ADAPTIVE 2D TO 3D IMAGE CONVERSION USING A HYBRID GRAPH CUTS AND RANDOM WALKS APPROACH

by

Mohammad Fawaz

Bachelor of Engineering, Ryerson, 2010

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Masters of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

ADAPTIVE 2D TO 3D IMAGE CONVERSION USING A HYBRID GRAPH CUTS AND RANDOM WALKS APPROACH

Masters of Applied Science 2012

Mohammad Fawaz

Electrical and Computer Engineering

Ryerson University

Abstract

This thesis proposes an adaptive method for 2D to 3D conversion of images using a user-aided process based on Graph Cuts and Random Walks. Given user-defined labelling that correspond to a rough estimate of depth, the system produces a depth map which, combined with a 2D image can be used to synthesize a stereoscopic image pair. The work presented here is an extension of work done previously combining the popular Graph Cuts and Random Walks image segmentation algorithms. Specifically, the previous approach has been made adaptive by removing empirically determined constants; as well the quality of the results has been improved. This is achieved by feeding information from the Graph Cuts result into the Random Walks process in two different ways, and using edge and spatial information to adapt various weights.

This thesis also presents a practical application which allows for a user to go through the entire process of 2D to 3D conversion using the method proposed in this work. The application is written using MATLAB, and allows a user to generate and edit depth maps intuitively and also allows a user to synthesize additional views of the image for display on 3D capable devices.

Acknowledgements

I would like to start off by thanking the most important person in helping me complete this thesis and that would be my supervisor, Dr. Dimitrios Androutsos. Through his support as well as guidance I was able to not only get through 2 years of M.A.Sc. studies, but I had a great time the entire way. He respects his students and understands the connection between what we do in the lab and what is out there in the real world, and this is extremely valuable for anyone who plans on working in the industry eventually. Most importantly, Dr. Androutsos provided the best environment to get work done; the lab was full of the right tools for any situation. Overall, working with Dr. Androutsos has been a great privilege.

I would also like to thank my two colleagues and friends, Raymond Phan and Richard Rzeszutek, whom I owe so much for the help they've given me throughout my two years of studies. Together they provided the code implementations of the Graph Cuts and Random Walks algorithms used in my thesis, as well as many other parts of the code which would have taken me forever to write out myself. More importantly, they provided advice and support when I was having difficulty forming my thesis as well as throughout when things just didn't make sense. Also, I don't think I could have made it to ICASSP2012 without both their help, Raymond for the hours of work he put into editing and transcribing my paper into LaTeX and Richard for getting me started with an idea and all the useful code which made it easy for me to generate results. I wish them both the best of luck moving forward.

Finally, I would like to thank my family, friends and girlfriend who all helped me get through the tough times with support and encouragement. It was initially my dad who pushed me to do an M.A.Sc. and now that it's done, I think I can proudly say I am glad I listened to my parents' advice this time.

Contents

List of Tables

List of Figures

List of Appendices

Chapter 1

Introduction & Background

DEPTH perception in multimedia is something that has seen a great rise in interest in recent years. 3DTVs have made their way into the living rooms of families, 3D cinema releases are more frequent and the quality is leaps and bounds greater than what was seen before. The sudden boom can be seen as a resurrection of a past technology which, due to more recent advancements in both display technologies and content generation, can now be enjoyed by more people and with a higher level of quality. Specifically, the type of 3D which is the focus of this thesis and is used in describing today's 3DTVs or movies is known as *Stereoscopic 3D* and the process through which depth is perceived using stereoscopic images is known as Stereoscopy.

Stereoscopic 3D is the illusion of 3D depth created by using a pair of left and right images, henceforth referred to as a stereoscopic pair, and making sure each image is only seen by the left and right eyes respectively. The brain is accustomed to creating depth in the world around us from what is seen with both eyes, because each eye does not see the same thing as the other and so the disparity between the views in each eye is one of the ways the brain recognizes depth. This particular depth cue is referred to as parallax; Figure 1.1 below illustrates what is meant by parallax.

In Figure 1.1, point P lies on the stereo plane and corresponds to objects which will appear flat on a screen when viewed. Negative parallax would be for objects which appear in front of the screen and

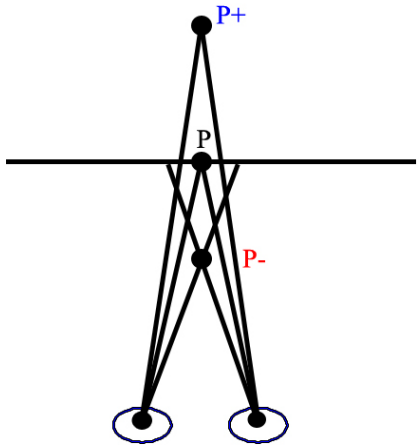


Figure 1.1: Anaglyph Rendering of an Image from the Superman Cartoon

positive parallax is for objects further away from the screen. By manipulating the separation between pixels in left and right images the different types of parallax can be achieved.

1.1 3D Display Technology

When discussing stereoscopic 3D, it can be noted that research is generally broken up into 2 categories. On one side, there is research which concerns 3D display technologies, or in other words the means through which people can view stereoscopic 3D. The other avenue is research covering content generation and distribution.

Most of the common content display techniques involve the use of specialized glasses such as the previous generation of red and blue anaglyph lenses. More recently, polarized lenses have been used and furthermore there are even methods which do not require the use of lenses at all and are referred to as auto-stereoscopic display methods.

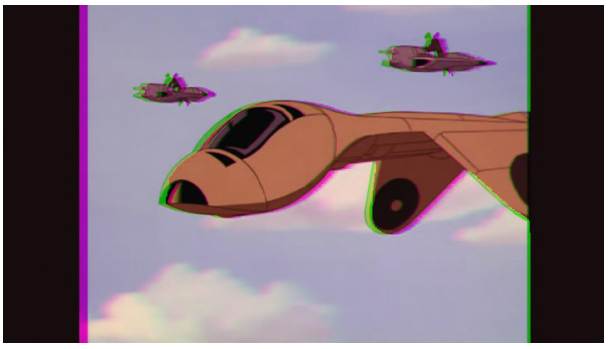
1.2 3D Content Generation

3D content generation is the other large area of research today concerning depth perception. Advancements in display technology are what triggered the recent boom in 3D, but it is the content which holds the interest of those who use the technology. Content generation can be broken down into 3 main areas: 3D content capture using specialized hardware, 3D rendering using computer graphics software and finally 3D content creation from regular 2D content.

1.2.1 3D Capture

The most direct way to create stereoscopic 3D content is to capture the scene stereoscopically. This is done through the use of dual cameras. The centres of each cameras aperture should be approximately the same distance apart as the average human eyes. The cameras also need to be along the same horizontal axis, remaining level at all times. Once the scene has been captured using such a method display can be left up to one of the previously discussed methods.

(a) Anaglyph Lenses Used to View Anaglyph Images



(b) Anaglyph Rendering of an Image from the Superman Cartoon

Figure 1.2: Anaglyph Technology

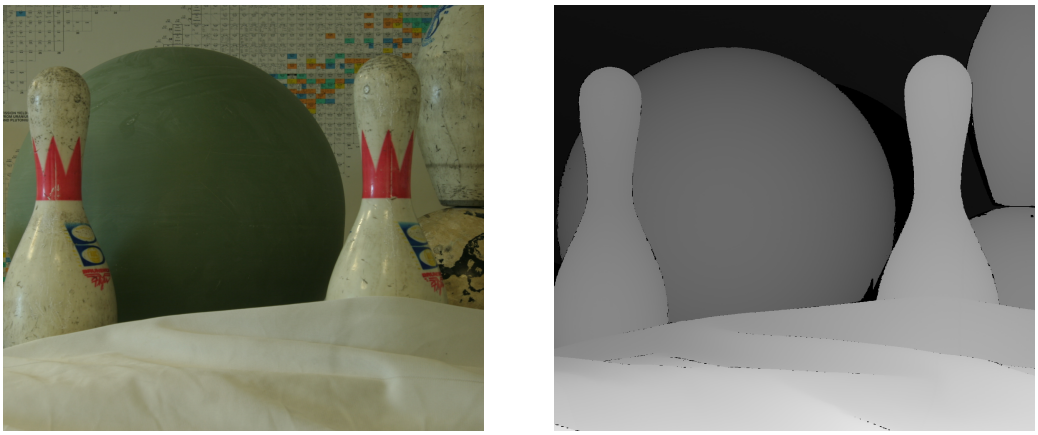
1.2.2 3D Render

The next technique is what is being used to transform some older 2D media to 3D and applies to most 2D content which is composed primarily of CGI or computer generated imagery. Normally, when generating models in a computer rendering environment, an artist picks a position for a virtual camera in the scene. This affects how the scene is viewed and the lighting and various other details. In order to create left and right views of the scene, the artists need to add another camera to the scene, then the two cameras will act in the same manner as if two real cameras existed. From there two scenes are rendered, one for the left view and one for the right.

1.2.3 2D to 3D Conversion

The last method of 3D content generation is through conversion of existing 2D content into 3D, where only one view exists and the goal is to create the second view using information in the existing view. This is not a trivial process and a lot of research has gone into different methods for doing this including the work done in this thesis. The most common form of 2D to 3D conversion involves generating what is known as a depth map and thus generating the second view using DIBR, which is described later. A depth map is a grey scale image which indicates the depth of a pixel in an image. Generally, pixels which are nearer are assigned high values on the grey scale while objects that are further away are given smaller values.

Similar to a depth map is a disparity map, which maps pixels between left and right images according to the distance between them which is proportional to the real life distance as seen through both eyes. There is a relationship between disparity and depth, whereby pixels that are closer to the foreground will have a larger disparity between them and pixels which are in the background have a smaller disparity. The approaches for 2D to 3D conversion which look at generating a depth map and then using that to create a second view is called Depth image Based Rendering or DIBR.



(a) Bowling1 Image (b) Corresponding Depth Map

Figure 1.3: Bowling courtesy of Middlebury Stereo database [1]

DIBR DIBR stands for Depth Image Based Rendering and is the process of using a depth map along with an image to generate synthesized views of the scene. This process is not trivial and several problems may arise along the way. One major issue is that of optical occlusions which occur when pixels in one view are hidden or do not exist in the other view. In this case estimations must be done to recreate the data which is occluded. This work does not focus on occlusions and instead uses a simple method for dealing with them. The use of the generated depth map to create a second view can then lead to other problems such as not all pixels in one image mapping to a unique pixel in the other image and so holes become an issue. Filling the holes is something else which needs to be considered.

Automatic depth map generation When generating the depth map the algorithms which exist can be decomposed into two categories: those which find a depth map automatically and those which require some sort of user intervention, usually in the form of pixel labelling. The automatic approaches require little to no user input but generally have several preprocessing and post processing steps. Furthermore, as of today there are no fully automatic methods which do a good enough job of 2D to 3D conversion which can be used for actual conversion of large scale productions. Because of this, focus has shifted

more towards semi-automatic methods such as the work in this thesis.

Semi-automatic depth map generation Semi-automatic 2D to 3D conversion focuses on taking in some sort of user input to aid with the conversion process. There are various ways to accept user input, but the most common is in the form of user provided labels. Labels correspond to depths in an image. The idea is to give the algorithm some hints as to what depth should be assigned to pixels. For this, user provided brush strokes are used as input to the algorithm which then can produce a depth map.

1.3 Background

Before going into detail about the proposed algorithm past works and related studies should be noted. This section focuses on giving an idea of what past and current research exists in the area of 2D to 3D conversion. The list is in no way comprehensive as a focus has been placed on research done in the last 5 years, most significant research involved with 2D to 3D conversion has been published within that time period and coincides with the large gain in popularity of 3D as a medium of entertainment.

This particular work focuses on a semi-automated approach to 2D to 3D conversion, specifically making use of two very well-known segmentation algorithms by Grady[2] and Boykov *et al.*[3]. Semi-automated approaches only cover a small fraction of the spectrum concerning 2D to 3D conversion research. The larger majority of research has gone into automatic methods, and only in recent years has there been a little more interest shown in semi-automatic approaches. As explained previously, semi-automated methods rely on some form of user input to aid with the conversion process; the input is usually in the form of some sort of label applied to an image. Automated approaches however attempt to estimate depth based on other cues, most often motion, edge information, and in some cases focus are used.

1.3.1 Automatic 2D to 3D conversion

Automatic 2D to 3D conversion methods rely on a variety of techniques to find depth cues in a 2D image. Some of the more popular methods are structure from motion, structure from edge information, colour based depth extraction, and depth from geometry. Some methods make use of encoded video as motion information is already embedded into the video stream. The next few sections will go through each method and give a brief summary of related works.

STRUCTURE FROM MOTION The first group of automated methods which also happens to be the largest subset of 2D to 3D conversion methods are those that rely on structure from motion. Structure from motion is the process of extracting 3D information based on the local movement of objects between frames in a video sequence[4].

Knorr *et al.'s* [5] approach uses structure from motion to analyze a frame sequence and using information from subsequent frames to generate a right view directly. No depth map is generated; instead the algorithm takes into consideration the camera properties and the desired disparity requirements of the generated view and then tries to find subsequent frames which match the requirements. They then expand on this idea in [6] by adding in information from camera position and object properties to improve the generated views.

Kim *et al.* [7] use a typical motion-to-depth approach but instead of arbitrarily converting motion to depth values the characteristics of the display environment are considered. Type of display, type of audience, geometric characteristics all go in to the conversion of motion to depth. Kanade-Lucas-Tomasi (KLT) feature tracker is used for motion estimation. DIBR is used to generate views.

In Xu *et al.'s* [8] method they use a mix of structure from motion and depth from color. Optical flow is used to extract motion from a set of frames, meanwhile the image is converted to gray scale and using the Minimum Discrimination Information (MDI) principle, the grayscale and motion information are merged together. MDI attempts to minimize the differences between two images according to some

factors. The minimization is done between the optical flow image, and the greyscale image which has been masked using the optical flow image. The final step of the process is to assign proper depths to each segmented object, and for this a set of rules are used to assign depth values.

Po *et al.* [9] use depth from motion by block based motion estimation which is merged with color segmentation results to produce a depth map. Linear interpolation is used for hole filling.

Chang *et al.* [10] use depth by motion based on block matching between frames to generate a depth prior and Depth from Geometrical Perspective to generate a scene depth. The two results are then merged together using a weighted sum and the final result is filtered using a cross bilateral filter.

In [11] and [4], filtering techniques are used to improve the quality of the depth maps produced using structure from motion. In [4] an Asymmetric Edge Adaptive Filter (AEAF) is used to post process an initial depth map generated using depth-from-motion techniques and H.264 compressed video. The same AEAF is used for hole filling during the DIBR process.

MOTION FROM ENCODED VIDEO Methods which use motion from encoded video attempt to extract the motion information from the compressed domain. Generally, compression standards such as H.264 and MPEG store keyframes along with information about motion in a video sequence for the purpose of reducing bandwidth. The motion information can be used to retrieve structure from motion. Essentially these methods are similar to the previous section however the motion estimation step has already been done.

Su *et al* [12] use a system whereby they first separate moving objects from the background of a video sequence. To do this the area with motion vectors close to 0 are deemed background and other areas are moving objects. The background is matched with a gradient depth using vanishing point and vanishing line as well as edge detection. The object is then given a depth based on its location on top of the background. Finally they implemented their design on a multicore processor and found a large speedup when compared to using a typical x86 processor.

In [13] the conversion is for MPEG videos and is based on various depth cues including motion parallax, atmospheric perspective, texture gradient, linear perspective and relative height. Videos are temporally segmented using shot detection methods and then classified as one of 3 types. This makes the method more adaptive to a wider range of videos.

[14], [15] and [16] use encoded information from the H.264 coding standard. In [14] the authors convert 2D to 3D using motion information found in the H.264 coded bit stream of a video. First a depth estimate is obtained block by block and then the estimate is up-sampled using a joint bilateral filter to get the final depth map.

EDGE INFORMATION AND OTHER COLOUR CUES The next group of methods relies on edge information and other colour cues in order to segment objects in a scene. Edge information is generally used to find complexity in objects, while colour information is used for segmentation purposes as well as a depth cue on its own as is seen in [17].

Chang *et al.* [18] use edge information along with a depth gradient hypothesis to create a depth map. The depth map is then filtered using a cross bilateral filter in order to smooth the edge boundaries of the depth map. DIBR is used to generate synthesized views for stereoscopic display.

Similar to this work Zhang et al [19] use 2 cues for depth map generation, and fuse them together to produce a final result. The first cue, named the dark-channel prior, indicates distance from objects to the camera due to variances in atmospheric light. The second cue is based on object motion. The dark channel prior and motion priors both produce holes. To remedy this, morphological opening and closing operations are used.

Chang et al [20] use edge detection and K-means segmentation to find the depth map of an image. The edge detection and registration is used to find motion information in the scene. This information is fused with the results of the K-means segmentation and over-segmentation using various boundary conditions and thresholds.

GEOMETRY AND SCENE MATCHING A significant body of research is focused on geometric analysis. Most notably a lot of the work attempts to assign depth priors to certain areas of an image based on geometric cues. For example, a wall vanishing into the background would have a gradient depth prior assigned to it; the task of the algorithm would then be to determine that the wall was of that particular class of object.

In [21] the authors introduce the idea of occlusion analysis in order to fill in depth information. First a depth model is found based on the geometry of the scene. Then a saliency map is generated which is used along with the occlusion analysis to supplement the depth model.

Jung *et al.* [22] look at generating a depth map by classifying objects as a certain type using Bayesian learning. Each object has its corresponding depth model and combined they form a depth map.

[23], [24] and [25] make use of depth priors. In the case of [23] the authors focus on conversion of sports content. Based on the type of sport, ground planes, player sizes and background a depth prior is selected in order to generate depth maps. Objects are segmented and assigned depths based on the prior information.

Deng *et al.s* [26] work proposes a 2D to 3D conversion method which makes use of so called planar extraction which segments the background of an image in to 1 or 2 horizontal planes and 1 or more vertical planes representing walls, ground, sky, doors, etc The extraction of foreground from background is done using previously proposed methods. The background and foreground depths are then overlapped.

In [27] an image is broken up into depth layers using a depth ordinal based on occlusion information and region growing. Once the object layers are extracted a depth is assigned using an object matching technique which looks at a library of objects with predefined depth information and then decides on the best match.

DEPTH FROM FOCUS Depth from focus is based on the idea that in certain scenes objects which are in focus are foreground objects while objects out of focus are background. These methods work well

for a small subset of videos and images where this particular shooting technique is employed, but in a full length film this may not always be the case.

[28] and [29] use focus information directly from an image to extract depth. In [28] objects of different focus are extracted using Gaussian filters with various parameters. In [29] occlusions are considered as well.

Both [30] and [31] use information from the wavelet domain to detect focus. In [30] the depth map is generated by analyzing high frequency components of the wavelet domain of an image in YUV color space. High frequency components represents focused objects and therefore could be of interest in terms of locating foreground objects. The depth map which is generated is binary.

OTHER METHODS The rest of the research that goes into automated 2D to 3D conversion does not necessarily fall into any of the other categories but cover a mix of topics.

In [32] a low cost embedded auto-stereo game conversion system is presented. Depth is generated by extracting hidden info which is used to generate the original scene. Software and hardware components are used to speed up processing.

Lie *et al.* [33] propose a method for which the depth map of a key frame can be propagated to other frames by using motion compensation and a trilateral filter along with a feedback loop to refine the results. The original depth map is assumed to have been generated by hand but the technique could work hand in hand with other methods which generate depth maps automatically/semi-automatically.

Wu *et al.* [34] use bidirectional optical flow to extract corner points for tracking and then using the Mean Shift Algorithm objects are segmented. Depth values are assigned by generating a bounding rectangle around each object and then comparing it with the bounding box of the same object in a previous frame. Depending on the changes in width and length the depth value can be calculated.

In [35] SIFT is used to locate features in the image which are then tracked across consecutive frames using a bidirectional feature matching method in order to determine the number of motion layers in the

scene. The image is then over segmented and each block is assigned to a motion layer using Graph Cuts.

1.3.2 Semi-Automatic 2D to 3D Conversion

The other subset of 2D to 3D conversion methods is called Semi-Automatic conversion, where the semi-automatic part comes from the fact that some user interaction is involved in the conversion process. This interaction can be in the form of output correction, prior information or anything else where a user would be called upon to provide further information to the algorithm. This subset while being smaller has recently seen a rise in interest as researchers realized there is no fully automatic methods which can do a good enough job of conversion to be used on large scale productions. Semi-automatic methods sacrifice real-time conversion for high quality results which when compared to the already existing methods of conversion is still a very good compromise.

Take for example a recently re-released movie, The Titanic, the conversion process for this movie took over a year and a workforce of over 450 people[36]. This ends up making the process too costly for any smaller film production companies to even consider. Therefore, compared to this painstakingly long process, semi-automatic methods are very much desirable.

Some past work in the area of semi-automatic 2D to 3D conversion include B.Ward *et al.s*[37] Depth Director, which uses typical structure from motion and Graph Cuts mixed with over-segmentation to generate an initial depth map. Afterwards, user interaction is allowed for modifying the depth regions and/or applying depth templates. There are other methods which use Graph Cuts for user-aided segmentation such as [38], [39] and [40]. In [39] the authors take the conversion to the mobile space and use over-segmentation to speed up the process.

The idea of using user provided strokes is also something found in most user-aided conversion techniques. User labelling such as in [41] provide an intuitive way for users to interact with the images; labels simply correspond to objects which users see and wish to segment. The labeling itself can also be used to assign the depth value to the segmentation, thereby serving a dual purpose such as in [42], [43]

and subsequently the work in this thesis which builds on these two methods.

1.4 Motivation and Goals

This thesis was motivated by a paper prepared for the IEEE International Conference on Image Processing (ICIP) 2011[42]. The idea behind the initial work was to find a new method of semi automated 2D to 3D conversion which could make use of both Graph Cuts for segmentation as well as add in Random Walks. A number of papers which mention the use of Graph Cuts for depth map generation had been published by various authors as has been mentioned previously. In some cases Graph Cuts produced the sole depth prior while in others Graph Cuts was used along with other cues to complete a multi-cue fusion approach to the depth map generation problem. None had used Graph Cuts along with Random Walks before, and due to the similarities between both algorithms as well as ease of implementation it was decided that investigation into a method of 2D to 3D conversion which fused the properties of both algorithms was worthwhile. The results in the paper proved that Graph Cuts and Random Walks can in fact be fused together for the purpose of depth map generation. The decision to focus on semi-automated methods was also a simple one, currently 2D to 3D conversion methods which are used in large scale productions are tedious and very time consuming. By providing a user interaction step while maintaining some form of automated execution the hope is to reduce the time required to convert a large film or other form of media, while still producing good quality results.

The goal of this thesis is thus to develop a better way to fuse the two algorithms together than what was done before, which will allow the conversion process to work for a variety of images and hence be adaptive. The main contributions of this thesis are thus:

Graph Cuts Depth Prior Graph Cuts was used to generate a depth map which itself was modified and fed into the Random Walks algorithm as a label. The use of Graph Cuts as a depth prior is not novel however using it in the way described in this thesis is.

Adaptive Edge Weighting A modification to the edge weighting scheme of Random Walks was also presented which takes into account the strength of the edges around objects to adaptively incorporate information from the Graph Cuts depth prior. This was an expansion of the work done by Phan *et al.* [42] which simply averaged the Graph Cuts and Random Walks results.

2D to 3D conversion application An application was created which made use of the work presented in this thesis in order to allow users to convert 2D images to 3D. The application demonstrates the potential of this method for the use in large scale conversion projects of not only images but videos as well.

In Chapter 2 the two segmentation algorithms which were used, Graph Cuts and Random Walks are presented and details about each algorithm are given. Chapter 3 will begin by comparing the advantages and disadvantages of both the algorithms presented and will talk about the methodology behind the fusion of the two algorithms. This chapter will end with results comparing the depth maps produced by the method in this thesis to other methods. Chapter 4 will give details about the implementation of the algorithm as well as demonstrate its use in the 2D to 3D conversion application which was created for this thesis. Chapter 5 will present a summary of the work and any suggested future work which can be done to improve this thesis.

Chapter 2

Graph Cuts and Random Walks

TAKING a segmentation result and turning it into a depth map directly is a very intuitive process which became popular after researchers realized that absolute depth values in a scene are not necessarily as important as relative depth of objects. This makes it so that the depth map generation problem can be reduced to one where segmentation is the goal and the depth assignment can be handled separately as long as the proper order of depth is correct.

Take for example Figure 2.1, the rabbit, fish and spade can exist at whatever depth from the camera, and neither the user nor the algorithm have to know what that absolute depth is. Instead, if the algorithm can assign arbitrary depth values but in such a way as to place, say the rabbit ahead of the fish, and the fish ahead of the spade, then, when the audience views this scene in 3D the 3D effect will still be seen clearly.

Because of this property, the approach in this thesis to depth map generation at its core involves finding the best way to segment objects in a scene and assign them depths. As a semi-automated algorithm, the depth assignment can be left up to the user, however the segmentation must then be done based on this user input. Two very well-known semi-automatic segmentation algorithms exist which

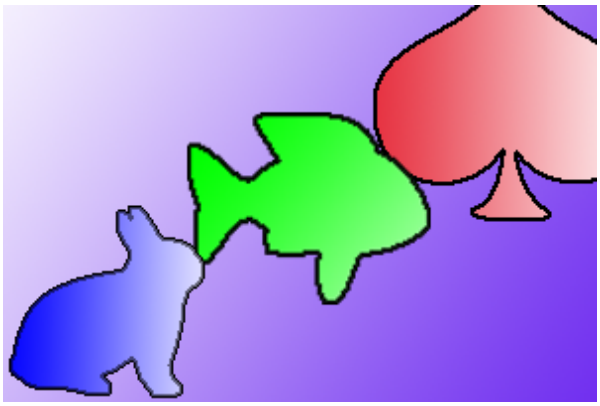


Figure 2.1: Rabbit, Fish and Spade Image

this work uses and in this chapter both algorithms are introduced, as well as show some side by side comparisons of results generated using each one. Ultimately the proposed work here is to take both algorithms and fuse them together in such a way as to incorporate features from both processes.

2.1 Graph Cuts

The first segmentation algorithm which is used in this work was introduced in [3] and was developed by Boykov *et al.*. The application of the combinatorial graph cuts algorithm for segmentation was shown to be a simple yet practically efficient amongst other benefits. Graph Cuts itself is a maximum flow/minimum cut algorithm. What this means is given a graph $G(V, E)$ where V are the set of vertices and E are the set of edges, the algorithm attempts to find the cut in a graph whereby the capacity of the edges being cut is a global minimum. This value also happens to be equal to the maximum flow of the graph. In Graph Cuts, a modification of this algorithm is made such that a source, s, and sink, t, nodes are introduced. The cut which is calculated is then called an *s-t cut* of the graph.

2.1.1 Algorithm

An image is first mapped to a graph $G(V, E)$ as mentioned previously, where the vertices are pixels and the edges are some sort of cost relationship indicating similarity amongst connected pixels. The goal of the Graph Cuts algorithm is to separate the object from the background. In practical situations these labels are either user applied such as in an image editing software, or automatically generated based on a simple object detection algorithm or other automatic scheme. The corresponding graph has edges linking every pixel to each of the source and sink nodes (t-links), as well as edges in between pixels themselves (n-links). When the algorithm is run the goal is to find the cut which slices the minimum cost edges leaving behind a segmentation of object from background.

When calculating the solution to the Graph Cuts problem the energy of the cut must be examined. Essentially, the goal of Graph Cuts is to minimize an energy function, where the foreground and background labelling that produces the least amount of energy is the ideal labelling. This energy is given in Equation 2.1 consists of two cost terms.

$$E(A) = \lambda R(A) + B(A) \tag{2.1}$$

Where,

$$\begin{gathered} R(A) = \sum_{p \in P} R_p(A_p) \qquad \text{(regional term)} \\ B(A) = \sum_{[p,q] \in N} B_{p,q} \delta_{A_p \neq A_q} \qquad \text{(boundary term)} \end{gathered} \tag{2.2}$$

P - The set of all pixels in an image

N - The neighbourhood set of pixels

A - The foreground and background labelling assigned to an image

$R(A)$ is a regional term, and varies based on what label is assigned to an individual pixel. This term

assumes that costs, such as $R(obj)$ and $R(bkg)$ are given beforehand. $R(A)$ will vary based on how the pixel intensity matches with the object intensity and background intensity. This can be determined using intensity models.

$B(A)$ is the boundary term and that measures the cost of assigning different labels to neighbouring pixels. Pixels p and q which are similar to each other should have a high $B(A)$ whereas dissimilar pixels will have a $B(A)$ close to 0. Table 2.1 gives a summary of the various edge weights used by the Graph Cuts algorithm.

Edge	Weight(cost)	for
$\{p, q\}$	$B_{p,q}$	$\{p, q\} \in N$
$\{p, S\}$	$\lambda R_p(bkg)$	$p \in P, p \notin O \cup B$
	K	$p \in O$
	0	$p \in B$
$\{p, T\}$	$\lambda R_p(obj)$	$p \in P, p \notin O \cup B$
	0	$p \in O$
	K	$p \in B$

Table 2.1: Graph Cuts Edge Weights

With the edge weights specified, the minimum cut C can be found exactly in polynomial time using any algorithm for two terminal graph cuts assuming that all edge weights above are negative.

2.2 Random Walks

The random walks algorithm for segmentation was first proposed by L. Grady [2]. Much like the graph cuts algorithm, the random walks algorithm starts off with an image and seed pixels belonging to either the background or objects. The algorithm then labels each unknown pixel by answering the question: given a random walker starting at the location of the pixel, what is the probability that it will reach each of the various seed pixels. The result is a vector of probabilities with each element representing the calculated probability of reaching that label, the label with the highest probability is then assigned as

the label to the unknown pixel.

It has been shown that the solution to this problem does not require the simulation of any random walkers and instead can be reduced to a large linear system which can be solved using existing linear system solving methods. The proof for this was given in [44] and states that the solution to the random walker problem is equivalent to the solution to the Dirichlet problem [45]. Alternatively, there exists a connection between the solution to the combinatorial Dirichlet problem and electric circuit theory, whereby the nodes of the graph are nodes in the circuit and the edges are resistors with inverse edge weights (edge weights being an indication of pixel similarity such as the case with Graph Cuts). Boundary conditions or seeds are given as voltage sources with fixed voltage [46].

2.2.1 Algorithm

The methodology behind the Random Walks process is to iteratively take each label and assign it a unit voltage of 1V, the rest of the labelled nodes are then fixed to 0 or grounded. To solve for the probabilities, the large sparse linear system is solved which will give the voltages at each unknown node. This process is repeated for K-1 times, where K is the number of different labels. The reason being, the probabilities for the last iteration can simply be found by taking the sum of all other probabilities and subtracting from 1 (principle of superposition in circuit theory).

One way of deriving the system of equations is given in [47] whereby the first step is to analyze a single 4-connected node in an electric circuit. Figure 2.2 below shows a node i of the graph and the 4 other nodes it is connected to, separated by resistors.

The voltage at node i can be found using Kirchoffs Current Law (KCL) at that point. According to KCL, the sum of all the currents going out of the node i must be equal to zero. To generalize this in an equation, it can be said that for an N-connected graph the sum of currents $I(k,i)$ going from i to k is:

$$\sum_{k=0}^{N-1} I_{i,k} = 0 \tag{2.3}$$

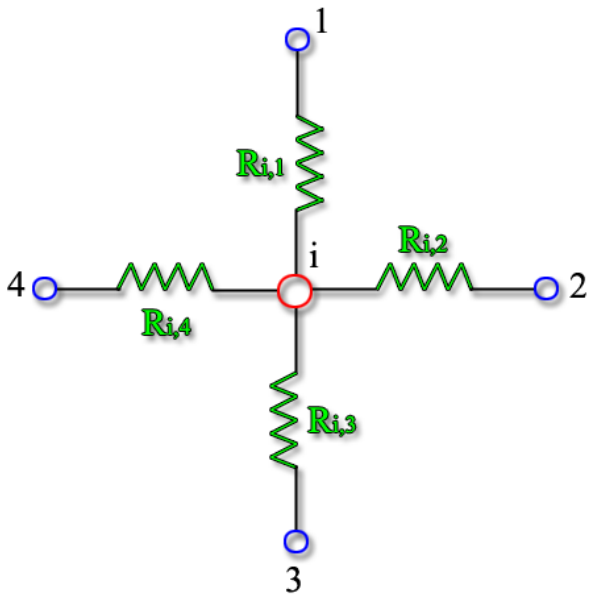


Figure 2.2: Random Walks Node Structure

This equation can be rearranged by replacing using $V = I \backslash R$ (ohms law), and for simplicity the conductance G between two nodes is taken as opposed to R, the resistance. This gives the following:

$$\sum_{k=0}^{N-1} G_{i,k}(v_i - v_k) = 0 \tag{2.4}$$

The above product can be broken down by separating the pixels whose voltages are already known (seeds) from those which are unknown and the equation can be made to look more like the system of equations which will be solved.

$$\sum_{k=0}^{N-1} G_{i,k} v_i - \sum_{k \notin S} G_{i,k} v_k = \sum_{k \notin S} G_{i.k} v_k \tag{2.5}$$

or,

$$G_{\Sigma}^{i} v_i - G_{U}^{k} v_k = G_{S}^{k} v_k \tag{2.6}$$

where S represents the set of all seed nodes. The next step in the derivation is to take the equation above which is for a single N-connected node and to apply it to all nodes of the graph. This is done by converting the equation into a Matrix expression by substituting in:

$$G_{\Sigma}^{(i,j)} = \begin{cases} G_{\Sigma}^{i} & i = j \text{ and } i \notin S \\ 1 & i = j \text{ and } i \in S \\ 0 & \text{otherwise} \end{cases} \quad G_{\Sigma}^{(i,j)} = \begin{cases} G_{i,j} & j \notin S \\ 0 & \text{otherwise} \end{cases} \quad G_{\Sigma}^{(i,j)} = \begin{cases} G_{i,j} & j \in S \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

and vectors $\vec{v}$ and $\vec{s}$ for the unknown and known voltages respectively. The equation then becomes

$$G_{\Sigma}\vec{v} - G_{U}\vec{v} = G_{S}\vec{s} \tag{2.8}$$

G_U and G_S are known as sub-adjacency matrices and together compose G the complete adjacency matrix of the graph. The fact that G is a sparse, symmetric and banded matrix can be exploited to speed up computation. G_Σ indicates the degree of each node in the graph, the degree is simply the sum of edge weights connected to a certain node. One final rearrangement can now be done to generate the final form of the Random Walks system of equations and thats to replace $G_\Sigma - G_U$ with L the laplacian matrix, which is simply defined as the difference between the degree matrix (in this case G_Σ) and the adjacency matrix (in this case G_U) and $G_S\vec{s}$ with $\vec{b}$ which is the boundary vector. The final equation is then:

$$L\vec{v} = \vec{b} \tag{2.9}$$

From here any type of linear system solver can be used to solve for $\vec{v}$ such as LU decomposition.

2.3 Edge Weights

Much as in Graph Cuts, edge weights are calculated based on pixel similarity, typically a function such as

$$w_{i,j} = exp(-\beta(g_i - g_j)) \tag{2.10}$$

where g_i and g_j are image intensities at pixels i and j. Rzeszutek *et al.*[47] found that a better weighting function for use with Random Walks is in fact the sigmoidal as opposed to the exponential. The problem with using an exponential as the edge weights is how quickly the exponential drops off, thereby potentially generating small weighting for pixels which otherwise would be considered similar. The sigmoidal function:

$$G_{i,j} = \frac{2}{1 + exp(-\beta d(i,j)^{\gamma})} \tag{2.11}$$

Is a good choice of weighting function and can be controlled using two parameters, β and γ. Figure 2.3 illustrates the difference between the two weighting functions over the range [0,0.15] where most of the drop occurs, the sigmoid is a little more lenient in its weighting admitting a wider range of edge differences. This can be good for providing partial noise rejection while maintaining an ability to better deal with soft edges.

The edge weights are one way in which the Random Walks algorithm can be exploited in order to fuse it with Graph Cuts, more on this will be explained in Chapter 3.

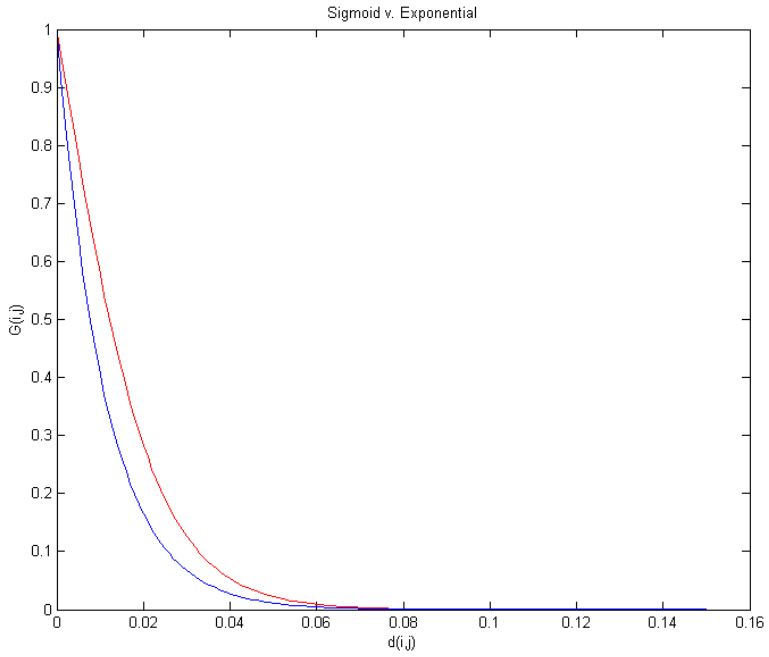


Figure 2.3: Sigmoid weighting function versus Exponential β=90, γ=1

Chapter 3

Hybrid Adaptive Graph Cuts and Random Walks

When considering a method for combining the two previously discussed algorithms, it was important to consider which features of each were desired to keep in the new algorithm. Below is a quick summary of some features of each algorithm from which the desirable ones can be selected.

- Thresholded segmentation
- Accurate strong edge segmentation
- Accurate segmentation given less seeds

Figure 3.1 shows the result of using graph cuts to create a depth map of the fish image. The red highlighted fin illustrates the hard accurate segmentation given a strong edge. Because graph cuts uses the exponential edge weighting the strong edge here is clearly differentiated from the background. Another property shown here is the thresholded segmentation result produced by graph cut. What is

(a) Fish Image

(b) Depth Map Using Graph Cuts

Figure 3.1: Fish Image segmented using graph cuts

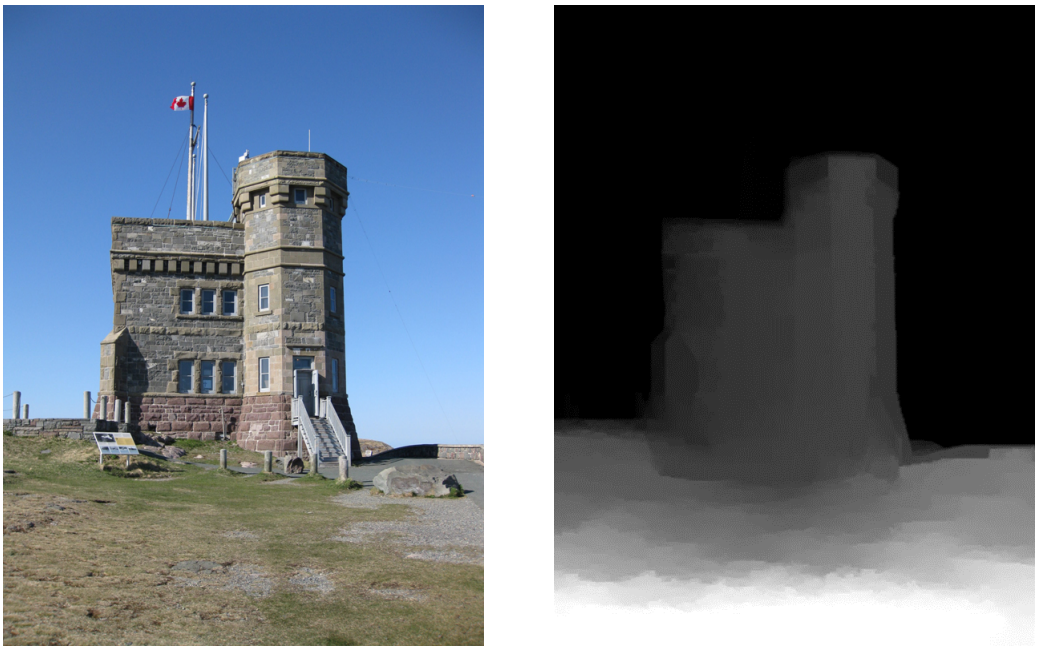
meant by this is that the final depth map is composed of integer valued depths. Graph cuts is a binary segmentation algorithm, and is designed to work on a graph with two seeds: Those coming from the background, and the foreground (objects), that is to say a background and a foreground object. When dealing with a multi-label problem, the algorithm iterates through each different label treating that as foreground while all other labels are made to be background, as a result if N labels were specified, N different binary maps will be produced. The way a label is ultimately assigned to a pixel is by looking at which segmentation produces the maximum flow for that pixel. While both features are considered good features in the realm of segmentation, only the former is desired when dealing with depth map generation. The reason why the latter is not desired is simply because flat depth maps will produce flat

objects in 3D. If all the different depths do not smoothly merge into each other, objects just appear to exist on separate planes when viewed in 3D. This problem is known as the *cardboard* effect and is one of the main problems when dealing with automated and semi-automated 2D to 3D conversion techniques.

Moving on and looking at the Random Walks algorithm, the following features can be noted:

- Continuous (gradient) segmentation
- Robust to some types of noise
- Respects weak edges

Of the above set, a few are illustrated in Figure 3.2.



(a) Tower Image (b) Random Walks Generated Depth Map

Figure 3.2: Tower Image segmented using random walks

The gradient segmentation can be clearly seen in the ground leading up to the tower. This smooth transition makes for a more natural looking scene when the image is converted to 3D. At the same time however, the flat face of the tower contains an unwanted merging between the building and the

background. Ideally, the combined algorithm should keep the desired gradients while flattening out other areas.

From previous observations, the final algorithm should not add too much more computational time to the process. As far as noise robustness is concerned there are previous works done [48] which have improved the noise robustness of the Random Walks algorithm but for the purpose of this work the standard Random Walks algorithm is used.

3.1 Approach

Restating the objective, the goal is to combine the Graph Cuts and Random Walks algorithms so that certain features of both are retained while simultaneously being adaptive. With that in mind, a novel way to combine the algorithms was found. Figure 3.3 shows an overview of the entire process, from the initial user label to the final image pair. The red highlighted areas are the additions made to [42] and which are discussed in this work. The system takes in an image, with user-defined depth labels for each pixel, and the output is the corresponding depth map. The user needs to simply mark different objects of interest and assign each a corresponding depth value. In this system, the red channel is used to accommodate for the user labeling.

These red strokes denote how close or how far the user believes the point is from the camera. Lighter strokes denote closer points, and darker strokes denote farther points. The precise depth does not need to be known since the final result is normalized and can be scaled again when synthesizing the stereoscopic image pair. Using the aforementioned labelling, a multi-label Graph Cuts segmentation algorithm is performed on the image. In [42] the Random Walks segmentation algorithm is performed in parallel with Graph Cuts, and the depth map generated from Random Walks (RW) is combined with Graph

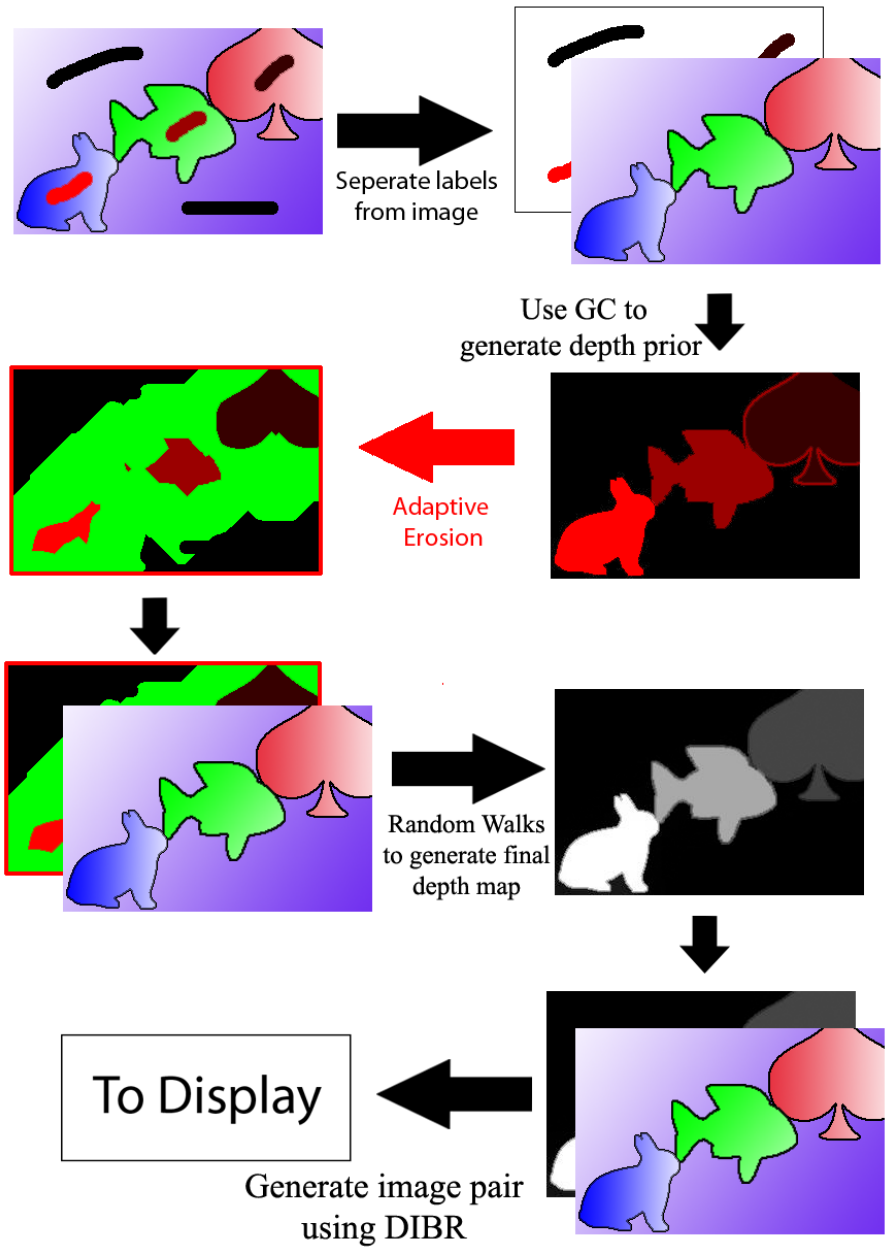


Figure 3.3: Algorithm Flow Chart

Cuts (GC) using a geometric mean and a static weighting.

$$DM = RW^{\alpha} GC^{1-\alpha} \tag{3.1}$$

Where α is a weighting factor chosen empirically. The problem with this approach is that the constant α needs to be chosen manually and does not ensure consistent results across different types of images. Furthermore, simply averaging the two results means all objects in the image are uniformly combined without regard to their depth position or size.

What is different in this work is that the output of the Graph Cuts stage is used as a depth prior that provides a rough initial estimate to ensure better accuracy. After that depth prior is generated, the labelling regions that denote each user-defined label are adaptively shrunk using a distance transform and shrinking radius which is discussed in section 3.1.3.

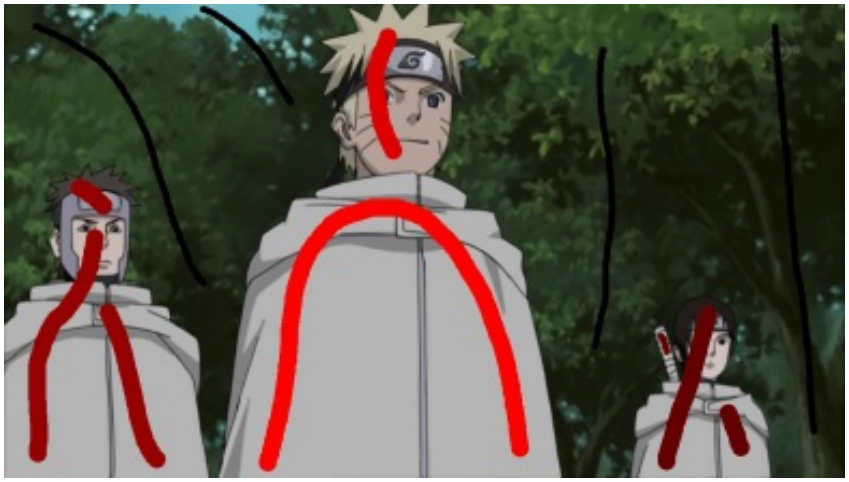
The shrunk labels are added to the initial user label and fed into the Random Walks algorithm with the image data. Contrary to [42], an adaptive way of merging the two depth maps is determined, and thus creating a final depth map that is more coherent in comparison to the original framework.

3.1.1 Graph Cuts Depth Prior

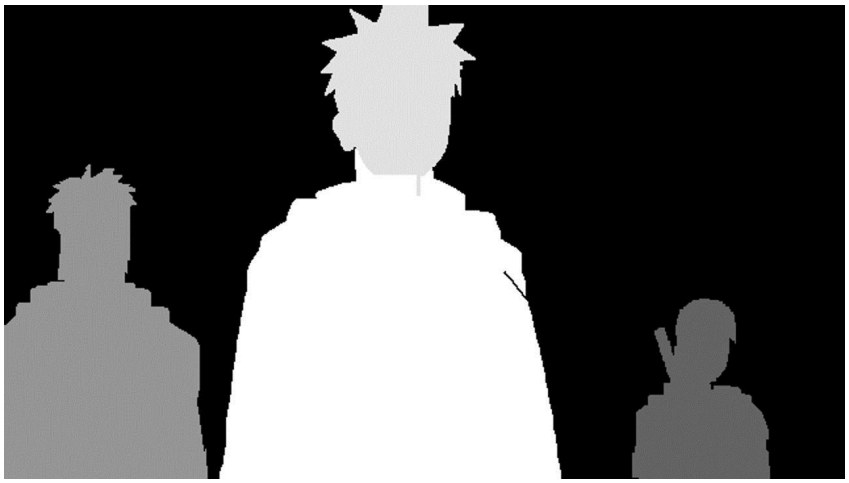
The algorithm begins by using the user-defined labels along with the original image to generate a depth map using Graph Cuts algorithm described in Chapter 2. The resulting depth map consists of labels in the range $k = [1, N_D]$ where N_D is the number of labels in the original user defined labelling. Graph Cuts requires that its labels be integer values. Therefore, to combine the depth prior with the original labels the integer labels must first be converted back to the range [0, 1]. To do this, a lookup table is used which maps the user-defined depth values to integer values.

3.1.2 Converting the Graph Cuts generated depth prior into an augmented user-defined label

The depth prior from the previous step assigns a label to all pixels as can be seen in Figure 3.4(b), if this result is taken directly and converted into a label there would be no unknown pixels for Random Walks to solve for and so that approach would not suffice. Instead, some preprocessing must be done before the depth map can be augmented with the original user-defined label.



(a) Scene with user labels applied



(b) Resulting Depth Map Using Graph Cuts

Figure 3.4: Segmenting a scene from the anime Naruto using Graph Cuts

The goal of the preprocessing is to transform the Graph Cuts depth map into a suitable label for Random Walks. The approach to achieve this will be to shrink various labels adaptively. Shrinking labels will produce unknown pixels which will be solved for by the Random Walks algorithm. The amount of shrinking dictates how much of each algorithm is used. The less an object is shrunk the flatter the final result will be as the graph cuts result will heavily dictate the final segmentation. However, by shrinking more, Random Walks will take over and produce a smoother segmentation. An added benefit to providing more seeds to the Random Walks Algorithm is that it will become sped up a little and the results produced will be a cleaner segmentation, Chapter 4 discusses the computational time of the algorithm in more detail.

3.1.3 Distance Transform to Create a Depth Prior

The way the depth prior is shrunk is by using a distance transform on each label separately. The distance transform computes the distance between every pixel in an image and the nearest non-zero pixel. The Euclidean distance is generally used and is given by:

$$d(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{3.2}$$

Where $d(i,j)$ is the Euclidean distance between the pixels i and j, and (x_i, y_i), $(x_,y_j)$ are the row and column positions of i and j respectively. As an example take Figure 3.5 representing a block in an image.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 1.4142 & 1.000 & 1.4142 & 2.2361 & 3.1623 \\ 1.0000 & 0 & 1.0000 & 2.000 & 2.2361 \\ 1.4142 & 1.0000 & 1.4142 & 1.0000 & 1.4142 \\ 2.2361 & 2.0000 & 1.0000 & 0 & 1.0000 \\ 3.1623 & 2.2361 & 1.4142 & 1.0000 & 1.4142 \end{bmatrix}$$

Figure 3.5: Distance Transform on a simple matrix

In order to make use of the distance transform, the algorithm separates each label and a different distance transform is calculated for each label. Before actually computing the distance transform a bit of preprocessing is required. The label is first inverted so that zero represents the label and non-zero represents the area which is not covered by the label. The distance transform is then calculated and inverted as well. The label is then shrunk by thresholding the distance transform for some radius which in this case is chosen adaptively as will be explained next.

The radius which is used for thresholding the distance transform is calculated according to the following equation:

$$\rho(l) = max(\rho(l))\left(1 - \frac{1 - D(l)}{3} - \frac{S(l)}{3} - \frac{T(l)}{3}\right) \tag{3.3}$$

Where $\rho(l)$ is the calculated threshold of label l and $max(\rho(l))$ is the maximum value of the distance transform for label l. Each of the other 3 terms will be explained in more detail below.

Depth Factor $D(l)$ The depth factor $D(l)$ represents the depth of an object as has been determined by the Graph Cuts algorithm. This value is normalized across all labels. The lookup table which is created during the Graph Cuts phase is used to assign depths to each label. By inverting the size factor objects in the foreground are shrunk more, and objects in the background are shrunk less. This is because most of the depth detail should be seen in the foreground objects and Random Walks is better

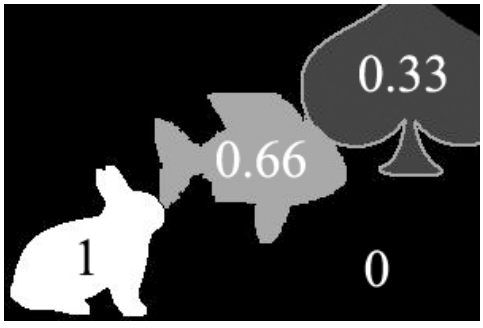
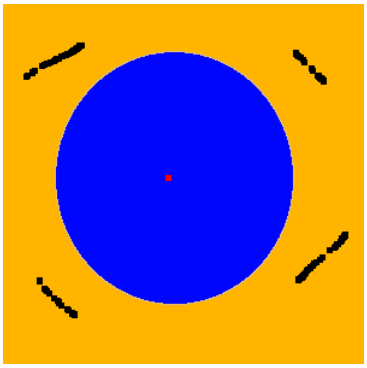
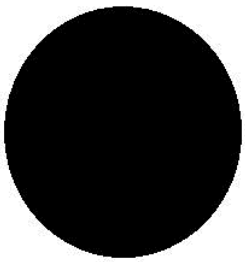


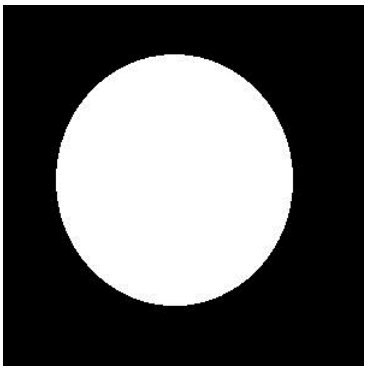
Figure 3.6: Depth Factor



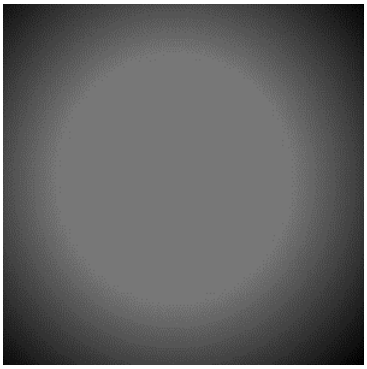
(a) Circle with labeling indicating object and background



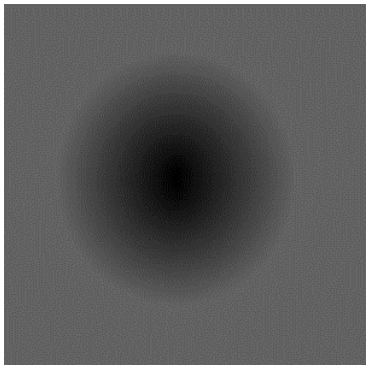
(b) Label 1 (background) shown in white



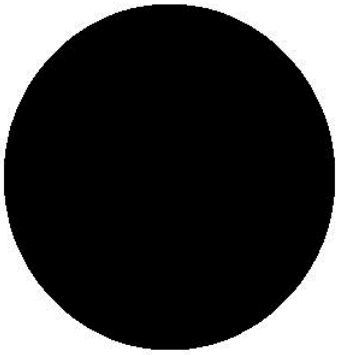
(c) Label 2 (circle) shown in white



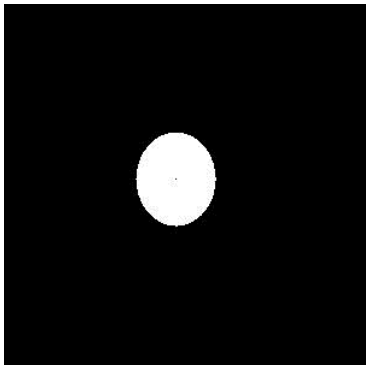
(d) Inverse distance transform for the background



(e) Inverse distance transform of the circle



(f) Background label after thresholding



(g) Object label after thresholding

Figure 3.7: Application of the distance transform to a simple image

suited as it maintains depth gradients.

Size factor $S(l)$ The size factor is calculated as follows for a label l:

$$S(l) = \frac{\sum_{i \in 1:N, j \in 1:M} l(i,j)}{M \times N} \tag{3.4}$$

where M and N are the dimensions of the image. The larger the size of a label the less the shrinking radius is. This is done in order to speed up the algorithm and to make sure large objects which could be segmented poorly given a little number of seeds are actually segmented correctly using the additional Graph Cuts information.

Texture factor $T(l)$ The texture factor is computed in a multi-step process. First the label from graph cuts is used as a mask over the initial image. This is to isolate the object which is represented with that particular label. Afterwards, a Canny Edge Detector [49] is used with an upper threshold of 0.2 and lower threshold of 0.08. Using the edge map and the original label the following equation is used to calculate the texture factor.

$$S(l) = \frac{\sum_{i \in 1:N, j \in 1:M} l_e(i,j)}{\sum_{i \in 1:N, j \in 1:M} l(i,j)} \tag{3.5}$$

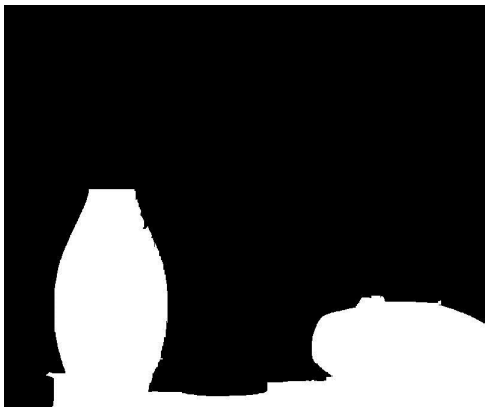
where, l_e is the edge map of label l. Objects that have a greater number of edges will be shrunk less. This is because complicated designs on an object, say an emblem on a bowling pin, or a tile on a monopoly board, can tend to interfere with the Random Walks algorithm as it may take the hard edges to mean actual object discontinuity.

By allowing more seeds to pass from the graph cuts stage the Random Walks result will represent a truer segmentation.

The results of applying Equation 3.3 to a depth prior can be seen in Figure 3.9. In Figure 3.9(a), the original user-defined labeling is shown, with the style of labeling previously mentioned. The points

(a) Bowling Image

(b) Graph Cuts Depth Map of a)

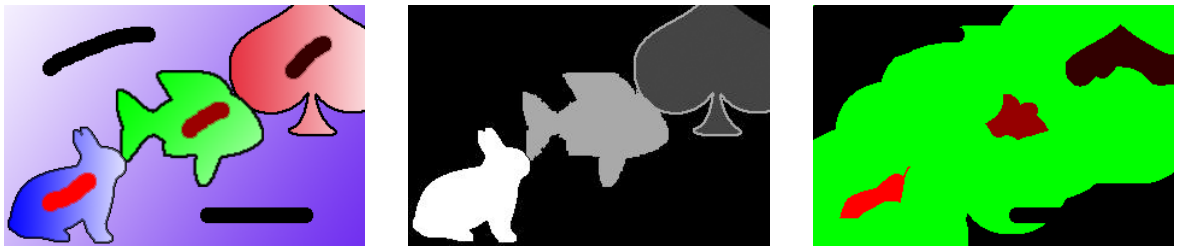


(c) Single label which will be used as a mask over initial image

(d) Edge map of the masked area

Figure 3.8: Texture Factor used in the label shrinking equation

that are untouched are considered to be unknown, and the system will thus assign depths to these points. In Figure 3.9(b), the depth prior generated from the Graph Cuts algorithm is shown. Finally, in Figure 3.9(c), the labels that are to be used for the Random Walks algorithm are illustrated, after applying the adaptive shrinking and converting back to a label. It can be seen that the number of unknown pixels has significantly reduced in size, which the Random Walks algorithm can benefit from as previously mentioned.



(a) Original User Labeling (b) Graph Cuts Depth Map) (c) Result of Adaptive Shrinking

Figure 3.9: Adaptive Shrinking of an Image

3.1.4 Random Walks using Graph Cuts Augmented Label Map and Modified Edge Weights

Once the label from the previous stage is generated, the initial user-defined label is augmented with this result. This is done in order to respect any labels the user had provided, which may not have overlapped with the generated label. In this work, a further modification is done to the edge weighting used by the Random Walks algorithm which was described in Chapter 2.

The original Euclidean distance is modified to account for an extra dimension. This dimension can be considered as a 4th colour, and is taken from the Graph Cuts depth map. As stated previously, Graph Cuts performs well with hard edges, and Random Walks works well on weaker edges. The modified

Euclidean Distance is as follows:

$$d(\vec{c_i}, \vec{c_j}, \vec{d_i}, \vec{d_j}|\alpha) = \sqrt{d(\vec{c_i}, \vec{c_j})^2 + \big(\alpha(\vec{d_i}, \vec{d_j})\big)^2} \tag{3.6}$$

where $d(\vec{c_i}, \vec{c_j})$ is the Euclidean distance of the CIE L*a*b* components, $\vec{c_i}$, $\vec{c_j}$, between pixels i and j, and $\vec{d_i}$ and $\vec{d_j}$ are the normalized Graph Cuts depth labels of pixels i and j. α is a scaling factor which determines how much effect the Graph Cuts result has on the weighting. In [42], this weight was static, and was empirically determined depending on the image that was used. To circumvent this situation, in this work, α is determined adaptively using edge information from the image. The objective is to allow the Graph Cuts result to dominate in areas of strong edges due to its edge preserving properties, while suppressing the depth map everywhere else. To do this, a Canny Edge detector is applied to the grayscale counterpart of the image, and α is generated for each pixel using the following formula:

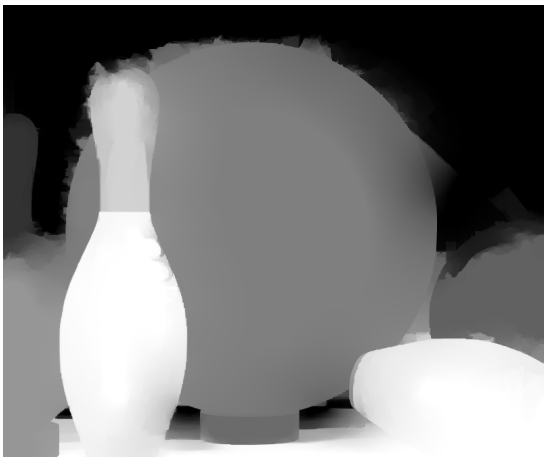
$$\alpha = 1 - exp\big(-I_e(i,j)\big) \tag{3.7}$$

where $I_e(i,j)$ is the intensity of pixel at location (i,j) in the edge detected image. Figure 3.10 shows the differences between a depth map using a constant α of 0.5 and a depth map using the adaptive weighting in Equation 3.7. It can be seen that in Figure 3.10(b), the bowling ball on the left, which in Figure 3.10(c) has a portion of it classified as the further back bowling pin, is now more uniformly labeled with a single label. Furthermore, the various objects depth gradients are not as flat.

3.2 Results

In this section a number of example images will be given demonstrating real world application of the algorithm developed. The results will be compared to a number of methods including, Graph Cuts, Random Walks and past attempts at fusing the two such as was done in [40] using averaging and weight

(a) Bowling Image with labels



(b) Depth map generated using adaptive weighting)

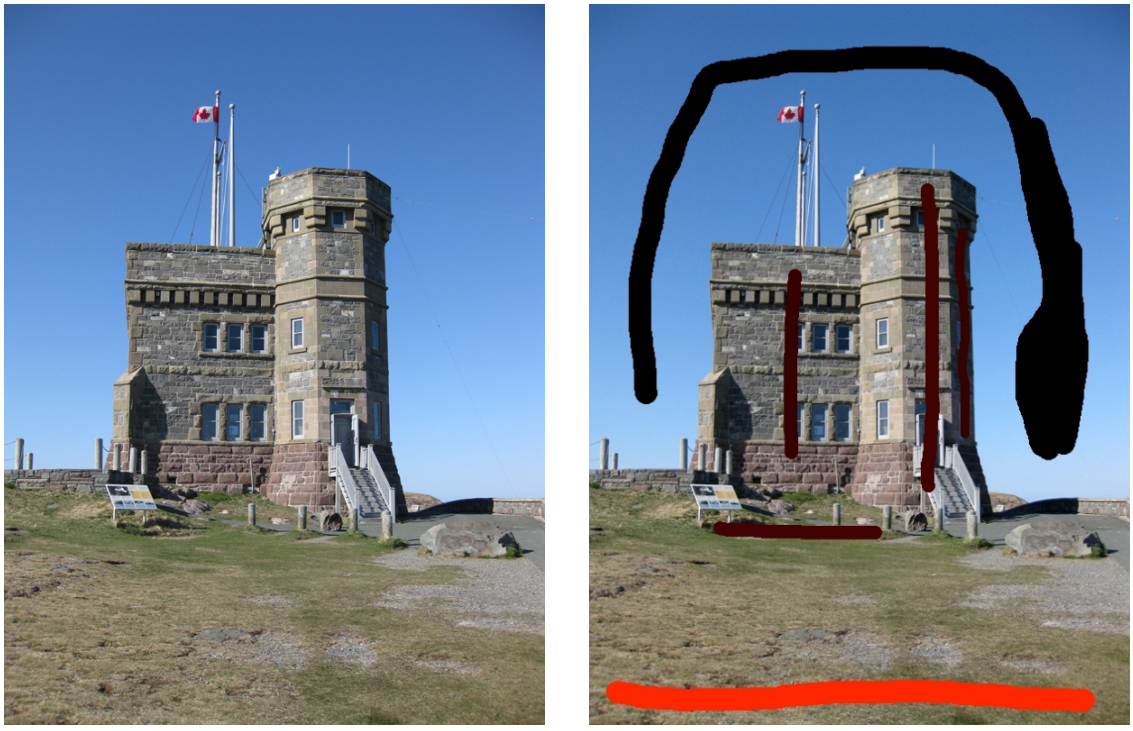
(c) Depth map generating using 0.5 weighting

Figure 3.10: Comparison of Adaptive Edge Weighting to 0.5 Edge Weighting

augmentation.

3.2.1 Tower

The first example is an image depicting a tower. The ground leading up to the tower slowly merges into the background as it approaches the tower in the distance. The tower itself is made up of strong edges separating it from the blue sky, however weak edges exist on the face of the tower between the flat face and large column. Figure 3.11(a) shows the original image and Figure 3.11(b) shows the image with the user applied labels.

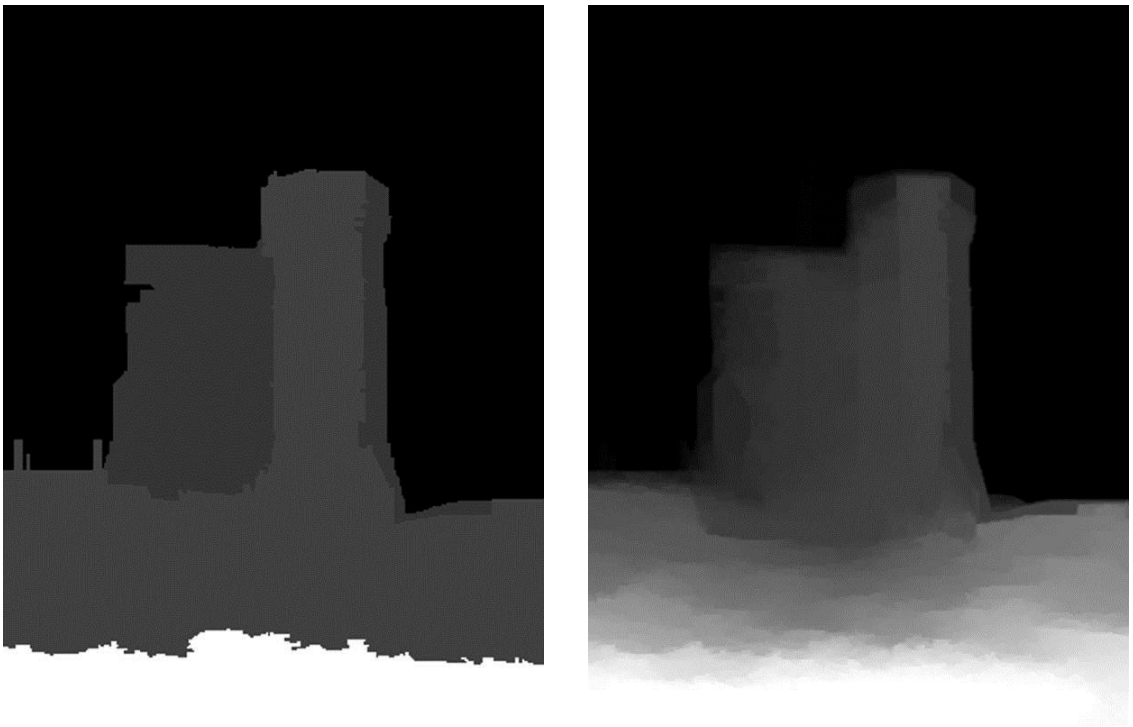


(a) Tower Image)

(b) User Labels Applied

Figure 3.11: Applying labels to the Tower Image

The resulting depth map using graph cuts, random walks and the adaptive hybrid algorithm are shown in Figure 3.12(a), Figure 3.12(b) and Figure 1.3 respectively.



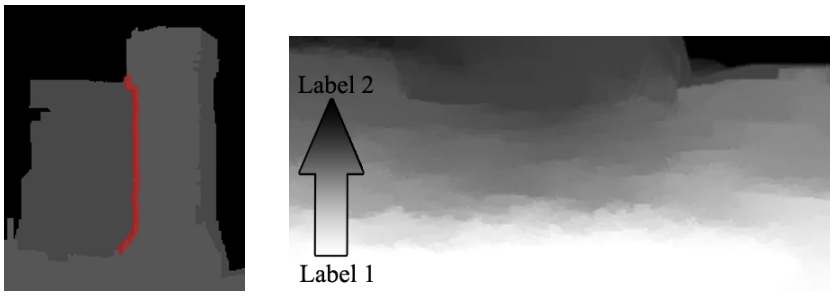
(a) Graph Cuts Result

(b) Random Walks Result

Figure 3.12: Tower Depth Maps Using Random Walks and Graph Cuts

The Graph Cuts result in Figure 3.12(a) shows a flat yet accurate segmentation of the image. This can be accredited to the solid edges which are present in the image. Even the weak edge between the right and left parts of the tower is properly detected and the depths are differentiated as shown in Figure 3.13(a). The problem with the Graph Cuts result however, is the split in the segmentation of the ground, ideally some sort of merging between the two labels would be preferred as that makes for a more natural looking 3D image.

The Random Walks result in Figure 3.12(b) on the other hand is almost the opposite of the Graph Cuts result. The gradient in the ground is clearly maintained as can be seen in Figure 3.13(b), this is because Random Walks computes a probability map when it performs its segmentation, and because the ground is fairly uniform the segmentation ends up being a gradient moving from one label to the other.



(a) Closeup on Graph Cuts Tower Segmentation)

(b) Closeup on Random Walks Gradient

Figure 3.13: Some Features of the Graph Cuts and Random Walks segmentations

Figure 3.14: Hybrid Adaptive Depth Map of the Tower Image

The combined result in Figure 1.3 shows improvement over both other images. The gradient is kept although not as much as with just Random Walks, this is because of the added seeds from the Graph Cuts depth prior which makes the segmentation less ambiguous. Looking at the face of the building, it can be seen that the erroneous left face is now fixed and is labeled as one object.

3.2.2 Monopoly

The next image is of a Monopoly board taken from the Middlebury Stereo Database[1]. This image is a tough image to segment given few labels due to the board tiles, which all have hard edges surrounding them, therefore at each tile a spike in the edge weights will occur and the segmentation wont be as accurate. This issue is seen clearly in Figure 3.16 which is the Graph Cuts result. The bottom left and top right of the Monopoly game board are both classified as background as well, the bottom right is classified as part of the foreground.

The Random Walks Result in Figure 3.17 suffers from a different problem. While the tiles of the board are not as badly segmented as in Figure 3.16 the background has an unnecessary gradient to the left. The fused result in Figure 3.18 shows improvements to both problems. The tiles are now better segmented in some cases as with the top right and bottom the segmentation is better than with Random Walks. The gradient to the left of the board is now no longer present either.

The Monopoly image demonstrates a situation which will be discussed in a little more detail in Chapter 4 and that is the problem of when more seeds are needed by the user to better segment an image. In this case the original strokes which did not include all the tiles made the segmentation difficult, despite the little information provided the algorithm still does produce very good results. The remedy for this would be to allow the user to edit the labels after a first result is produced in order to improve the generated depth map. Again this is discussed a little more in Chapter 4.

Finally, as a comparison, the ground truth disparity map for this image is included in Figure 3.19. The ground truth is calculated using the original left and right images and represents the disparities between

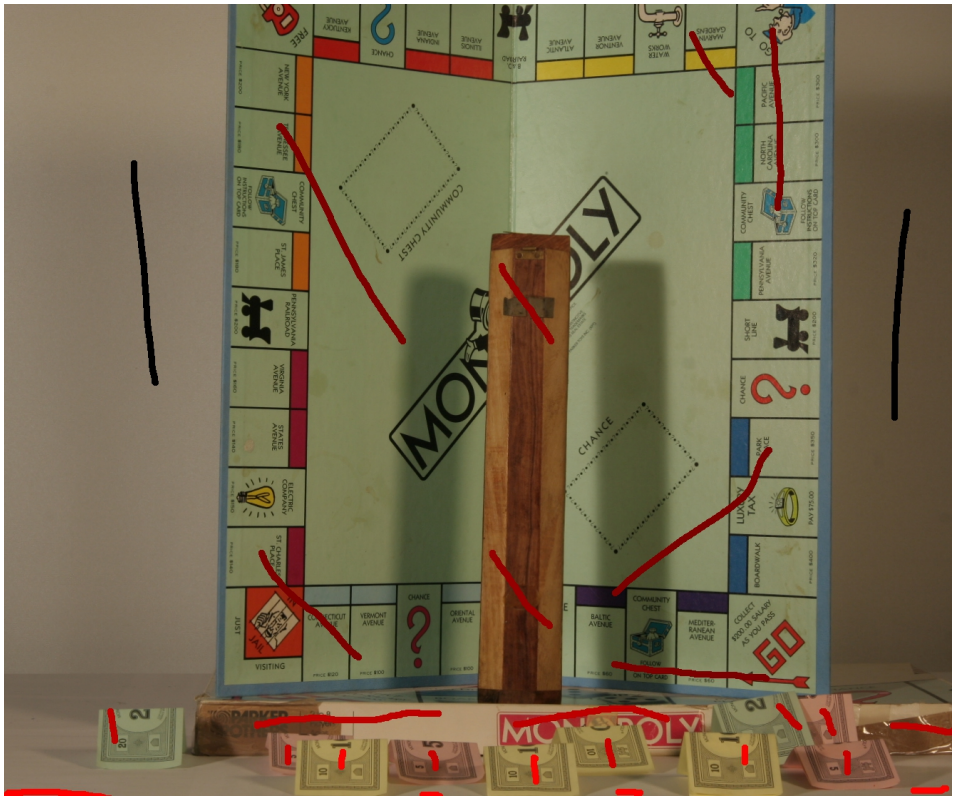


Figure 3.15: Monopoly Image with User Labels

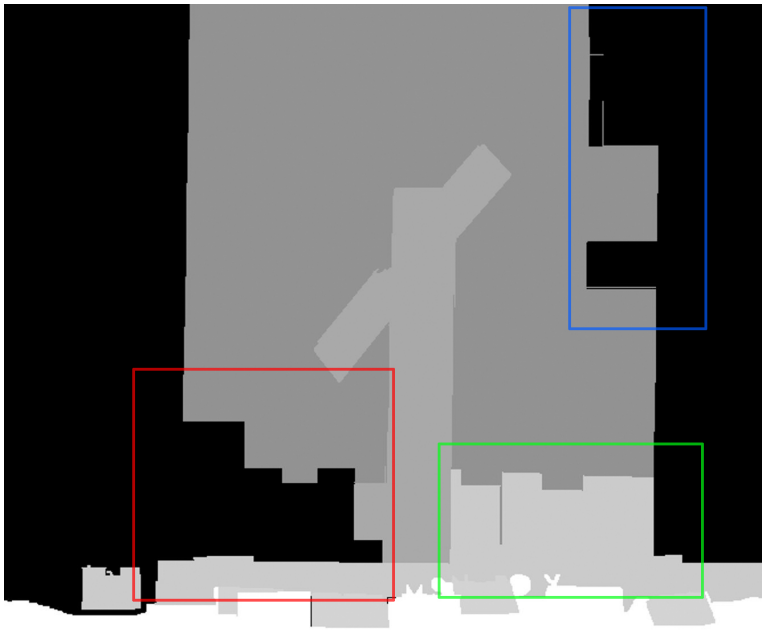


Figure 3.16: Graph Cuts Segmentation of Monopoly with Highlighted Errors

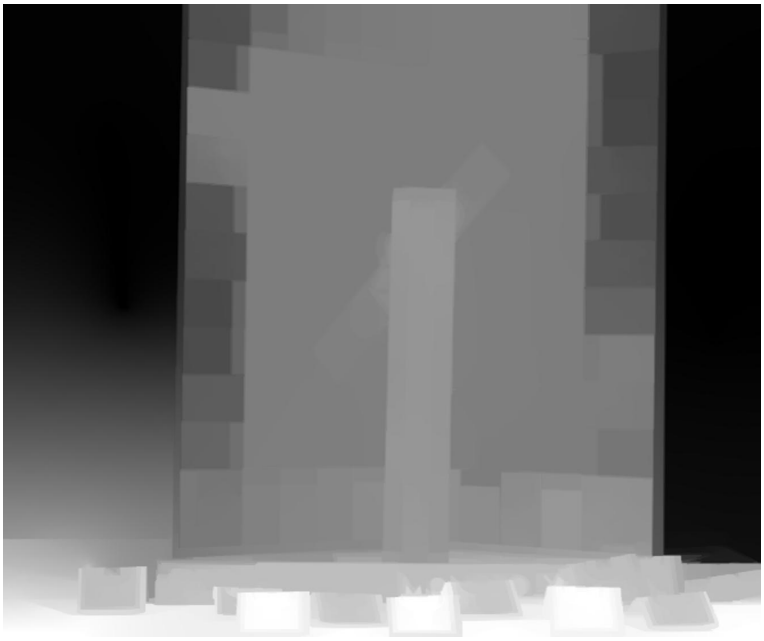


Figure 3.17: Random Walks Segmentation of Monopoly

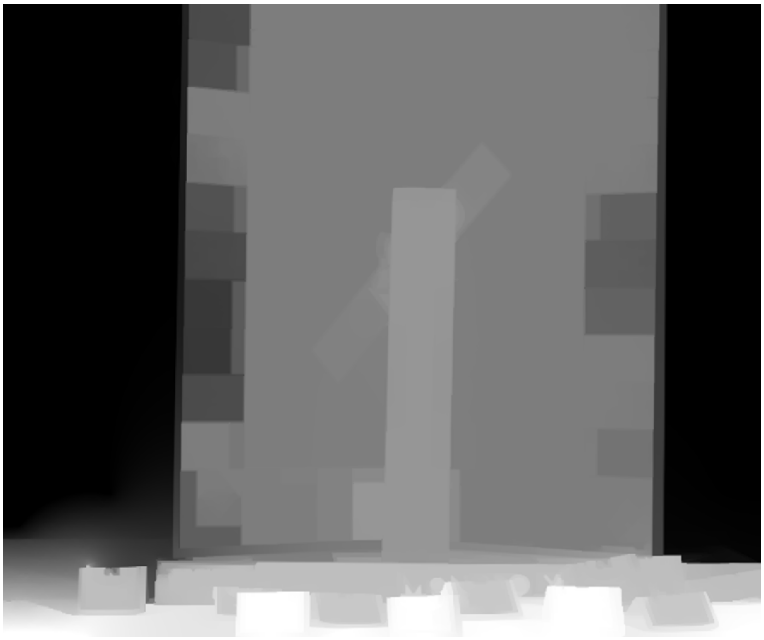


Figure 3.18: Hybrid Adaptive Depth Map of the Monopoly Image

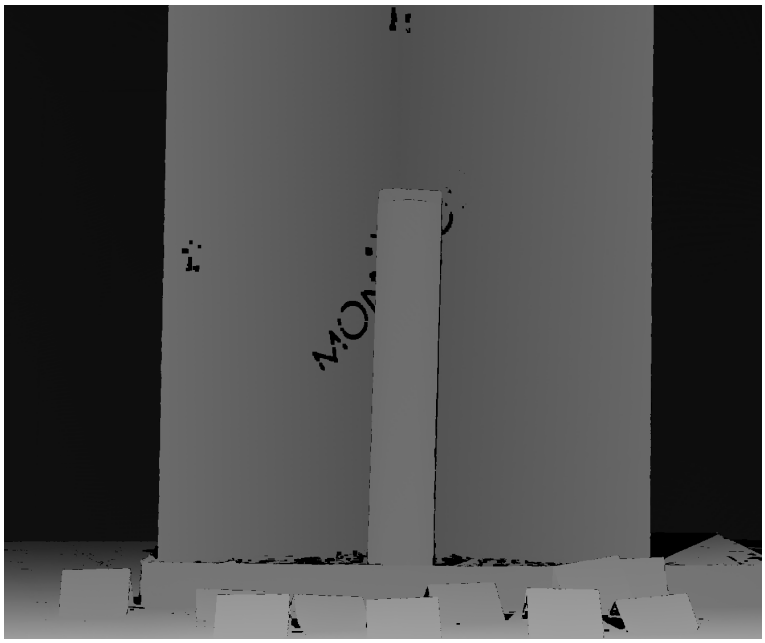


Figure 3.19: Ground Truth Disparity Map via [1]

pixels, however in this thesis the depths/disparities are user assigned. Therefore, when comparing the ground truth to the results the segmentation quality is what should be compared and not neccessarily the grayscale intensities. With that in mind, aside from the errors when segmenting the board tiles, the method presented in this thesis does a good job at segmenting the image when compared with the ground truth.

3.2.3 Superman Cartoon Scene

The next image is an example of the kind of results which can be expected for cartoon images. Cartoon images are made up of fairly solid edges, as can be seen in Figure 3.20, this makes it easy for objects to be segmented whether by using Graph Cuts or Random Walks. Despite this, both methods do have flaws which are improved on in the combined result.

Looking at Figure 3.21(a), the algorithm does a fairly good job at segmenting the image however there is an erroneous section of the left part of the airplane wing. This issue is fixed when using Random

Figure 3.20: Scene from the Superman Cartoon with User Labels Applied

Walks in Figure 3.21(b) however a new issue arises near the person in the bottom left, the segmentation is not consistent and different parts are erroneous. Furthermore, one of the buildings in the background is segmented using two different labels. This is because of the darker face of the building which creates a spike in the edge weights along the middle edge and causes the segmentation to fail.

(a) Graph Cuts Segmentation (b) Random Walks Segmentation

Figure 3.21: Superman 2 Graph Cuts and Random Walks segmentations

The combined result again has no problem segmenting the building as one object; as well the areas near the wing and the person are now fixed. Some problems still exist, namely to the immediate right of the person where the mound of dirt is fused with the nearby ground.

Figure 3.22: Hybrid Adaptive Depth Map for the Superman Scene

3.2.4 Cones

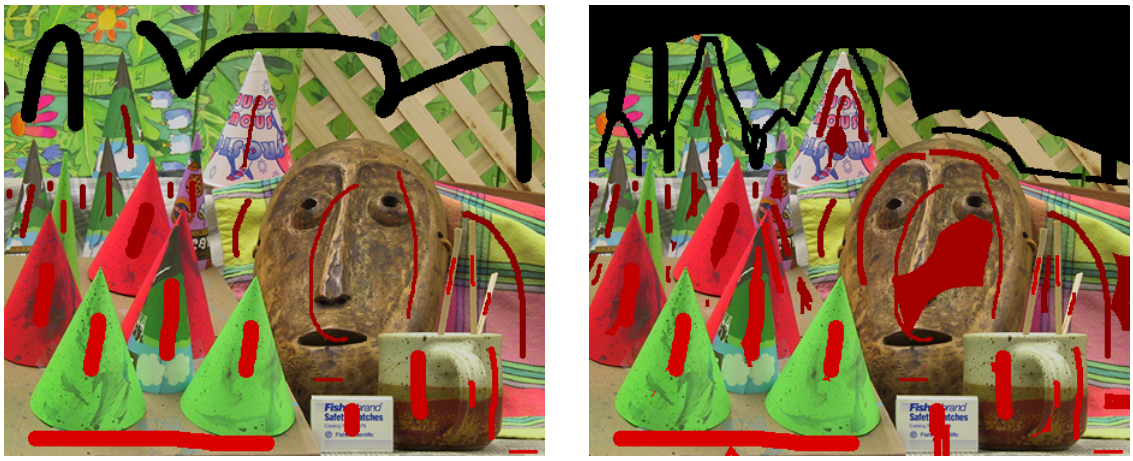
In this section the main purpose is to show a comparison between the depth map from the Middlebury Stereo Dataset[1] and a depth map generated using the method in this thesis. Again, note that it is not entirely possible to compare the Middlebury depth maps to the work done here simply because they are obtained using 2 pre-existing views taken at the same time. Whereas, the method shown in this thesis uses a single view to generate the depth map. That being said, comparing the Middlebury results with the work here can still be worthwhile for the purpose of judging segmentation accuracy, and in certain situations depth gradients.

The image used is the Cones image from the Middlebury Stereo Dataset shown in Figure 3.23. This

image is very difficult to segment given the weak edges and the patterns on the cones. Two sets of labels were used as shown in Figure 3.24; the first to produce an original depth map and the second was for an improved result.

Figure 3.23: Cones Image from the Middlebury Stereo Dataset[1]

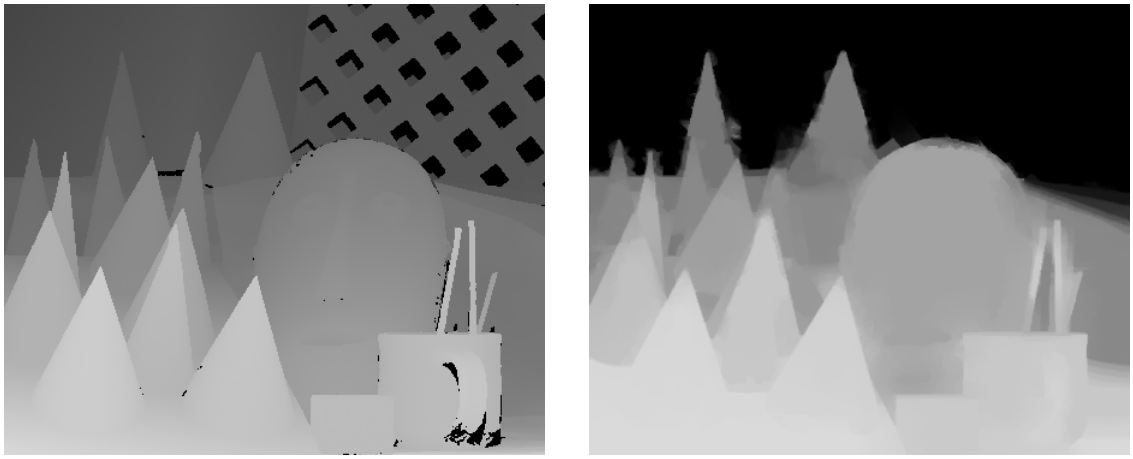
The final depth map and the Middlebury result are shown side by side in Figure 3.25. Right away it can be seen that the backgrounds are not the same, again this is because the Middlebury result uses a disparity check to generate the depth map which corresponds to a distance between pixels. In the case of Figure 3.25(b) it's not possible to know what this disparity will be and so it is all labelled with a uniform value of 0 as per the user's discretion. Otherwise, looking at the quality of the segmentation it can be seen that several of the object boundaries match and the gradients in the bottom left and right are also maintained to some extent.



(a) Image with original labels)

(b) Image with additional labels provided

Figure 3.24: The different labels used in the depth map generation process



(a) Middlebury Dataset depthmap)

(b) Depth Map generated using the method in this thesis

Figure 3.25: Comparison of Cones Depth Maps

3.2.5 Limitations

Appendix A shows more depth maps generated using a variety of older methods, specifically those developed by R. Phan *et al.*[42][43]. While the results are good when compared to older existing methods, it can still be seen that there is room for improvement. The main problem exists in creating proper

gradients in the depth maps. While Random Walks does generate some gradients, it alone is not enough to capture every gradient nor does it do a good job on any objects which aren't uniformly colored and flat. Furthermore, while the accuracy of the segmentation has been improved significantly in most cases, there does remain cases where object boundaries are incorrectly segmented. The solution to these problems is to look into adding more depth cues, or perhaps in the case of the gradient problem, by investigating a method to detect where gradients are and to generate the depth map accordingly, as opposed to relying on Random Walks to do this.

Chapter 4

Implementation

This chapter discusses some implementation details regarding the proposed algorithm, and the computational efficiency is explored and compared with other methods. Furthermore an application for 2D to 3D conversion is presented which streamlines the conversion process and gives the user several useful tools for developing good quality depth maps.

For this work MATLAB was used as the development language. Not only does MATLAB provide a lot of useful built in functions, scripting and testing are a lot more simplified than with other languages such as C++.

4.1 Graph Cuts and Random Walks Implementation

For the Graph Cuts portion of the code, a standard C library was used. In order to make use of the C library, an interface between the library and the MATLAB code is needed. MATLAB provides what is called an interface for C/C++ libraries called MEX-files, MEX stands for MATLAB Executable and allows for C, C++ or Fortran code to be run within MATLAB much like a MATLAB script or function

as long as the files are compiled correctly.

Despite the MEX-files providing an interface to the C++ Graph Cuts library, data cannot be sent directly to the C++ functions unless some preprocessing is performed. In MATLAB image data is mainly handled as 2D matrices, however the MEX function takes as input an interleaved array containing the image data and the seed data. The interleaved format in this case must be in the format shown in Figure 4.1.

Once the image and label have been interleaved they may be passed to the MEX function to be processed. Inside the MEX function a single call to the graph cuts library routine is what is used to perform the segmentation. The results which are returned are also interleaved and must be put back into proper matrix order once back in MATLAB.

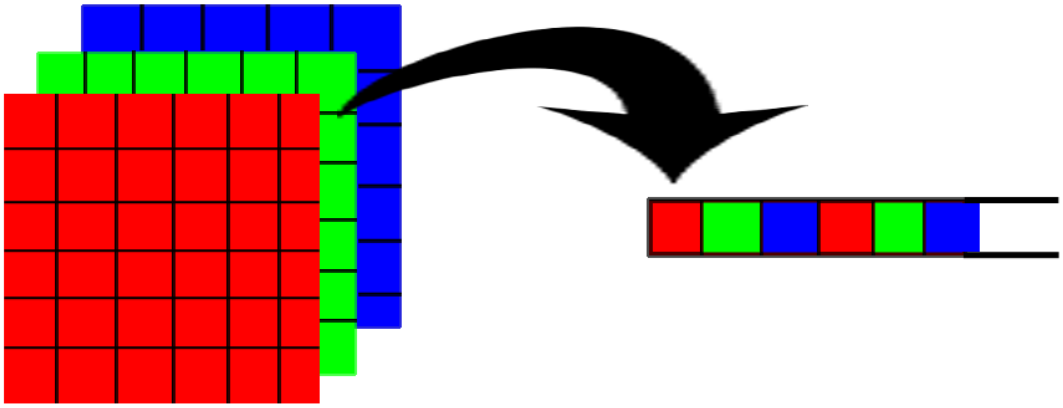


Figure 4.1: Matrix Image to Row Interleaved Image

The Graph Cuts library contains a highly optimized implementation of the Graph Cuts routine. The only disadvantage to using the library is the restriction on manipulating any of the graph cuts parameters, specifically edge weights. As was mentioned, the weighting for random walks was taken as the sigmoid function as opposed to the exponential weighting which is used by Graph Cuts, this can have an effect when attempting to make direct comparisons between the two algorithms. To remedy this, the parameters for the sigmoid the constants β and γ were chosen as 90 and 1 respectively to match

the value of β which is used by the Graph Cuts routine. As for the Random Walks implementation, everything was done all in MATLAB and is a straight forward translation of the algorithm to code, where the system of linear equations was solved using MATLAB's matrix division operator.

4.2 Performance

In order to measure performance of the proposed algorithm the MATLAB commands tic and toc were used around the code blocks which correspond to the Graph Cuts component, the Random Walks component and the code in between which prepares the Graph Cuts result to be used as a label in the Random Walks process. Furthermore a simple edit was done to the image and the time taken to regenerate the depth map was also measured. The measurements were taken on a PC with the following specifications:

- Intel i7 950 CPU @ 3.5 GHz
- 12GB DDR3 triple channel memory
- 120GB 3Gb/s Intel SATA2 SSD

Table 4.1 gives the times which were measured for the 3 images shown in Figure 4.2. It can be noted that the time to edit the depth map is always less than that of the random walks and this is because it is just the Random Walks block which is executed for depth map generations after the first time. The speed up can be credited to the increased number of seeds after the edit is performed. The overhead time refers to the time required to take the Graph Cuts result and turn it into a suitable label for Random Walks. The overhead for each image makes up a large chunk of the overall execution and is an area which could use optimization. The main component which contributes to the overhead execution time is the Canny Edge detection which is done for each label.

(a) Coast Scene)

(b) Tower Image

(c) Superman Cartoon Scene

Figure 4.2: Test Images

Image	Coast (800x600)	Tower (600x800)	Superman 2 (532x404)
Graph Cuts	0.937369s	1.629817s	0.503992s
Random Walks	1.549969s	1.576026s	0.877098s
Overhead	1.906271s	1.643386s	0.949143s
Edit	1.413576s	1.567015s	0.813056s

Table 4.1: Execution Time of the Algorithm for Various Images

4.3 2D to 3D Converter Application

In order to test the algorithm as well as the ability for users to generate accurate depth maps, a 2D to 3D conversion application was created. Figure 4.3 shows the main user interface of the application, which contains a few menu items as well as a window with information related to the brush size and color, which also corresponds to depth.

Under the File menu the user is given options to:

- Open an image
- Load an already existing label image
- Generate a depth map

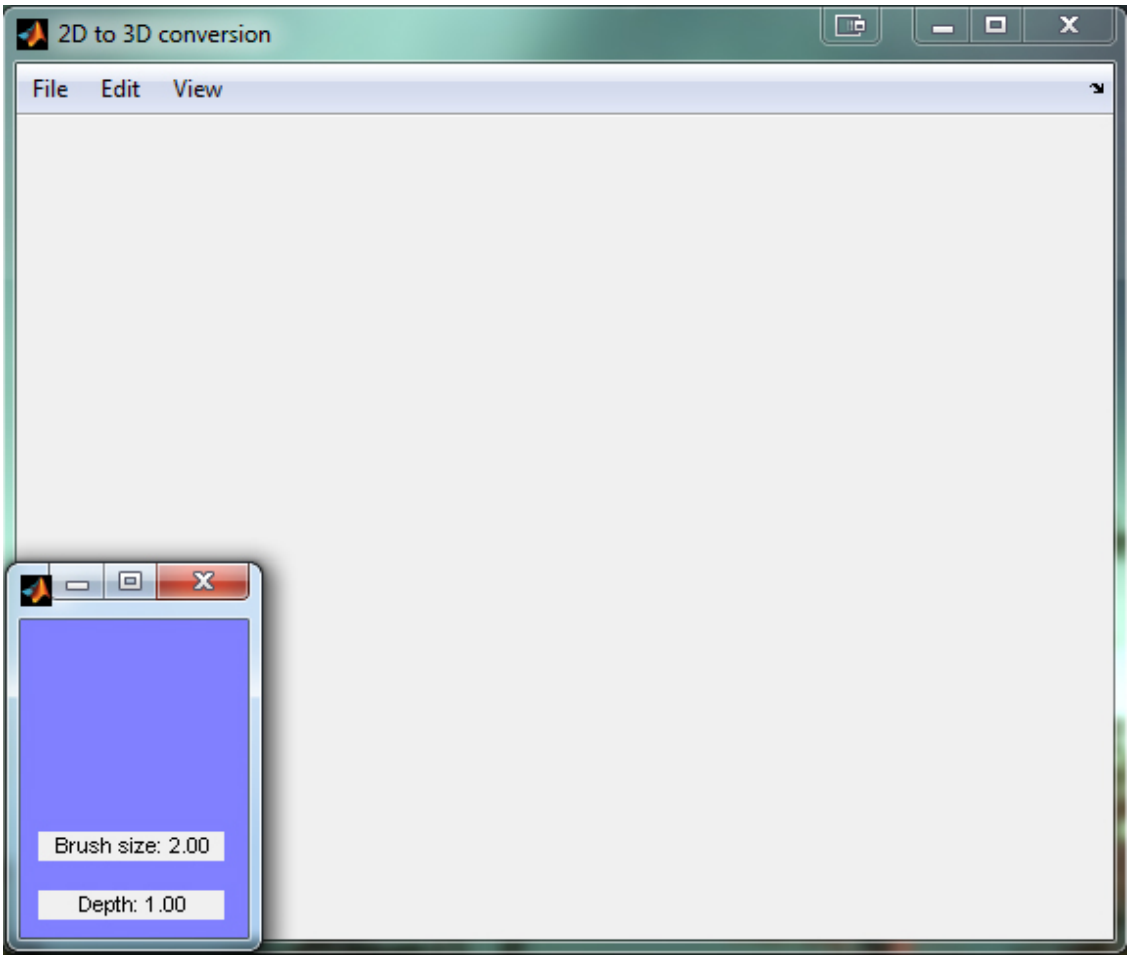


Figure 4.3: 2D to 3D Converter Main GUI

- Synthesize a new view of the image using the depth map

Once an image is loaded the process of labeling and then generating a depth map can begin. Take for example the image in Figure 4.4. The goal will be to label this image using the algorithms labeling scheme and then from there generate a depth map. The labeled image is shown in Figure 4.5 and corresponds to how the user would like to assign depths to the image. Finally the result of the algorithm is the depth map in Figure 4.6.

By looking at Figure 4.6 the user now can decide if they are satisfied with the result or if they would like to perform some editing to improve the depth map. Take for example the area highlighted

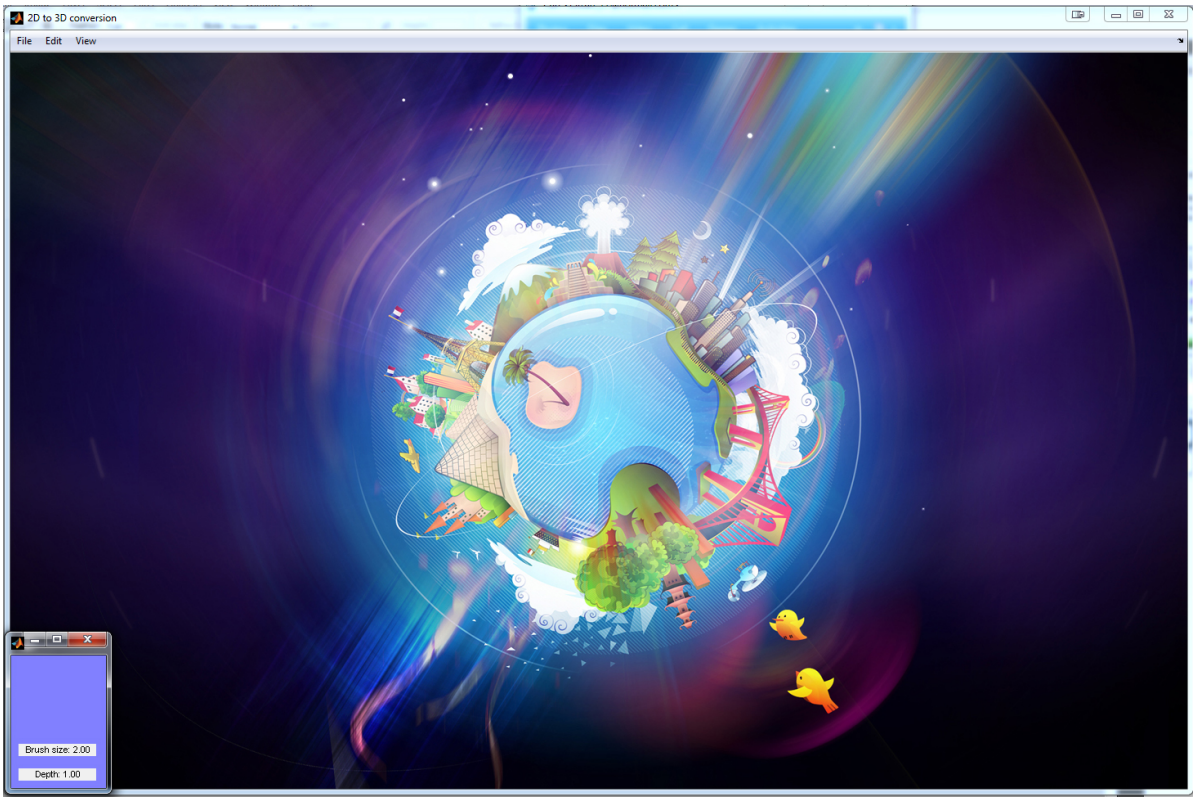


Figure 4.4: Image of a Planet to be Converted

in Figure 4.7(a) around the Eiffel Tower. If the user would like to clean up the area around here they can do so by adding in a few more strokes representing background. Figure 4.7(b) shows the cleaned up depth map. From here the user can continue to edit the depth map or move onto generating the 3D view.

As was shown in Table 4.1, the time to generate the new depth map is significantly less than the time it takes to generate the original depth map, and even slightly faster than the time to execute Random Walks alone. Once the depth map is generated the right view can be synthesized using the DIBR approach, which simply takes the original image and the depth map and applies a horizontal shift to each pixel in the image depending on the corresponding value of the pixel in the depth map. The

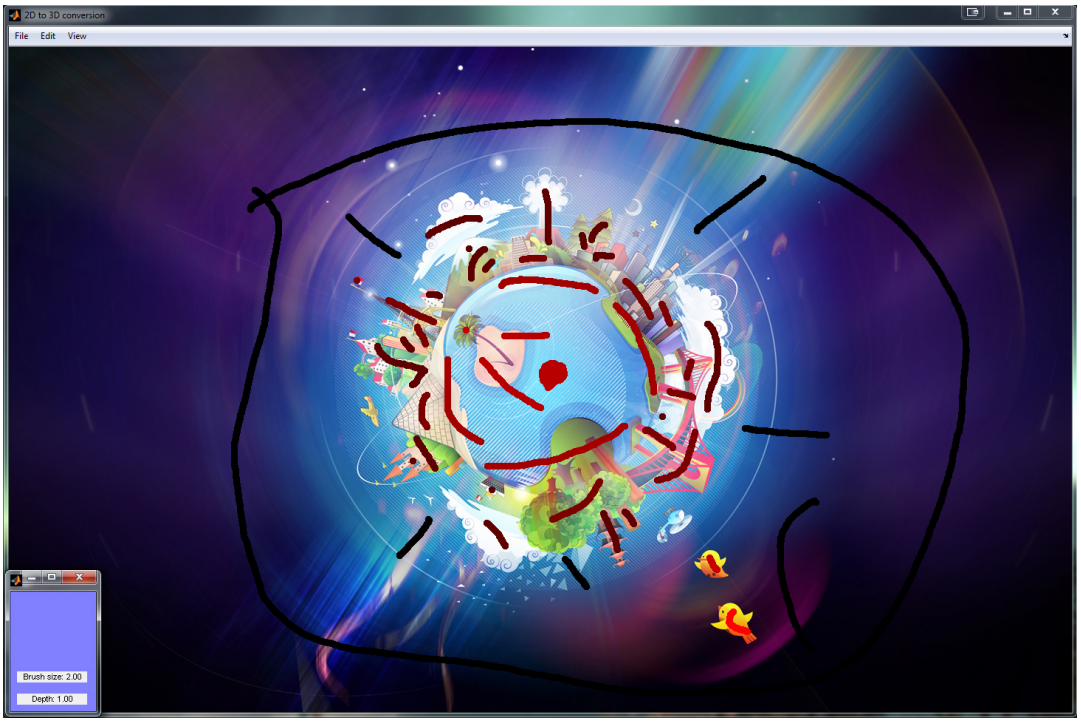


Figure 4.5: Image of the Planet with User Drawn Labels

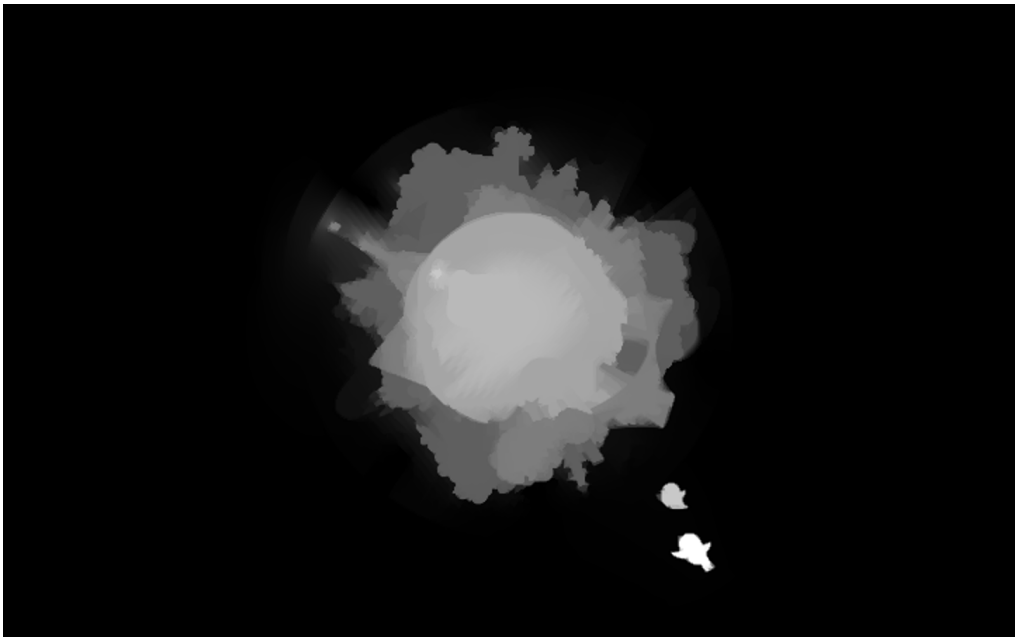
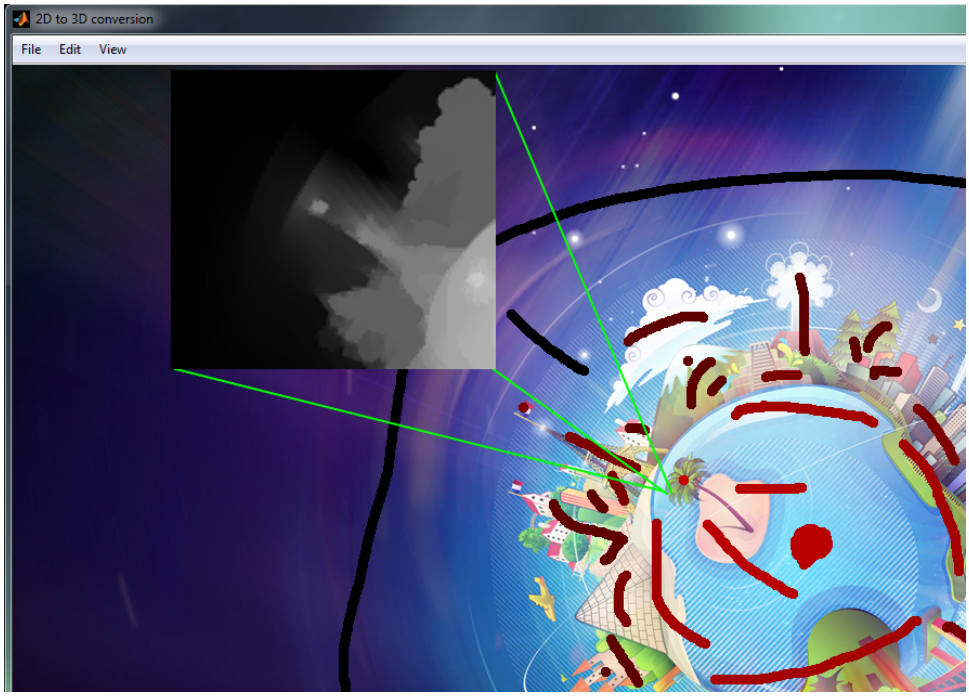


Figure 4.6: Depth Map Produced by the Algorithm

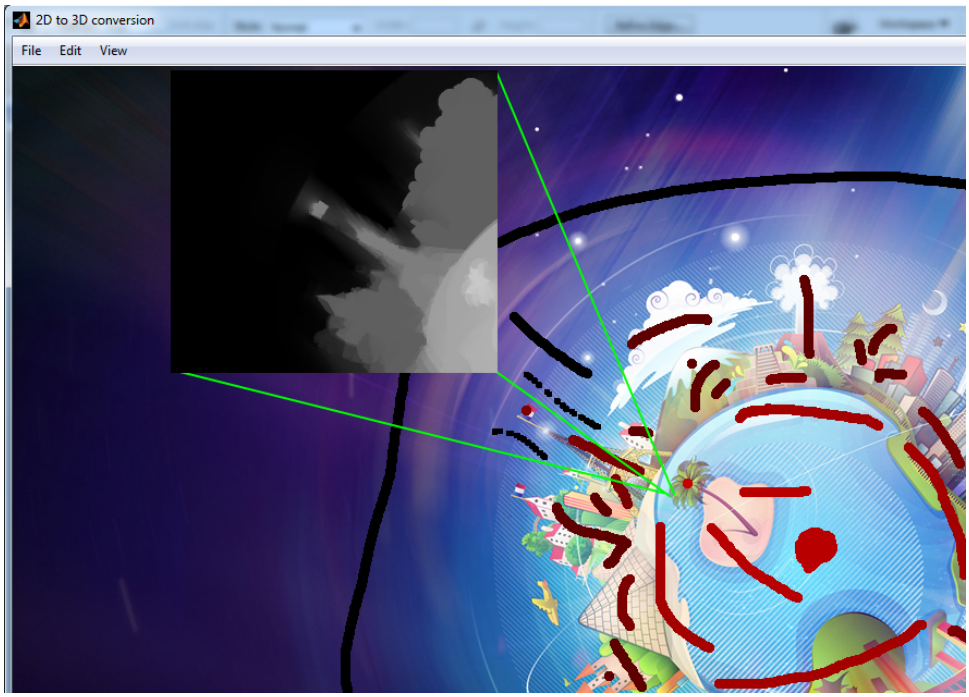
user also has the option to scale and bias the depth map such that:

$$FinalDepth = scale \times OriginalDepth + bias \tag{4.1}$$

The scale factor will control how near or far apart objects will appear in the generated 3D image, while the bias factor will determine where the convergence of the scene lies. Once the depth map is used to generate a right view any holes that may exist are filled using Random Walks. This is a simple approach to hole filling and is adequate for the scope of this work as it is more focused on the generation of the depth maps, however, more advanced techniques must be employed if the goal is to properly fix the issue of occlusions.



(a) Image with original depth map overlaid



(b) Image with corrected depth map overlaid

Figure 4.7: Depth map editing using additional labels

Chapter 5

Conclusion and Future Works

This thesis has presented a novel way to combine two segmentation algorithms, Graph Cuts and Random Walks, into a new algorithm for the purpose of 2D to 3D image conversion. Graph Cuts uses the solution to the min cut/max flow problem as a way to segment images while Random Walks solves the system of equations $Lv = b$. In both cases the algorithms take in a graph with same dimensions as the image where nodes are pixels and the edges are indicated using some similarity measure.

Both Random Walks and Graph Cuts have their advantages and disadvantages and the goal of the combined algorithm was to take the desirable features from both while suppressing the undesirable features. Graph cuts produces hard edged segmentation for strong and some weak edges but does not keep any gradients in the final result as the labels are all integer valued, this is a problem for some 2D to 3D applications where the depth of objects need to smoothly blend in with other objects around them. Random Walks on the other hand produces nice gradients in many scenes where such a feature is desired but at the same time it suffers from too much label bleeding anywhere where edges are not clearly defined. The combined algorithm was able to remedy these problems by taking a modified Graph Cuts result and feeding it into the Random Walks algorithm as a label. The results clearly show that in

all cases the combined algorithm produces better depth maps than when using either algorithm alone.

The proposed algorithm was used as the backbones for a simple 2D to 3D application which allows a user to completely convert any 2D image to 3D using a clean interface and simple tools. The application demonstrates the potential of a semi-automated approach for 2D to 3D conversion even if the scope is expanded to include 2D videos as well, such as with high end Hollywood productions.

5.1 Key Contributions

The main contributions in this thesis are as follows:

Graph Cuts Depth Prior Graph Cuts was used to generate a depth map which itself was modified and fed into the Random Walks algorithm as a label. The use of Graph Cuts as a depth prior is not novel however using it in the way described in this paper is.

Adaptive Edge Weighting A modification to the edge weighting scheme of Random Walks was also presented which takes into account the strength of the edges around objects to adaptively incorporate information from the Graph Cuts depth prior. This was an expansion of the work done by Phan *et al.* [42] which simply averaged the Graph Cuts and Random Walks results.

2D to 3D conversion application An application was created which made use of the work presented in this thesis in order to allow users to convert 2D images to 3D. The application demonstrates the potential of this method for the use in large scale conversion projects of not only images but videos as well.

The results are promising and demonstrate that even when compared to ground truth depth maps they are accurate.

5.2 Future Work

The main focus moving on with this work is to apply what has been done to 2D videos. The goal is to investigate an efficient and accurate way to generate depth maps for several frames in a sequence. This could involve investigation into object tracking and subsequently label transformation across frames or alternatively a method to translate the final depth map across frames. The key challenges with video however are related to memory usage and processing time, and as such optimizing the current method as much as possible will be important moving forward. Furthermore, interest has been shown in how this method can be applied to mobile platforms, such as the stereoscopic 3D phones available on the market today. The size of the screens typically found on mobile devices make it so that pinpoint detail is not as important when generating 3D images hence a trade-off between speed for usability can be made such that a user friendly mobile app can be developed.

Finally while the goal of this thesis from the onset was to develop a method for 2D to 3D conversion, the final result shows promise as simply a new segmentation technique, with that in mind research into how the current method can be optimized for image segmentation should also be done.

Appendix 1

Supplementary Results

This appendix includes depth map results for a set of three images. The depth maps were generated using the method discussed in this paper, as well as the two previously developed methods by R. Phan *et al*[42][43]. Readers should compare the quality of the depth maps generated using the proposed method versus the older methods. Furthermore, included in this Appendix is the depth maps generated using just edge augmentation, this is to allow the reader to see the significance of the label shrinking stage and why it was added to the algorithm.

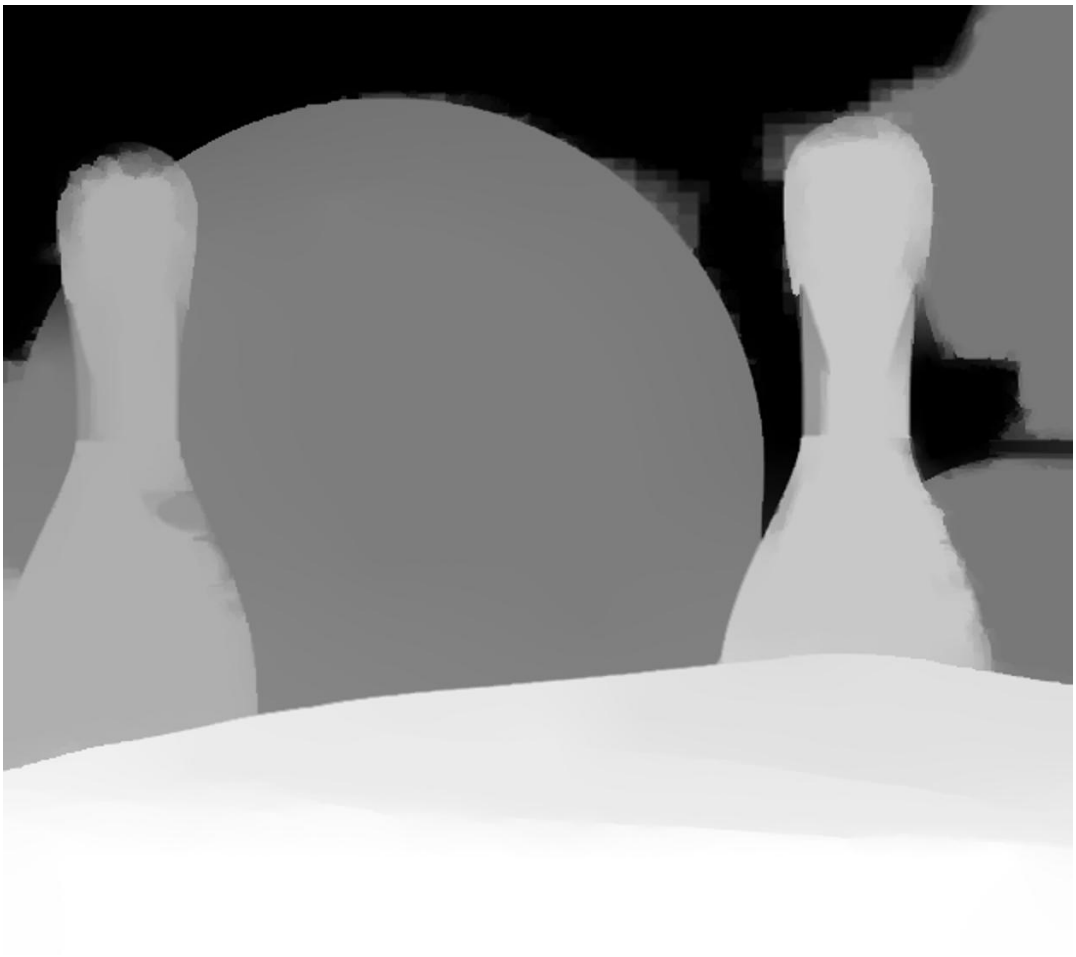
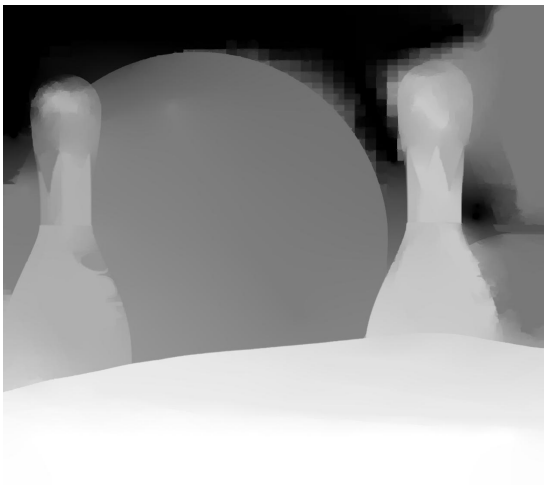
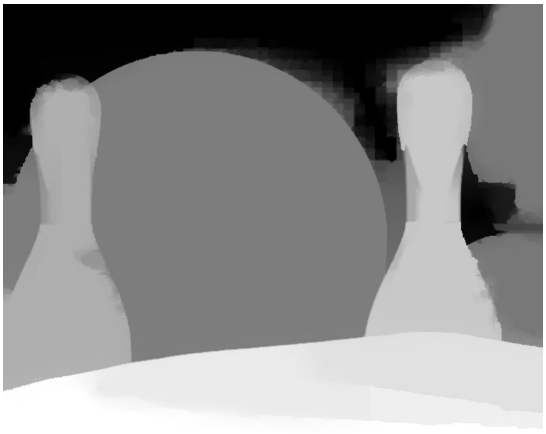


Figure 1.1: Depth Map Produced by the Proposed Algorithm



(a) Depth Map Using Geometric Averaging of GC and RW

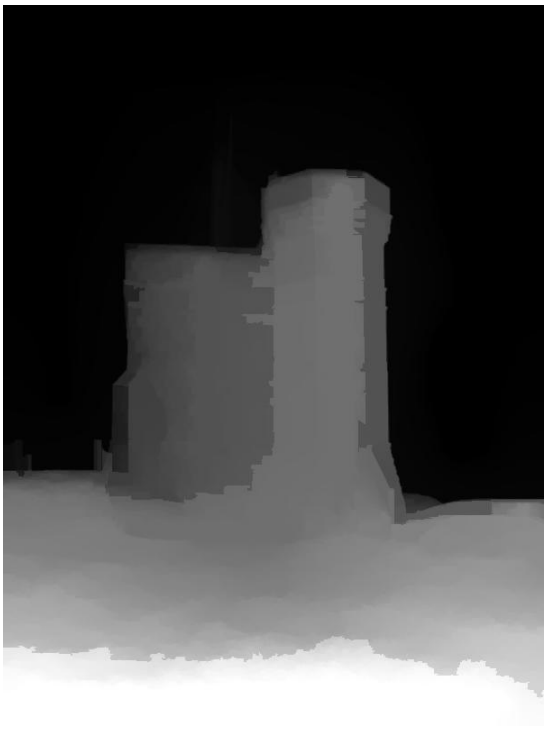
(b) Depth Map Using Weight Augmentation and an α of 0.5



(c) Depth Map Using Adaptive Weight Augmentation but no Shrinking of Labels

Figure 1.2: Depth map editing using old methods

Figure 1.3: Depth Map Produced by the Proposed Algorithm



(a) Depth Map Using Geometric Averaging of GC and RW

(b) Depth Map Using Weight Augmentation and an α of 0.5

(c) Depth Map Using Adaptive Weight Augmentation but no Shrinking of Labels

Figure 1.4: Depth map editing using old methods

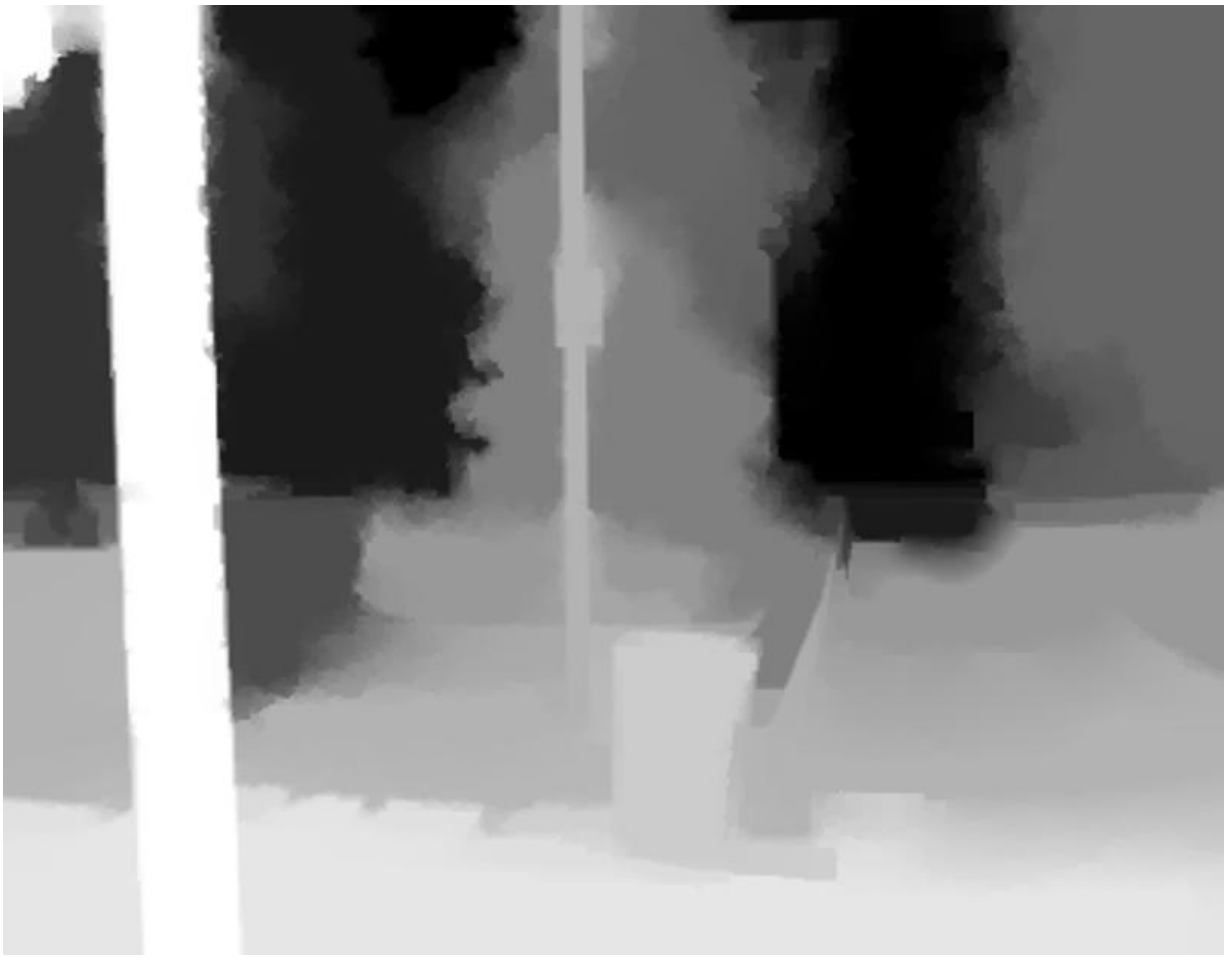
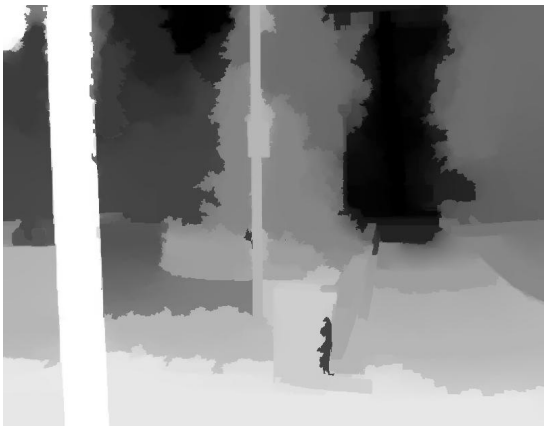
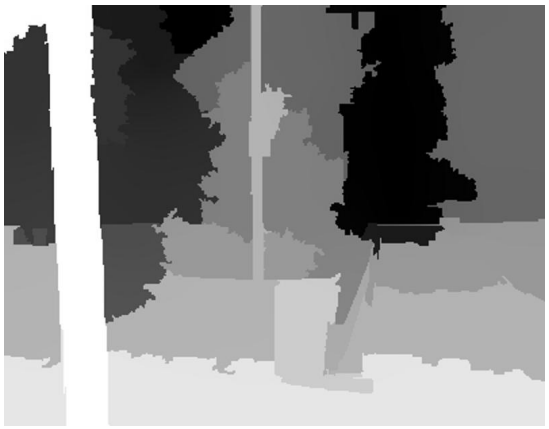


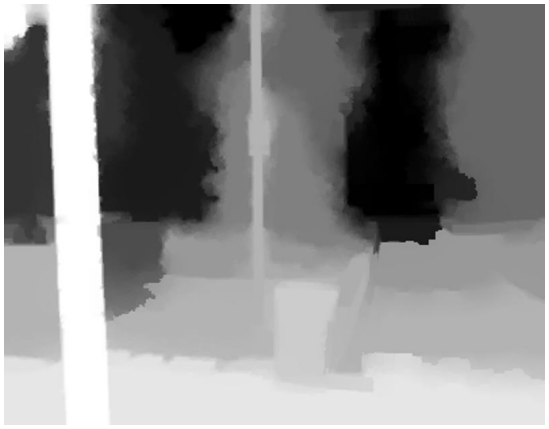
Figure 1.5: Depth Map Produced by the Proposed Algorithm



(a) Depth Map Using Geometric Averaging of GC and RW



(b) Depth Map Using Weight Augmentation and an α of 0.5



(c) Depth Map Using Adaptive Weight Augmentation but no Shrinking of Labels

Figure 1.6: Depth map editing using old methods

References

[1] H. Hirschumuller and D. Scharstein. Evaluation of cost functions for stereo matching. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2007.

[2] Leo Grady. Random walks for image segmentation. *Pattern Analysis and Machine Learning, IEEE Transactions on*, 28(11):1–17, Nov. 2006.

[3] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient n-d image segmentation. *Computer Vision, International Journal of*, 79(2):109–131, 2006.

[4] Liang-Hao Wang, Xiao-Jun Huang, Ming Xi, Dong-Xiao Li, and Ming Zhang. An asymmetric edge adaptive filter for depth generation and hole filling in 3dtv. *Broadcasting, IEEE Transactions on*, 56(3):425 –431, Sept. 2010.

[5] S. Knorr and T. Sikora. An image-based rendering (ibr) approach for realistic stereo view synthesis of tv broadcast based on structure from motion. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 6, pages VI –572 –VI –575, Oct. 2007.

[6] S. Knorr, A. Smolic, and T. Sikora. From 2d- to stereo- to multi-view video. In *3DTV Conference, 2007*, pages 1 –4, May 2007.

[7] Donghyun Kim, Dongbo Min, and Kwanghoon Sohn. A stereoscopic video generation method using stereoscopic display characterization and motion analysis. *Broadcasting, IEEE Transactions on*, 54(2):188 –197, June 2008.

[8] Feng Xu, Guihua Er, Xudong Xie, and Qionghai Dai. 2d-to-3d conversion based on motion and color mergence. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2008*, pages 205 –208, May 2008.

[9] Lai-Man Po, Xuyuan Xu, Yuesheng Zhu, Shihang Zhang, Kwok-Wai Cheung, and Chi-Wang Ting. Automatic 2d-to-3d video conversion technique based on depth-from-motion and color segmentation. In *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pages 1000 –1003, Oct. 2010.

[10] Chao-Chung Cheng, Chung-Te Li, Po-Sen Huang, Tsung-Kai Lin, Yi-Min Tsai, and Liang-Gee Chen. A block-based 2d-to-3d conversion system with bilateral filter. In *Consumer Electronics, 2009. ICCE '09. Digest of Technical Papers International Conference on*, pages 1 –2, Jan. 2009.

[11] Zhijie Zhao, Ming Chen, Long Yang, Zhipeng Fan, and Li Ma. 2d to 3d video conversion based on interframe pixel matching. In *Information Science and Engineering (ICISE), 2010 2nd International Conference on*, pages 3380 –3383, dec. 2010.

[12] Ching-Lung Su, Kang-Ning Pang, Tse-Min Chen, Guo-Syuan Wu, Chia-Ling Chiang, Hang-Rnei Wen, Lung-Sheng Huang, Ya-Hsin Hsueh, and Shau-Yin Tseng. A real-time full-hd 2d-to-3d conversion system using multicore technology. In *Multimedia and Ubiquitous Engineering (MUE), 2011 5th FTRA International Conference on*, pages 273–276, June 2011.

[13] Guo-Shiang Lin, Cheng-Ying Yeh, Wei-Chih Chen, and Wen-Nung Lie. A 2d to 3d conversion scheme based on depth cues analysis for mpeg videos. In *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pages 1141 –1145, July 2010.

[14] Jie Feng, Hai Huang, Yayu Zheng, and Wei Zhu. Stereoscopic 3d conversion from 2d video in h.264/avc compressed domain. In *Image and Signal Processing (CISP), 2011 4th International Congress on*, volume 1, pages 565 –568, Oct. 2011.

[15] M.T. Pourazad, P. Nasiopoulos, and R.K. Ward. Conversion of h.264-encoded 2d video to 3d format. In *Consumer Electronics (ICCE), 2010 Digest of Technical Papers International Conference on*, pages 63 –64, Jan. 2010.

[16] Xiaojun Huang, Lianghao Wang, Junjun Huang, Dongxiao Li, and Ming Zhang. A depth extraction method based on motion and geometry for 2d to 3d conversion. In *Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on*, volume 3, pages 294 –298, Nov. 2009.

[17] Sung-Fang Tsai, Chao-Chung Cheng, Chung-Te Li, and Liang-Gee Chen. A real-time 1080p 2d-to-3d video conversion system. In *Consumer Electronics (ICCE), 2011 IEEE International Conference on*, pages 803 –804, Jan. 2011.

[18] Chao-Chung Cheng, Chung-Te Li, and Liang-Gee Chen. A 2d-to-3d conversion system using edge information. In *Consumer Electronics (ICCE), 2010 Digest of Technical Papers International Conference on*, pages 377 –378, Jan. 2010.

[19] Zhebin Zhang, Yizhou Wang, Tingting Jiang, and Wen Gao. Visual pertinent 2d-to-3d video conversion by multi-cue fusion. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 909 –912, Sept. 2011.

[20] Yu-Lin Chang, Chih-Ying Fang, Li-Fu Ding, Shao-Yi Chen, and Liang-Gee Chen. Depth map generation for 2d-to-3d conversion by short-term motion assisted color segmentation. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 1958 –1961, July 2007.

[21] Jiahong Zhang, You Yang, and Qionghai Dai. A novel 2d-to-3d scheme by visual attention and occlusion analysis. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2011*, pages 1 –4, May 2011.

[22] Jae-Il Jung and Yo-Sung Ho. Depth map estimation from single-view image using object classification based on bayesian learning. In *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2010*, pages 1 –4, June 2010.

[23] Lars Schnyder, Oliver Wang, and Aljoscha Smolic. 2d to 3d conversion of sports content using panoramas. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 1961 –1964, Sept. 2011.

[24] V. Nedovic, A.W.M. Smeulders, A. Redert, and J.-M. Geusebroek. Depth estimation via stage classification. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2008*, pages 77 –80, May 2008.

[25] Fengli Yu, Ju Liu, Yannan Ren, Jiande Sun, Yuling Gao, and Wei Liu. Depth generation method for 2d to 3d conversion. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2011*, pages 1 –4, May 2011.

[26] Xiao-Ling Deng, Xiao-Hua Jiang, Qing-Guo Liu, and Wei-Xing Wang. Automatic depth map estimation of monocular indoor environments. In *MultiMedia and Information Technology, 2008. MMIT '08. International Conference on*, pages 646 –649, Dec. 2008.

[27] Yue Feng, Jinchang Ren, and Jianmin Jiang. Object-based 2d-to-3d video conversion for effective stereoscopic content generation in 3d-tv applications. *Broadcasting, IEEE Transactions on*, 57(2):500 –509, June 2011.

[28] Yi-Che Chen, Yi-Chin Wu, Chih-Hung Liu, Wei-Chih Sun, and Yung-Chang Chen. Depth map generation based on depth from focus. In *Electronic Devices, Systems and Applications (ICEDSA), 2010 Intl Conf on*, pages 59 –63, April 2010.

[29] Y. F, J. Jayaseelan, and J. Jiang. Cue based disparity estimation for possible 2d to 3d video conversion. In *Visual Information Engineering, 2006. VIE 2006. IET International Conference on*, pages 384 –388, Sept. 2006.

[30] Te-Wei Chiang, Tienwei Tsai, Yu-Hong Lin, and Mann-Jung Hsiao. Fast 2d to 3d conversion based on wavelet analysis. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 3444 –3448, Oct. 2010.

[31] Ge Guo, Nan Zhang, Longshe Huo, and Wen Gao. 2d to 3d convertion based on edge defocus and segmentation. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 2181 –2184, April 2008.

[32] Hao Liu, Wei Guo, Chao Lu, and Jizeng Wei. An efficient stereoscopic game conversion system for embedded platform. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 1235 –1240, Nov. 2011.

[33] W.-N. Lie, C.-Y. Chen, and W.-C. Chen. 2d to 3d video conversion with key-frame depth propagation and trilateral filtering. *Electronics Letters*, 47(5):319 –321, 3 2011.

[34] Yanfei Wu, Ping An, Ping Wang, and Zhao Yang Zhang. Stereoscopic video conversion based on depth tracking. In *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pages 1190 –1193, Oct. 2010.

[35] Youwei Yan, Feng Xu, Qionghai Dai, and Xiaodong Liu. A novel method for automatic 2d-to-3d video conversion. In *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2010*, pages 1 –4, June 2010.

[36] David Cardinal. Titanic: How do you convert a movie to 3d, anyway? `http://www.extremetech.com/extreme/124965-titanic-how-do-you-convert-a-movie-to-3d-anyway/2`.

[37] B. Ward, Sing Bing Kang, and E.P. Bennett. Depth director: A system for adding depth to movies. *Computer Graphics and Applications, IEEE*, 31(1):36 –48, Jan. 2011.

[38] M. Guttmann, L. Wolf, and D. Cohen-Or. Semi-automatic stereo extraction from video footage. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 136 –142, Oct. 2009.

[39] H. Emrah Tasli and K. Ugur. Interactive 2d 3d image conversion method for mobile devices. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2011*, pages 1 –4, May 2011.

[40] Junsoo Kim, Yoonsik Choe, and Yong-Goo Kim. Robust mrf-based object tracking and graph-cut-based contour refinement for high quality 2d to 3d video conversion. In *Communications, Computers and Signal Processing (PacRim), 2011 IEEE Pacific Rim Conference on*, pages 358 –363, aug. 2011.

[41] Xun Cao, Zheng Li, and Qionghai Dai. Semi-automatic 2d-to-3d conversion using disparity propagation. *Broadcasting, IEEE Transactions on*, 57(2):491 –499, June 2011.

[42] Raymond Phan, Richard Rzeszutek, and Dimitrios Androutsos. Semi-automatic 2d to 3d image conversion using scale-space random walks and a graph cuts based depth prior. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 865 –868, Sept. 2011.

[43] R. Phan, R. Rzeszutek, and D. Androutsos. Semi-automatic 2d to 3d image conversion using a hybrid random walks and graph cuts based approach. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 897 –900, May 2011.

[44] S. Kakutani. Markov processes and the dirichlet problem. *Proceedings of the Japan Academy*, 21:227–233, 1989.

[45] R. Courant and D. Hilbert. *Methods of Mathematical Physics.* John Wiley and Sons, 1989.

[46] P. Doyle and L. Snell. *Random Walks and Electric Networks.* The Mathematical Association of America, 1984.

[47] R. Rzeszutek. Image segmentation through the scale-space random walker. Master's thesis, Ryerson University, 2009.

[48] T. El-Maraghi R. Rzeszutek and D. Androutsos. Interactive rotoscoping through scale-space random walks. In *Multimedia and Expo. IEEE International Conference on*, pages 1334–1337, 2009.

[49] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 8(6):679–698, Nov. 1986.