The second part $f_2$ is more interesting; it models the relation $h_{s_i,s_j}$ between the plane parameters of two superpixels $i, j$. It consists of the **connected structure** model, **Co-linearity** model and **Co-planarity** model. In the end, a MAP Inference is used to minimize the errors.

The last piece of the puzzle is the image features; here they used responses from the 17 different filters. The first 9 are the Law's mask(can be found in [SSN09b]). The last 6 are oriented edge detectors. They apply the mask on different scales and uses adjacent patches for features.

### 2.4.2 Non parametric based method

The Non-parametric method by [KLK12a] is a type of technique which uses **similar** images' available depth map to estimate depth for the target images or videos. In order to utilize method, the entire database needs to be available at runtime. When similar images candidates are found, a **candidate warping** process is executed.

The pipeline is straightforward. Firstly, candidates frames for a given image are searched by the K-Nearest-Neighbour(KNN) method, it is too costly to walk through the entire database image by image to perform warping so they used GIST [OT01] descriptor to encode the large images into a few hundred bytes. KNN returns the top 7 candidates frames for the matching step. To have enough variety of matching candidates, they ensured each video in the database contributes no more than one matching frame.

As the candidate frames closely match the input frames in the feature space, they should also share the very similar semantics of the scenes. A candidate warping processing is the next step. Here, the critical assumption the this work is that *the distribution of the depth is comparable among the input and the candidates*. Based on this assumption, the warping is done by finding the pixel-to-pixel correspondence from a target frame to a candidate using the SIFT-flow[LYT11] method. SIFT-flow is essentially a multiple scale matching method. The idea of it is expanding the search area in the dense optical-flow algorithm from small patch into entire image. For every pixel, the SIFT descriptor is calculated by sampling a patch around that pixel. As every pixel is sampled, this will encode an image into a SIFT image. Directly performing matching on SIFT image to SIFT image costs too much, thus a coarse-to-fine fashion searching is applied instead. It operates as following:

- perform the matching for every pixel on the very small scale

- After the matches are found, expand the image to a higher scale and limit the search locally to higher scale match
- perform step 1 until the highest scale

Depths per-pixel are transferred from the candidates to the target using the SIFT-flow matching method, as we have 7 candidates, 7 transferred depth map is available at the moment. An energy minimization algorithm is performed as the next step to smooth the transferred depth map. The energy is determined in equation 6

$$E(D) = -\log(P(D|L)) = \sum_{i \in pixels} E_t(D_i) + \alpha E_s(D_i) + \beta E_p(D_i) + log(Z) \tag{6}$$

$E_t$ is the data term that measures the difference between optimized depth $\hat{d}$ and depth $d$ in the candidates. $E_s$ is a smooth term that penalize the depth difference between adjacent pixels. $E_p$ is the difference between $\hat{d}$ and the prior depth(average all depth images in database). Based on [KLK12a], this method out performed the Make3D method by a small difference of the 534 images Make3D dataset.

### 2.4.3 Model based approaches

A geometric based method [RVCK16b], which does not require training and can be applied to nature scenes, was proposed in 2016. The approach comprises two stages. A **motion segmentation** stage and a **reconstruction stage**. Unlike the previous methods, such methods must runs on consecutive frames, but unlike the Structure from Motion(SfM) technique, it can handle dynamic scenes to some extent.

The Motion segmentation stage takes dense optical flow field $f = (f_x, f_y)$ as input and segments into $L + 1$ rigid motion labels $u_l$, as a variational labeling problem. If the motion is rigid, it automatically oversegments non-rigid objects into approximately rigid parts. It is formulated as follows:

$$(u_l, F_l) = \arg\min \sum_{l=1}^{L+1} u_l \cdot g(F_l) + ||W_l \Delta_{ul}|| \tag{7}$$

Where $g(F_l)$ is the symmetric distance to epipolar lines for each motion model $l$ and homogeneous coordinates $x^i = [x^i, y_i, 1]$

$$g(F_l) = d(x_1^i, F_l, x_2^i)^2 + d(x_2^i, F_l, x_1^i)^2 \tag{8}$$

$W_l$ is a smooth term and $F_l$ in the equation is the fundamental matrix. The energy is solved based on an iterative algorithm described in paper.

The second part is the **Reconstruction** by triangulating the scene. It begins by estimating each inlier correspondence in the motion segmentation stage using the motion model $F_l$, which yields a set of depth estimation $\{z_l\}$. Then the relative scales of each individual $z_l$ needs be estimate. It took a **connected scene** assumption, dynamic objects occludes static objects assumption and a smoothing constraint. To ensure the robust estimations, they also adopt a superpixel segmentations,

for each superpixel $k$ $\theta_k = [\theta_k^1, \theta_k^2, \theta_k^3]$, and $L$ different depth scale for previous triangulation result. For all independently moving objects, The estimation is done by

$$E(s, \theta) = E_{ord}(\theta) + E_{sm}(\theta) + E_{fit}(s, \theta) \tag{9}$$

It has an ordering term, a smoothness term and a fitting term. The ordering term ensures that dynamic objects occlude static objects and the smoothness term ensures the two superpixels coincide at the boundary. The fitting term tries to find the correct scale for the motion models in order to satisfy the previous two.

# Chapter 3

# Methodology

The implementation of the stereoscopics is described in this chapter. Several methods were used. The first implementation is Superpixel matching based method. The second one is a Neural network based method.

## 3.1 Super-Matching Method

The first step towards rendering stereoscopics is generating the depth-map for images. The method described in this section is a superpixel matching method. the first step of the method utilize the same assumption [KLK12b], which is fact that that the similar scenes share the same depth distribution. However, in the following step, we extended it to the superpixel level. The method followed the pipeline in figure 7. In the beginning, 7 similar images are retrieved through GIST descriptor[OT01] based on their gist similarities as candidates, we then break RGB input frames and candidates frames into SLIC superpixels for matching. We will describe the matching in detail in section 3.1.3. As every superpixel is assigned to its matching counterparts, we can transfer the depth value of matches into input images. Afterwards, it requires a smoothing procedure as the matching may produce spatially incoherent depth distribution. To achieve spatial smoothness, an optimization is used to produce final results.

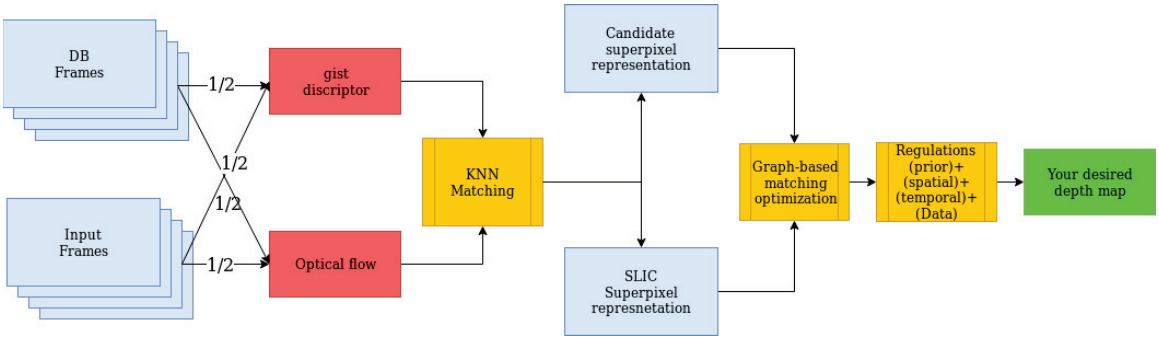



Figure 7: superpixel matching pipeline

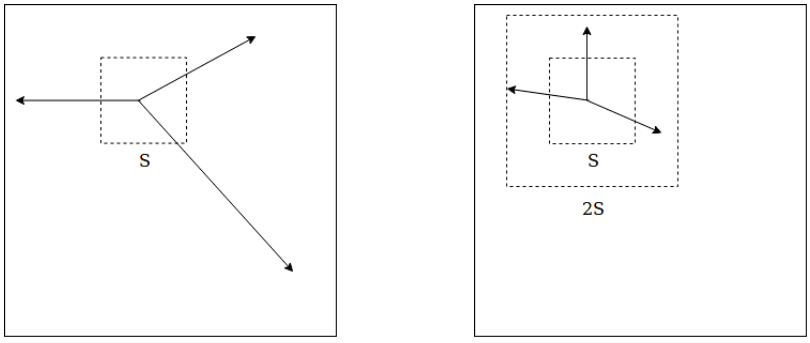
### 3.1.1 Superpixel Representation

Both candidates and RGB input need to be broken into superpixels. Here we briefly introduce the superpixels method. The over segmentation algorithm partitions images into non-equal size segments. A superpixel represents a group of adjacent pixels with the following properties

- the pixels in the same superpixel tend to share the similar color.
- image edges(can be recovered by edge detection) mostly lies on superpixels boundaries.

Variants of over segementing algorithms were proposed and the implementation we used is SLIC [ASS+12b]. It is a k-means clustering algorithm with minor difference. As shown in figure 8. At updating step, instead of search entire image, the search space is a double superpixel size rectangle. The distance measurements consist of color distance $d_c$ in CIELAB color space and pixel's position distance $d_s$ and they merge into single measure formula 10. Where the $N_c$ and $N_s$ is the norm of each measurement.

$$D = \sqrt{(\frac{d_c}{N_c})^2 + (\frac{d_s}{N_s})^2} \tag{10}$$




(a) Traditional K-means clustering algorithm

(b) SLIC limits search space in $2S$

Figure 8: SLIC searching scheme

A sample superpixel representation is shown here.

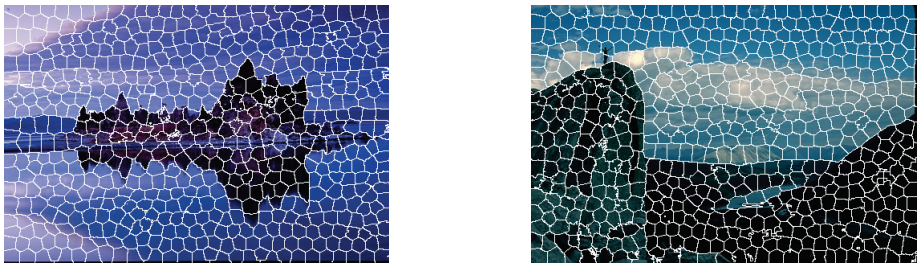


Figure 9: Superpixel representation, the image above is broken into non overlapped superpixels, a typical size image usually has a few hundred to one thousand superpixels.

### 3.1.2 Descriptors for Superpixels

As we mentioned in the pipeline 7, descriptors generation is the next step after we have the superpixel segments for every image. The descriptors serve as a metric for matching. Feature descriptors like Scale Invariant Feature Transform [Low04] and Speed up Robust Features [BETVG08] are working on fixed shape feature points methods. Those methods sample a patch around the feature points, the shape of the patch depends on the method. For example, SIFT uses a $16 \times 16$ rectangle patch, DAISY [TLF10], uses a circle like patch for computing descriptors. Our case is a little different, our patch is determined by the area of superpixels. Patches are not in consistent shape so sample method has to be different as well.

Still, we need good a appearance descriptor for superpixel representations. We considered hand coded descriptors, such as super-parsing [TL10]. In the method, the descriptor for every superpixel coded with information like global information, shape, location, texture, etc. The dimension of the descriptor turned out to be a few thousand. Similar approach like PatchMatch [BSGF10] also adopted similar method, but they also consider adjacent top, left, right, down patches when encoding descriptors.

Here we are measuring the similarity of appearance of superpixels .We are not willing to get too complicated, so we applied a variant of SIFT descriptor. It still based on Histogram of Gradients and has the same shape of the SIFT descriptor. The Method for constructing descriptors is described in .

**Algorithm 1:** Superpixel Descriptor construction

**Data:** Superpixels sps
**Result:** Superpixel descriptor $D$

```
A, M ← magnitude and angle from applying Sobel filter;
B ← superpixel pixel label mask;
L ← List of superpixels;
S ← initalize sector map for image for i in L do
    l ← pixels of sp;
    for p in in sp do
        compute witch of the 4 × 4 sectors p falls in;
        update sector map;
    end
end
for pixel in image do
    compute the location of sector with sector map S and angle A;
    updating descriptor with M;
end
normalize D;
```

At the end of the algorithm, we are able to construct a size of 128 descriptor. Each descriptor is divided into 16 sectors with 8 bins in each sector. In the algorithm, we used a "sector map", it records which sector the pixel belongs to, ranges from 0 to 15. The sector index of that pixel is

determined by the relative location of the pixel in the superpixel. If the superpixel has ill shape(this means it does not has enough width and height to divide), we will skip processing it. As we have have sector map of the entire image, the bin of that pixel belongs to is an offset of angle in a sector. In this way. We are able to build an sufficient descriptor for superpixels. Here we have a visualization of the sector map in figure 10

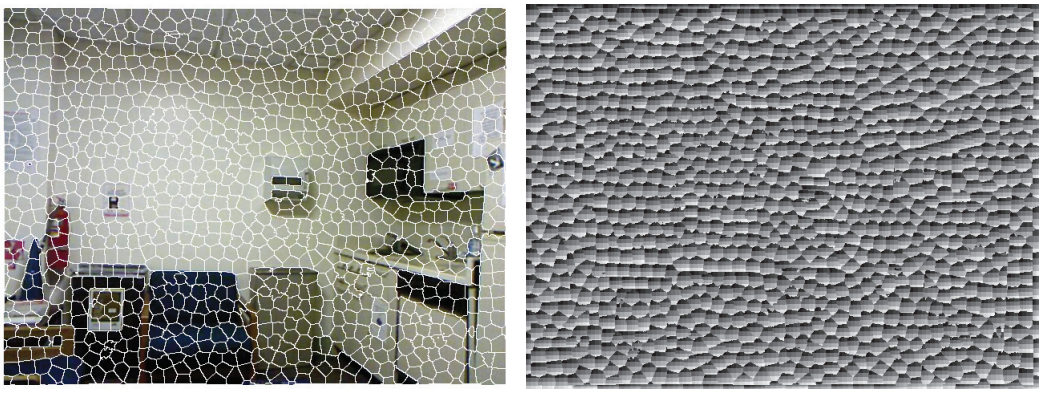


Figure 10: Sector map of superpixels, the image on the left in the superpixel contour, the right is the visualization of sector map. The sector map is coded in $[0 - 15]$

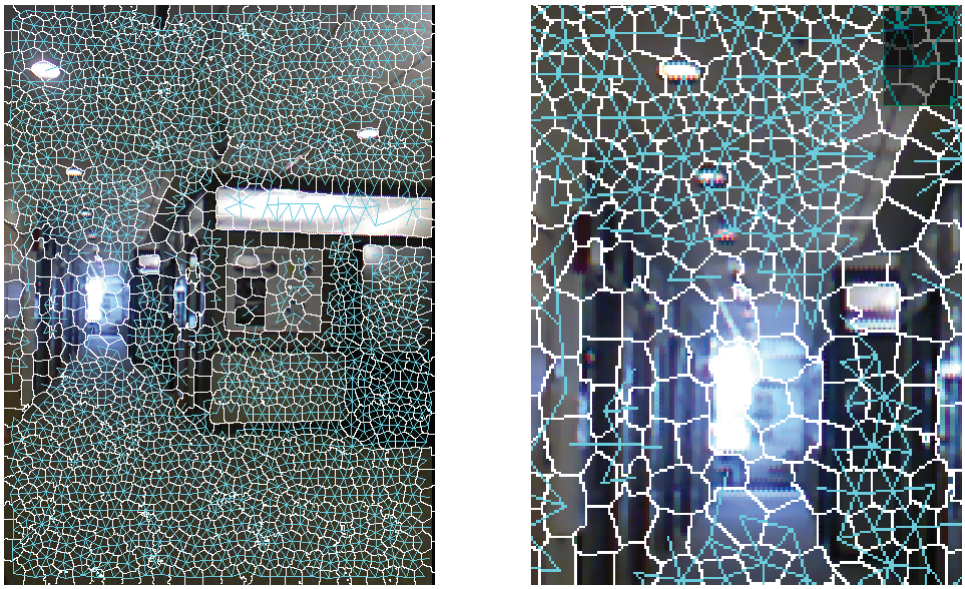
### 3.1.3 Superpixel Matching

After segmenting an image 9 into superpixels, matching those superpixels one by one is the next step. An typical image with $500 \times 500$ resolution can have less than 1000 superpixels. With computed descriptors we end up with an image of roughly $1000 \times 1000$ descriptor set. The matching operation on descriptor set to the candidates that we use GIST to retrieve from database is not real time, but it is still manageable. One can use superpixel descriptors as a metric and then use nearest neighbour matching method like **FLANN** to match superpixel independently. In this algorithm, however. We need to consider not just descriptor metric but also a smoothing metric. The details are explained in next section.

#### Graph Representation

After we segmented a image into superpixels, those superpixels naturally came with attributes like adjacency. To our benefit, the superpixel geometry relationship can be used as cues for local-correspondence-awareness. That is, *the adjacent superpixels should be matched to closed superpixels in other other image.* In the end of the algorithm, we still need to match candidates and RGB input densely. But instead of operating on pixel pyramid like [KLK12b], the operation is done on the superpixel graph. A superpixel graph consists of a set of nodes $N$(superpixels) and a set of edges $E$. Adjacency can be found by a searching algorithm. An example is shown in figure 11. If two superpixel are connected, that is, if there is no edge detected between two superpixels, a blue line is drawn between them on contours. Details of implementation can be found in section 4. After the

graph is built, we then formulated the our original depth rendering problem into a graph matching problem.



(a) An example of the graph of superpixels, the blue lines here indicate two superpixels are connected

(b) A closer look on the superpixel graph under a smaller range of the the image

Figure 11: Superpixel graph

Here are more details about superpixel geometry explanation. As we can see in figure 9 and figure 11, adjacent superpixels tend to belong to the same object, thus they shares similar depth values. If just applying descriptor distance metric, chances are adjacent superpixels get matches which are far from each other. As shown in figure 12, **SP1**, **SP2**, **SP3**, **SP4**, **SP5** on the left image have two different set of matches **SP1'**, **SP2'**, **SP3'**, **SP4'**, **SP5'** in the middle image and **SP1"**, **SP2"**, **SP3"**, **SP4"**, **SP5"** in the right image. Both set matches have high metric in descriptor distance, however, the matches in the middle are distributed in different location of the image which will produce depth map lack of coherence after transferring. The matches in the right image also have low distance in descriptor metric, but instead, the matches are also closer to each other. Thus the superpixel depth-transfer could produce closer depth values.

Given the information above, we could formulate the approach into a energy minimization problem, In section 3.1.3 we will describe in details.

### Matching as Energy Minimization

To address the local-correspondence-aware superpixel matching problem, we need to take two different metrics into count.

1. Superpixel descriptor distance
2. Adjacent superpixels matches distance in the matching image

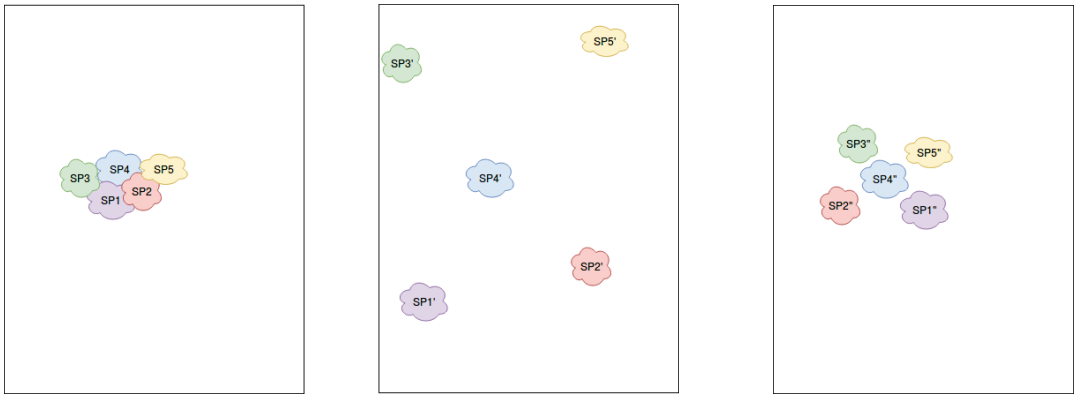



Figure 12: Two different set of matches, left image is the one has the superpixels to find matches for, the middle image have matches far away from each other, the right image have matches closer to each other

The two distances correspond to the two energy terms in the in the optimization process. With these two metrics, we can form two energy terms in order to minimize it. The superpixel descriptors distance forms the data term $E_d$. When we minimize the energy, we transfer the depth from more similar superpixels based on appearance. The second term is the smooth term $E_s$ because we want the matches in the target image to form a cluster instead of distributed across the entire image.

Formally, we are minimize the total energy $E$

$$\begin{aligned} E &= E_d + E_s \\ &= \sum_i \left(D(sp_i) - D(M(sp_i))\right)^2 + \sum_i \left( \sum_j d(M(sp_i), M(sp_j)) \right) \end{aligned} \tag{11}$$

In the equation 11, the first part $D(sp_i)$, $M(sp_i)$ are the descriptor of the superpixel $i$ and the matched superpixel $j$. $d(sp_i, sp_j)$ is the distance of superpixel $i$ and $j$ in the superpixel graph. To compute the second part $E_s$, one need to iterate through the matches of adjacent superpixels $j$ of superpixel $i$ and calculate the graph distance. To minimize the energy, we developed a mono-descent algorithm 2 to optimize the energy.

The algorithm runs until converge, At every iteration, it exams the energy of superpixel $i$ and all its adjacent superpixels $j$, because every update on it will only affect $i$ and $j$s. For those superpixels. If the energy change $e_c$ is negative, it means the match worth changing. thus we will re-assign the match. By the end of the processing, we would get a local minimum of the energy and corresponding matches as an output.

**Algorithm 2:** Graph Optimization Algorithm

**Data:** 2 Superpixel Graph $V_1, E_1, V_2, E_2$

**Result:** Optimized Match M

```
D ← All pairs shortest path;
M ← randomly assigned matches;
E ← Initial Energy;
Repeat ← true;
while not Repeat do
    Repeat ← false;
    L ← permutation of the V1;
    for i in L do
        m ← randomly match from V2;
        N ← adjacent superpixel of i;
        e_c ← 0;
        for j in N do
            e_c+ = d(m, M(j)) − d(M(i), M(j))
        end
        if e_c < 0 then
            Repeat ← true;
            M(i) ← m
        end
    end
end
```

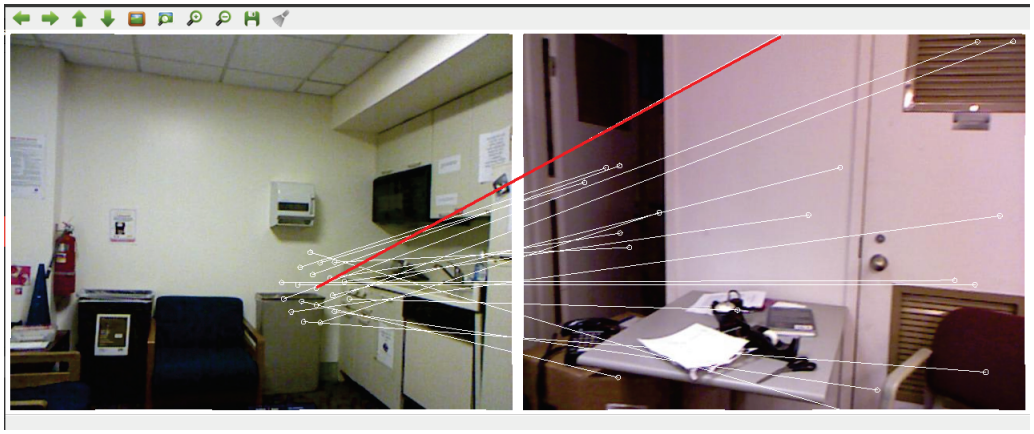
This optimization algorithm could produce local minimum. As the energy decreases step by step, it converges to a low energy point. But how close is this local minimum to the global minimum is not a guarantee. We are expecting to use `GraphCutOptimization` in the future.

Figure 13 is an example of running step of **Graph Optimization** algorithm. It optimize one match at a time, exams if the any match has lower energy then move the match to a new one. The final optimization results are shown in Chapter 4.

## 3.2 CNN Approach for Depth Prediction

In this section we are proposing a new approach to the depth map estimation problem, as the result that we realized our approach has its limitation, for example, it is not guarantee to have global minimum energy thus the result is not smooth enough. In addition to this, it has the inherited problem like depth transfer method [KLK12a], the method will not work if no similar images are available.

Although we have not found the silver bullet for mono depth cues for current state-of-the-art, we also realized that understanding image, depth perception are an very basic abilities for human or any other animal. Based on this, bringing up neural network on the task seems like a natural



(a)

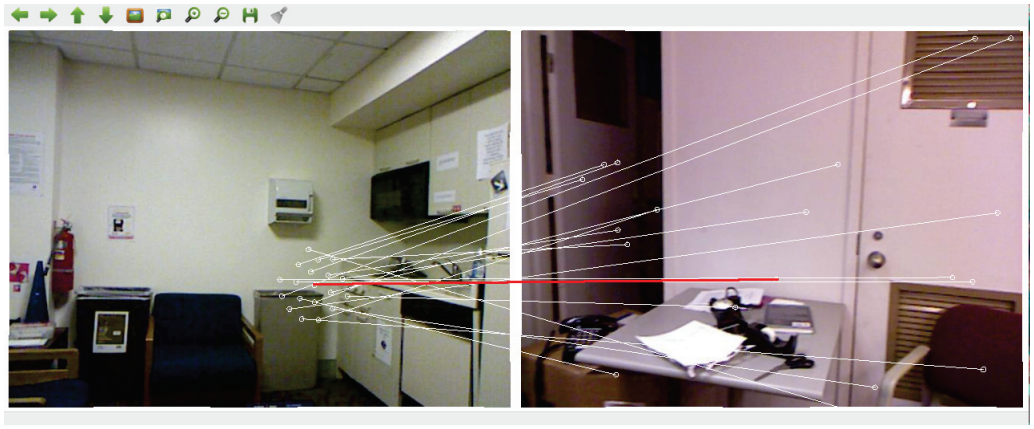


Figure 13: The optimizing process, image (a) is the matches before step, image (b) is the matches after optimizing step. Notice the **RED** line in (a), after moving the match, we can see that energy of geometry distances of matches decreases

.

choice. Some people may argue that as we have two eyes, the depth perception works like duo lenses camera, but there is no difficulty lies in distinguishing far and close objects for pictures or with only one eye. Neural network has proved to be very useful for many different tasks. Convolutional Neural Networks(CNN) as a famous branch, it has been successfully applied many recognition task. Started from LeNet [LBBH98], it is used to classify 10 classes of digits. Now the method is scaled up to classifying thousands of objects in images.

Before Neural network approach, the method we applied in the previous section and many methods before that [KLK12a], [SSN09b], [KB15] requires insight modelling the real world into mathematical models and one need to be careful to avoid non-convex problem, otherwise the model is hard to fit. In the end, the estimated depth will be able to work on specific scenes which fits their assumptions. The method we used is named non-parameters method, as they are using the K-nearest-neighbours method to search for depth. But essentially, this method is established on balancing different energies and heavily relies on the prior depth which is calculated at runtime. The study of such problem can start from features in images, then follow the long way of theories

like lighting model, perspective geometries, motion. At last one may end up too complicated work thus not addressable. Certainly studying all those methods individually is worthy but combining all of them to get it to work is difficult to get it to work.

So here we propose a second approach to the problem, with neural network in the pipeline, shown in the figure 14.

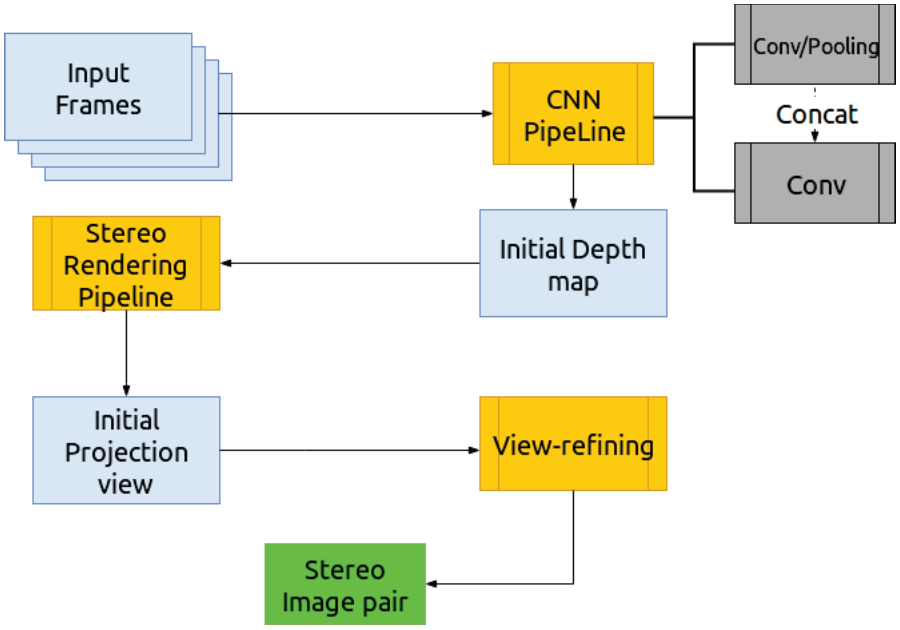



Figure 14: Pipeline with the neural network components

In the thesis we used an trained network architecture which was used to render the depth and normal maps. We used a multiple scale network architecture which was proposed in [EPF14], briefly those multiple scales are concatenated in a coarse-to-fine fashion in order to archive better resolution. The network is trained using RGB and depth map pairs using a loss function for the depths given by

$$\begin{aligned} L_{depth}(D, D^{\star}) = & \frac{1}{n}\sum d_i^2 + \frac{1}{2n}\left(\sum d_i^2\right) + \\ & \frac{1}{n}\sum |\bigtriangledown d_{x,y}| \end{aligned} \tag{12}$$

which measures the difference between the generated and ground-truth depth, and also the gradient of the depth because The gradient of the depth encourages local structure similarities. Similarly the loss function for the normals is defined as,

$$L_{normal}(N, N^{\star}) = -\frac{1}{n} N \cdot N^{\star} \tag{13}$$

which is equivalent to the cosine proximity.

An simplified architecture is shown with the diagram 15. As we can see, there are 3 scales of network, by concatenate the output of last scale to the beginning of next scale the network is thus formed. In section 3.2.1, 3.2.2, 3.2.3 we will describe the network in detail.

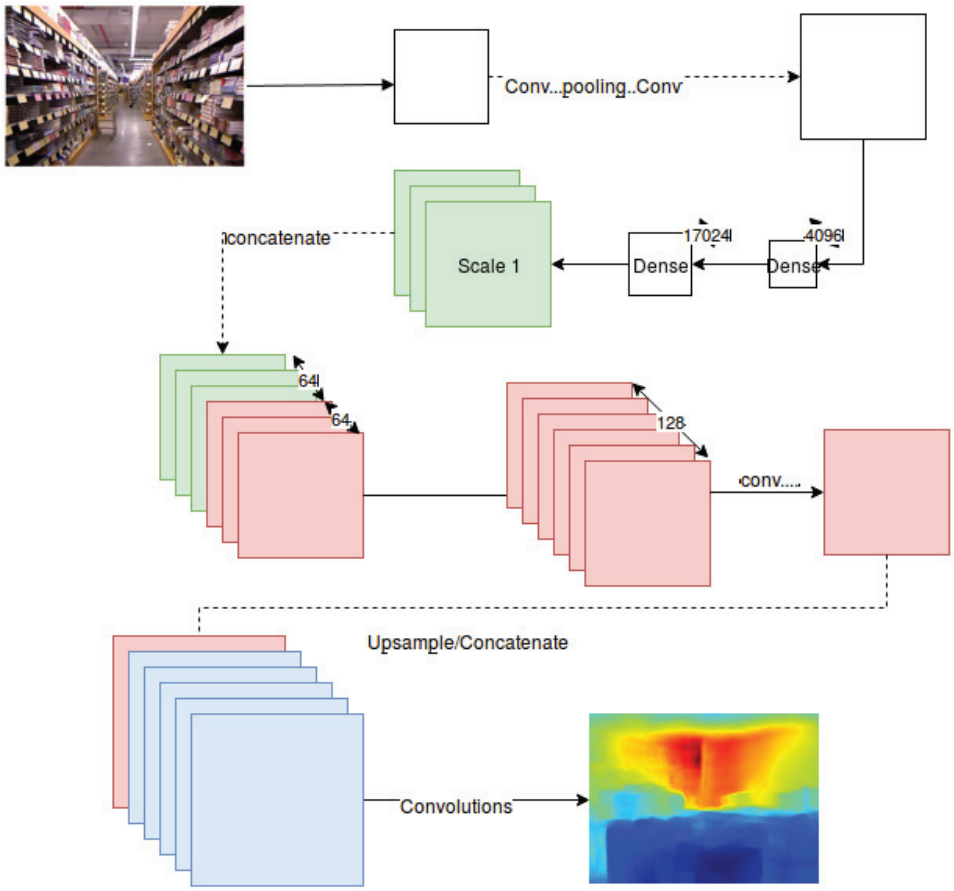



Figure 15: The diagram of the Convolutional Neural Network architecture we are using The first scale is represented in white, the second scale is in red and the last scale is blue .

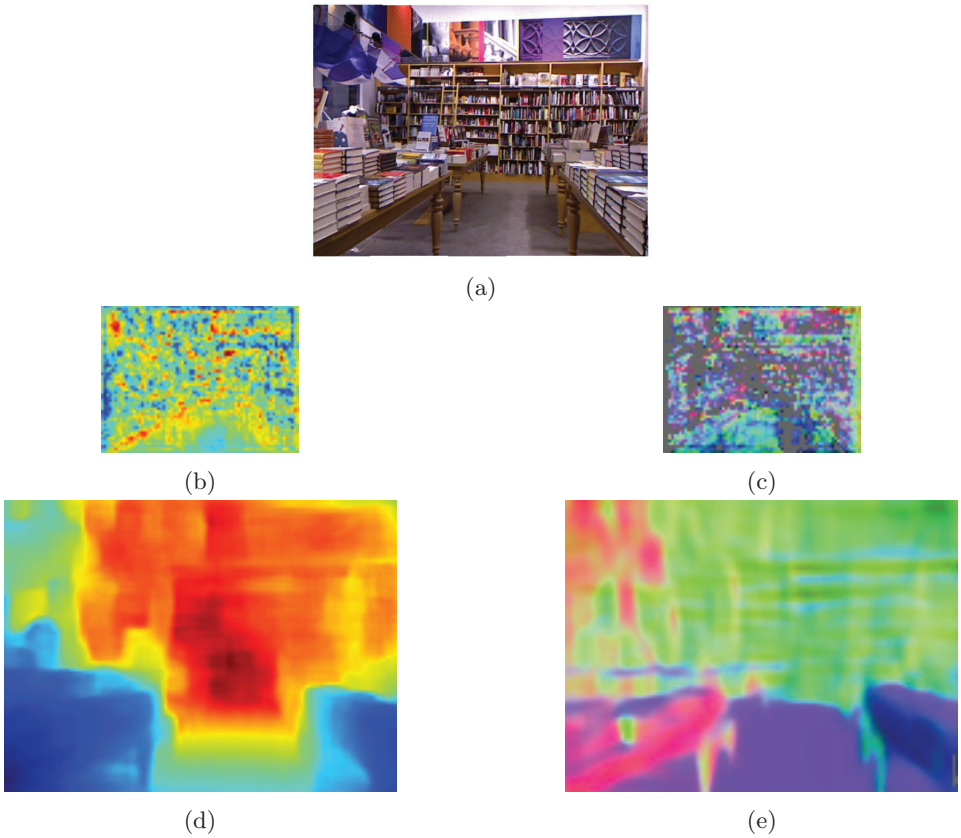
### 3.2.1 Scale 1

The first scale is more unique than other two scales, it has fully connected layers and only this scale has fully connected layers. This scale includes an pre-trained **imagenet VGG** network as an pre-processing process. After series of `Conv`, `Pooling`, `Dense` layers, it is passed to the final two fully connected layers. The first one has 4096 features, the second one has 17024 ($64 \times 14 \times 19$) features. Both these two fully connected layers use 0.5 dropout techniques to prevent overfitting. The output of the fist layer has only the resolution of 266. To merge with scale two, the dense features are firstly reshaped to $14 \times 19$ then upsampled to $64 \times 55 \times 74$.

### 3.2.2 Scale 2

After the first scale, there is no fully connected layers anymore. As a difference, the network also breaks into two paths for normal map and depth map.

To be able to concatenate with last layer, the input RGB images are convolved with a 9 convolution layer with depth of 64 at first. It uses padding to then produces same size of $64 \times 55 \times 74$ feature maps. Together with scale one output, it results of $128 \times 55 \times 74$ tensors. There are 5 convolution layer afterwards, with smaller kernel size $5x5$ and same padding technique. At the last convolution layer, it uses depth of $C$ (3 for normal and 1 for depth) of convolution layer to produce low resolution depth map and normal map.



(a) (b) (c) (d) (e)

Figure 16: Output of the neural network: the image of the top row is the orignal RGB image; the two images on the second row are the low resolution depth and normal map; the last row are the finer resolution depth and normal map. Note that depth and normal map are post-processed with montage technique to be more colourful

### 3.2.3 Scale 3

The last scale involves similar process as scale two, concatenation and convolution. There one major differences: in the concatenation part, after convolves the RGB image with depth of 64 kernel, the concatenation applies. Since The last scale output is only at depth of $C$, it is significantly smaller

than current scale, which outputs $64 + C$ depth of feature map. After a series of convolution, the end of scale 3 outputs an $C \times 109 \times 147$ features.

In summary, the network learns to predict coarse depth maps and then refine the prediction through multiple scales. At each scale there is a number of convolution layers and pooling layers. A crucial aspect of this network architecture is the concatenation of the output of previous scale to the input of the following scale in order to add additional depth of channels for convolution.

Here we are showing the example output of the neural network for the second scale and last scale. In the Figure 16 show the output of this process, namely the depth and normals respectively.

## 3.3 Virtual Camera Projection for rendering

This section is the full deduction of the virtual camera projection for depth image based rendering. We deduced the camera rendering equation for a parallel camera setup shown in the equation 24 later in the section, and also a non-parallel camera setup rendering equation. The equation simplify rendering process.

Cameras project 3D objects onto an image plane. Given extrinsic matrix $E$ and intrinsic matrix $C$, one can project a 3D point $P = [X, Y, Z]$ to a 2D image point $u = [x, y]$ using the dot product of the matrices:

$$\lambda u = C \times E \times P \tag{14}$$

The projection assumes no distortions with the camera. and the camera matrix looks like

$$C = \begin{bmatrix} f & 0 & a \\ 0 & f & b \\ 0 & 0 & 1 \end{bmatrix} \tag{15}$$

Where $f$ is the focal length, $a$, $b$ is the principle point offset. Extrinsic matrix $E$ is

$$E = R \times -t = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{32} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R & -Rt \\ 0 & 1 \end{bmatrix} \tag{16}$$

Where $R$ is the rotation matrix. Based on equation 14, we can also associate two projections $u_1 = [x_1, y_1]$ to $u_2 = [x_2, y_2]$ for the same 3D point $P = [X, Y, Z]$ with

$$\lambda_2 u_2 = C \times R' \times T \times R^{-1} \times C^{-1} \times \lambda_1 u_1 \tag{17}$$

$R'$ is the new camera rotation, $T$ is the translation from the original camera to a new camera. If we assume the new camera has the same rotation(which is the case), we could optimize the equation $R \times T \times R^{-1}$

$$R \times T \times R^{-1} = \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} I & -t \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} R^{-1} & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} I & -Rt \\ 0 & 1 \end{bmatrix} \tag{18}$$

To be able to get to the end, we still need to tear some of the matrices apart, for the convenience, equation 17 gets written as

$$\lambda_2 u_2 = \begin{bmatrix} f & 0 & a \\ 0 & f & b \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} I & -Rt \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} \frac{1}{f} & 0 & -\frac{a}{f} \\ 0 & \frac{1}{f} & -\frac{b}{f} \\ 0 & 0 & 1 \end{bmatrix} \times \lambda_1 u_1 \tag{19}$$

At this point, we should know that lambdas are the image depth values. By merging the right hand side, we get

$$\lambda_2 u_2 = \begin{bmatrix} f & 0 & a \\ 0 & f & b \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \lambda_1 \frac{x_1 - a}{f} - Rt^x \\ \lambda_1 \frac{y_1 - b}{f} - Rt^y \\ \lambda_1 - Rt^z \end{bmatrix} = \begin{bmatrix} \lambda_1(x_1 - a) - fRt^x + a \cdot (\lambda_1 - Rt_z) \\ \lambda_1(y_1 - b) - fRt^y + b \cdot (\lambda_1 - Rt_z) \\ \lambda_1 - Rt_z \end{bmatrix} \tag{20}$$

There are a few things worth pointing out here, the new image depth transferred from $\lambda$ to $\lambda + Rt_z$. Finding the $Rt$ becomes a crucial problem. But actually, there could be even more simplification, by the time we derived $R \times T \times R^{-1}$ at equation 18. We ignored a vital fact, the camera pose translation is relative the its current rotation. More precisely, it translates at the direction of $x$ axis of the rotation coordinate system. That is $R$ can be rewrite as basis

$$R = \begin{bmatrix} \vec{x}^t \\ \vec{y}^t \\ \vec{z}^t \end{bmatrix} \tag{21}$$

Since the coordinate axes have to be orthogonal to each other, then we have

$$Rt = \begin{bmatrix} \vec{Rx}^t \\ \vec{Ry}^t \\ \vec{Rz}^t \end{bmatrix} \times c\vec{x} = [c, 0, 0]^t = \vec{c} \tag{22}$$

Plugs the result to equation 20, we have

$$\lambda_2 u_2 = \begin{bmatrix} \lambda_1(x_1 - a) - fc + a \cdot (\lambda_1 - 0) \\ \lambda_1(y_1 - b) - f \cdot 0 + b \cdot (\lambda_1 - 0) \\ \lambda_1 + 0 \end{bmatrix} \tag{23}$$

Which implies

$$u_2 = \begin{bmatrix} x_1 - \frac{fc}{\lambda_1} \\ y_1 \\ 1 \end{bmatrix} \tag{24}$$

The equation 24 makes sense because the epipolar line is horizontal when the camera motion is horizontal as well. The final result is powerful, as we eliminated the camera calibrations. Figures 17 18 19 are a sample rendering based on the equation 24.

Figure 17: Original RGB image from Depth-Transfer Dataset

Figure 18: Depth map captured from Kinect

Figure 19: A rendered left-view of depth map, note that there is a small disparity between this left view and original view

There is an additional improvement on equation 19. In the equation, we took the parallel camera setting, but usual human eyes do not look in that way, we could add additional rotation matrix to tweak a bit. Here we remove the stew offset to simplify the process.

$$\lambda_2 u_2 = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} R_y & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -\vec{c} \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} \frac{1}{f} & 0 & 0 \\ 0 & \frac{1}{f} & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \lambda_1 u_1 \tag{25}$$

$R_y$ is a rotation matrix for $y$ axis. It goes the form of

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & cos\theta \end{bmatrix} \tag{26}$$

With a little more of notations, we also simplify the $inv(C) \times \lambda_1 u_1$ to $\hat{u}_1$.

$$\hat{u}_1 = \begin{bmatrix} \frac{\lambda_1 x}{f} \\ \frac{\lambda_1 y}{f} \\ \lambda_1 \end{bmatrix} \tag{27}$$

Now the equation 25 becomes:

$$\lambda_2 u_2 = C \times \begin{bmatrix} R_y & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -\vec{c} \\ 0 & 1 \end{bmatrix} \times \hat{u}_1 \tag{28}$$

It leads to

$$\lambda_2 u_2 = C \times (R_y \hat{u}_1 - R_y \vec{c}) \tag{29}$$

Because $\theta$ can be sufficiently small(less than ten degree), we could replace $\cos\theta$ with 1, $\sin\theta$ with $\theta$. After Combining equation 26 and 29 we could get

$$\lambda_2 u_2 = C \times R_y \times (\hat{u}_1 - \vec{c}) \approx \begin{bmatrix} f & 0 & f\theta \\ 0 & f & 0 \\ -\theta & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \frac{\lambda_1 x - fc}{f} \\ \frac{\lambda_1 y}{f} \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} \lambda_1 x - fc + \lambda_1 f\theta \\ \lambda_1 y \\ \lambda_1 - \frac{(\lambda_1 x - fc)\theta}{f} \end{bmatrix} \tag{30}$$

Equation 30 may looks difficult, as we don't want to divide everything by $\lambda_1 - \frac{(\lambda_1 x - fc)\theta}{f}$. But note that we could ignore $\frac{(\lambda_1 x - fc)\theta}{f}$, there are two reason for this:

- $\frac{(\lambda_1 x - fc)\theta}{f}$ is very small.
- There should be no change to y-axis. We only listed one rendering equation here, but actually we should render two symmetric view(left and right camera), then depth of scene should be symmetric as well.

In fact, if we remove the translation, we should find out that by equation 31 The x-axis offset only changes by $f\theta$.

$$\lambda_2 u_2 = C \times R_y \times (\hat{u}_1) \approx \begin{bmatrix} f & 0 & f\theta \\ 0 & f & 0 \\ -\theta & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \frac{\lambda_1 x}{f} \\ \frac{\lambda_1 y}{f} \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} \lambda_1 x + \lambda_1 f\theta \\ \lambda_1 y \\ \lambda_1 - \frac{\lambda_1 x \theta}{f} \end{bmatrix} \approx \begin{bmatrix} x + f\theta \\ y \\ 1 \end{bmatrix} \tag{31}$$

Now we are ready to write down the entire simplified rendering equation.

$$u_2 = \begin{bmatrix} x - \frac{fc}{\lambda} + f\theta \\ y \\ 1 \end{bmatrix} \tag{32}$$

For left camera, $c$ is negative, thus $-fc$ is positive. But angle is negative (because camera rotates at positive angle), the right camera view is the reverse.

## 3.4 Stereo View Rendering

Equation 24 and 32 provide a convenient method of deriving the depth for the second camera given the focal length $f$ and the camera offset $c$. This produces a depth map for the second camera which can be used to generate the anaglyph stereoscopic image. An important aspect of this process is that the drastically increasing or decreasing the focal length and camera offset results in small or large disparities between the depth maps which when used to generate a stereo image, the image causes discomfort. This is demonstrated in Figure 26 at section 4.2 where the focal length and camera offset were intentionally large. At the same time, here raises the problem of choosing focal lens $f$ and cameras offset $c$. From the formula 24 we could not determine well the sound choice of $c$ and $f$, the only limit that we can apply here is

$$fc \geq \min(\max Disparity) \cdot \min(depth)$$

The formula 3.4 points out the camera offset and focal lens should be determined by the biggest pixel disparity offset, thus, this value should also vary by the resolution of the image. By our experiment, 10 pixel max disparity provides a good rendering results for a typical image at size $500 \times 500$.

We need extra information to determine the correct value, therefore, we made extra assumptions utilize the formula 32. The process can be visualized by figure 20. In the image, we added two extra angles $\alpha_1$ and $\alpha_2$ in the system, they represents the "focus angle" for far plane and near plane. Followed our rendering assumption, $\alpha_1$ and $\alpha_2$ are intentionally chose very small. In addition to these two angles, we also need $D$ to determine the $C$. As the depth map is grayscale images that range from 0 to 255, a clear separation of foreground and background depends on disparity separation of foreground and background, also, the separation also depends on image resolution. So we took a "rectangle scene" assumption, which means the depth range is the width image. By this approach, the rendering setup is determined.

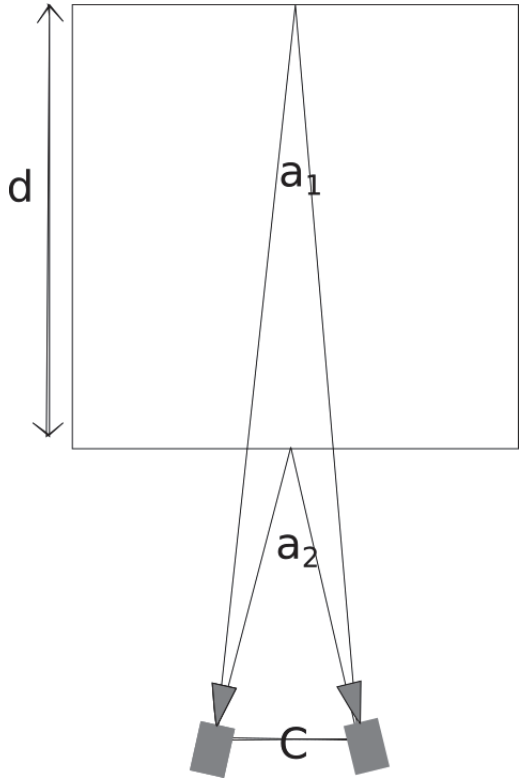



Figure 20: Camera Setup illustration

In addition to using appropriate values for the focal length and camera offset, one has to address issues arising from the rendering. For example, depth buffering needs to be enabled in order to avoid overwriting pixel values. Another problem that arises is the presence of holes [or cracks] in the final rendering. This can be overcome by decreasing the focal length and camera offset, however this may lead to decrease in the disparities which in turn leads to the aforementioned problem with discomfort. This problem has been also reported by others such as in [ZDdW10] and solutions have

been proposed such as oversampling the image and enlarging the warp beam. Another possible solution to this problem is in-painting where the values of neighbouring pixels are used to fill in the missing values. In our work we use in-painting and in particular Navier-Stokes based inpainting method [BBS01].

# Chapter 4

# Implementation and Results

This chapter describes the implementation details and evaluations.

## 4.1 Implementation

The system is implemented with careful, extensible design in mind, system is constructed by components and every component itself is a test case. That is the development style starts by

1. Developing standard test case, inheriting main code-base and building out-of-tree classes and implementations.

2. merge into the main code base if the test case is successful. Thus it can be inherited by later on by other test cases.

Each of the test case takes input output parameters, such component style implementation requires writing script to "glue" each of them together but also gave us the flexibility to plug in other out-of-tree components to the work without rewriting new interfaces.

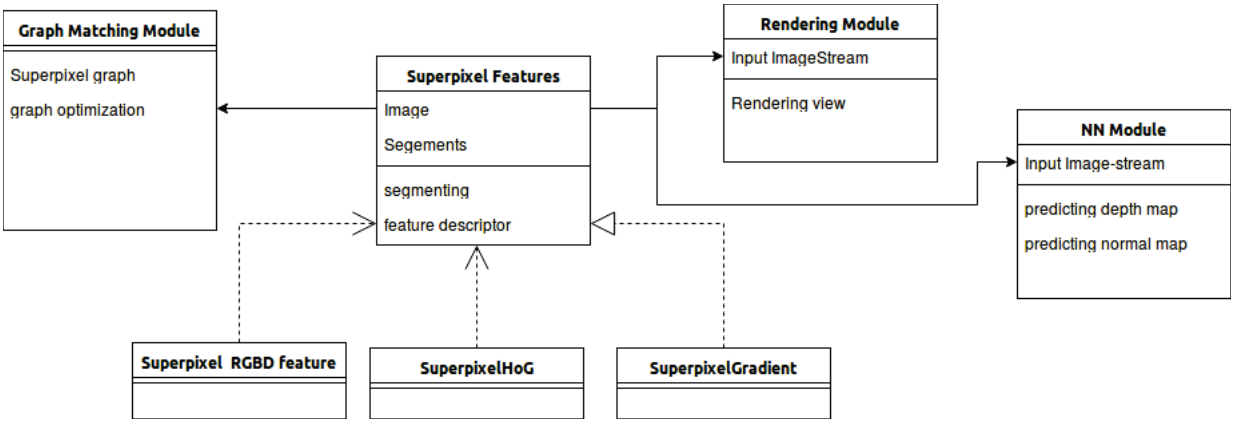



Figure 21: Components of the projects

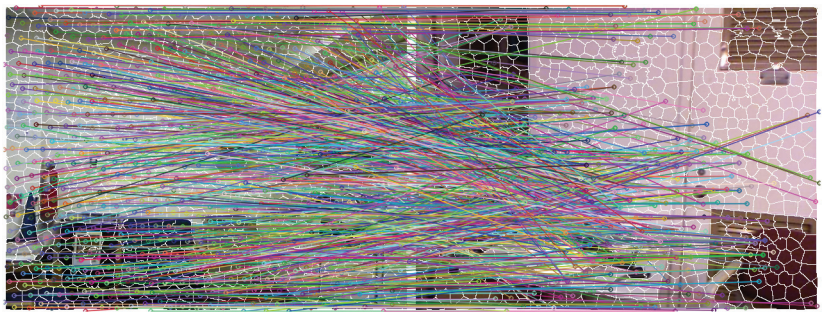
The project mainly uses **OpenCV** library. As shown in figure 21, the project is divided into 4 components. `Superpixel*` class is the most heavily inherited class. The base class does many jobs. For example, making superpixel segments; drawing contours for superpixels; removing small superpixels; building superpixel graph for the image and many debugging methods. As the base class is lack of descriptor generation, sub-classes are made with many different implementations. The current one we are using is `superpixelGradient` implementation, with implements the SIFT descriptor for Superpixels. Other implementations like `SuperpixelDRWN` [GZHZ14] uses many filtering (texton, LBP) results for the superpixels. `GraphMatch` class take two superpixel instances and use the algorithm we described in the section 3.1.3. As graph data structure is hard to manage, we used `Boost`'s `johnson_all_pairs_shortest_paths` function for computing the distance for each nodes in the graph. Rendering class is implemented at last, like `NN` module, it is more independent. Benefited from the rendering formula we derived. This module can run at real time. `NN` module is a wrapper around the implementation by [EPF14]. It does the image feeding and output extraction.

## 4.2 Results

The section includes the project results as well the corresponded evaluation.

Figure 22 is an example illustration of the results for the matching algorithm which we described in section 3.1.3.

(a) the left is the image to match. The right is the one we match against, it is the closest image retrieved by GIST



(b) Graph Matching results

Figure 22: The graph matching algorithm

Figure 23 is an instance of our depth transferring. The matching algorithm does smooth the

transferred depth. However, the result is not satisfying enough, so we changed to the CNN approach.

Figure 23: Single image Depth Transfer result(without smoothing), the right two grey scale images represent the depth

Figure 24 to figure 28 (next page) are five row images are the main results of the project. The represent the different rendering stages. We used two different rendering equation to archive different stereo effects. First one is the parallel stereo setup, the second one has angle between two camera.

More results are available on YouTube at `https://www.youtube.com/user/xeechou/videos?view=0&shelf_id=0&sort=dd`.

## 4.3 Evaluation

In order to evaluate whether our rendering results quantitatively, we used our rendering results as an input to the depth prediction pipeline. Intuitively, if the rendering results is accurate, it should produce similar depth maps, since they are only from slightly different views. Table 1 illustrates the accuracy of our rendering results. The metric here is the relative depth (range in $[0-1]$). We compared three different metrics, absolute difference; square difference and fractional depth difference $(d-\hat{d})/d$. The evaluation shows the mean fractional difference is less than 0.02, max difference is less than 0.08, which yields stable results.

| evaluation type | mean | max | min |
|---|---|---|---|
| absolute depth error | 0.0102313 | 0.0469497 | 0.00246998 |
| square depth error | 0.000275407 | 0.00255624 | $1.96123e-05$ |
| fractional depth error | 0.0184376 | 0.0865189 | 0.00501813 |

Table 1: Evaluation, per-pixel error for re-generated depth map

Figure 24: The original sample image

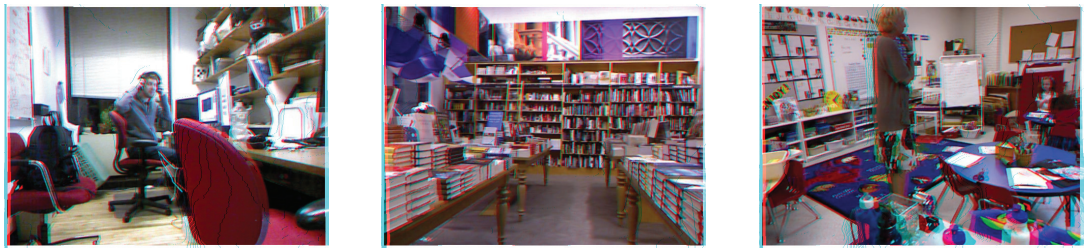


Figure 25: rendered anaglyph without in-painting

Figure 26: Rendered anaglyph with too large focal lens settings that causes discomfort

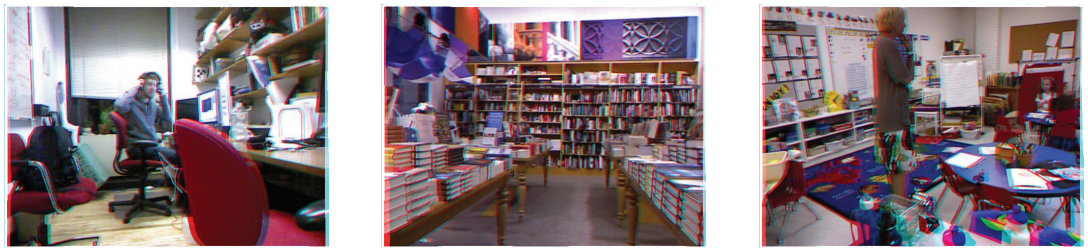


Figure 27: rendered anaglyph after in-painting

Figure 28: rendered anaglyph using new rendering formula

# Chapter 5

# Conclusion

We have presented a novel method for automatic conversion of 2D images/videos to 3D. We used two methods, one uses non parametric method, the second one leverages the strengths of Deep Learning to address the complex problem of depth estimation from a single image. The Convolutional Neural Network produces a depth map which is then used to render the anaglyph image. We use anaglyph images due to wide availability or red/cyan glasses however this does not limit our approach from being extended to other forms of stereo e.g. 3D TVs. Furthermore, we have presented a simplified formulation for computing the depth map of the second stereo camera given two parameters. The method has been tested with several videos and in the future we anticipate to evaluate the effectiveness of the approach with human participants.

## 5.1 Future Works

Our anaglyph rendering method is based on depth map, but both of our proposed methods has limitations. The first one is more limited than the other because we may not have a similar image to match. Aside from that we believe it has potential in producing better results. One insight is that we will transfer normal map instead of depth map directly, because normal map is indeed a more "local" feature than depth map. We hope also to change the optimizing technique to algorithms like **Graph Cut** [BVZ01], as it has proved its versatility in many applications.

The Neural network approach also has limitation, the first one is the output resolution, the current state-of-the-art CNN network can only deal with only resolution images. Scaling the resolution up is a big issue. The second one is size of the models, the model we are using can only work with specific size ratio. Relaxing such limit can be a interesting direction.

# Bibliography


[ASS+12a] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[ASS+12b] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2274–2282, November 2012.

[BBS01] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–355–I–362 vol.1, 2001.

[BETVG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[BSGF10] Connelly Barnes, Eli Shechtman, Dan Goldman, and Adam Finkelstein. The generalized patchmatch correspondence algorithm. *Computer Vision–ECCV 2010*, pages 29–43, 2010.

[BVZ01] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

[EPF14] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[Feh04] Christoph Fehn. Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3d-tv. In *Electronic Imaging 2004*, pages 93–104. International Society for Optics and Photonics, 2004.

[GZHZ14] Stephen Gould, Jiecheng Zhao, Xuming He, and Yuhang Zhang. Superpixel graph label transfer with learned distance metric. In *European Conference on Computer Vision*, pages 632–647. Springer, 2014.

[HIP+16] Hyowon Ha, Sunghoon Im, Jaesik Park, Hae-Gon Jeon, and In So Kweon. High-quality depth from uncalibrated small motion clip. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5413–5421, 2016.

[HWB12] Xiaoxia Huang, Ian Walker, and Stan Birchfield. Occlusion-aware reconstruction and manipulation of 3d articulated objects. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1365–1371. IEEE, 2012.

[KB15] Naejin Kong and Michael J Black. Intrinsic depth: Improving depth transfer with intrinsic images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3514–3522, 2015.

[KLK12a] Kevin Karsch, Ce Liu, and Sing Kang. Depth extraction from video using non-parametric sampling. *Computer Vision–ECCV 2012*, pages 775–788, 2012.

[KLK12b] Kevin Karsch, Ce Liu, and Sing Bing Kang. Depth extraction from video using non-parametric sampling. In *ECCV*, 2012.

[KTS11] Christian Kurz, Thorsten Thormählen, and Hans-Peter Seidel. Bundle adjustment for stereoscopic 3d. In *International Conference on Computer Vision/Computer Graphics Collaboration Techniques and Applications*, pages 1–12. Springer, 2011.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LKR+17] Thomas Leimkühler, Petr Kellnhofer, Tobias Ritschel, Karol Myszkowski, and Hans-Peter Seidel. Perceptual real-time 2d-to-3d conversion using cue fusion. *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LRKL13] Zhihan Lu, Shafiq Ur Rehman, Muhammad Sikandar Lal Khan, and Haibo Li. Anaglyph 3d stereoscopic visualization of 2d video based on fundamental matrix. In *Virtual reality and visualization (ICVRV), 2013 international conference on*, pages 305–308. IEEE, 2013.

[LSH14] Miaomiao Liu, Mathieu Salzmann, and Xuming He. Discrete-continuous depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723, 2014.

[LYT11] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):978–994, 2011.

[OT01] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vision*, 42(3):145–175, May 2001.

[pag] Rendering 3d anaglyph in opengl. http://www.animesh.me/2011/05/rendering-3d-anaglyph-in-opengl.html. Accessed: 2017-06-09.

[RVCK16a] René Ranftl, Vibhav Vineet, Qifeng Chen, and Vladlen Koltun. Dense monocular depth estimation in complex dynamic scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4058–4066, 2016.

[RVCK16b] Rene Ranftl, Vibhav Vineet, Qifeng Chen, and Vladlen Koltun. Dense monocular depth estimation in complex dynamic scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[SSN09a] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):824–840, 2009.

[SSN09b] Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2009.

[TL10] Joseph Tighe and Svetlana Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. *Computer Vision–ECCV 2010*, pages 352–365, 2010.

[TLF10] E. Tola, V. Lepetit, and P. Fua. DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830, May 2010.

[Wel78] W. Welling. *Photography in America: the formative years, 1839-1900.* Crowell, 1978.

[Wik17] Wikipedia. Stereoscopy, 2017. accessed 06-14-2017.

[ZDdW10] Sveta Zinger, Luat Do, and PHN de With. Free-viewpoint depth image based rendering. *Journal of visual communication and image representation*, 21(5):533–541, 2010.

[Zed] ZED Stereo Camera the world's first 3d camera for depth sensing and motion tracking. https://www.stereolabs.com. Accessed: 2017-05-06.