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

Mohammad Fawaz *Raymond Phan* *Richard Rzeszutek* *Dimitrios Androutsos*

Department of Electrical and Computer Engineering - Ryerson University
350 Victoria St., Toronto, Ontario, M5B 2K3, Canada
{mfawaz,rphan,rrzeszut,dimitri}@ee.ryerson.ca

ABSTRACT

In this paper, we propose 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 labeling 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, we have made the previous approach adaptive, as well as improved the quality of the results. This is achieved by feeding information from the Graph Cuts result into the Random Walks process at two different stages, and using edge and spatial information to adapt various weights. The results show that we can produce good quality stereoscopic 3D image pairs using a simple yet adaptive approach.

Index Terms— 2D to 3D conversion, Random Walks, Graph Cuts, Depth Perception, Depth Maps

1. INTRODUCTION

Depth perception in 2D images is an area in computer vision which has seen an increase in interest in the last several decades. Specifically, stereoscopy is the technique through which we can perceive depth by using pairs of left and right images. The images must be directed to each eye separately, and each image must represent the view which would normally be seen from that eye. Today we find stereoscopic 3D used in a wide range of multimedia applications, such as 3DTVs, IMAX 3D theatres, video games and animation. Research in this area can be decomposed into content delivery and content generation; however, the latter is more important due to the limited number of methods available for generating stereoscopic content for 3D technologies.

There are two popular methods for generating 3D content. One is to use multiple cameras to capture content at the same time, with the cameras being offset a certain distance from one another. This approach produces good quality 3D content however it is expensive, requiring specialized hardware, a significant amount of post processing, and can generally only be used for creating new content. The other approach is to take existing 2D content and convert it to 3D; this requires the estimation of depth from a single image, which is generally a difficult task, as depth information is naturally lost when dealing with a single view of a scene.

There have been several attempts at 2D to 3D conversion in the past, which focus on generating a depth map which can be used to create a stereoscopic image pair. The depth map is a grayscale image, indicating how far or near objects are from the screen. A bright value would indicate a close object and a dark value would indicate an object is far away from the screen. A depth map, along with a 2D image, can be used to generate a stereoscopic 3D image pair. Tam *et al.* [1] used the Cr channel of an image after converting to YCbCr as the depth map. Cheng *et al.* [2] generated depth maps by using edge information along with hypothesized depth gradient models. Jung *et al.* [3] use an object classifier based on the Bayesian learning algorithm in order to classify objects in a 2D image into 4 different types, and subsequently assign a relative depth value. Saxena *et al.* [4] used a supervised learning approach by training a classifier with a set of images and their corresponding ground-truth depth maps. Depth maps are predicted as a function of the input image by using a set of filters to generate a feature vector. The benefit of the above methods is that they are all relatively automated. However, they do not provide any means of correcting errors, and they generally require heavy preprocessing.

In this work, we focus on a user-guided, semi-automated approach for 2D to 3D conversion. An example of such an approach would be the work done by Guttmann *et al.* [5] where user-defined labels are used along with four equations defining piecewise continuity of the depth values both spatially and temporally and then solved in a Random Walks fashion. Afterwards, training and classifying with a SVM classifier is used for the final depth values. Phan *et al.* [6] use a hybrid Graph Cuts and Random Walks approach, where two depth maps are generated separately using each algorithm and then the two are combined by ways of a geometric mean. The proposed work expands on [6], by changing the way the two algorithms are combined to make it adaptive, without adding too much complexity or computational time.

2. METHOD

Our objective is to combine the Graph Cuts [7] and Random Walks [8] algorithms so that certain features of both are retained while simultaneously being adaptive. With that in mind, we determined a novel way to combine the algorithms.

Fig. 1 shows an overview of the entire process, from the initial user label to the final image pair. The red highlighted areas are what we focused on for 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 our system, we use the red channel 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

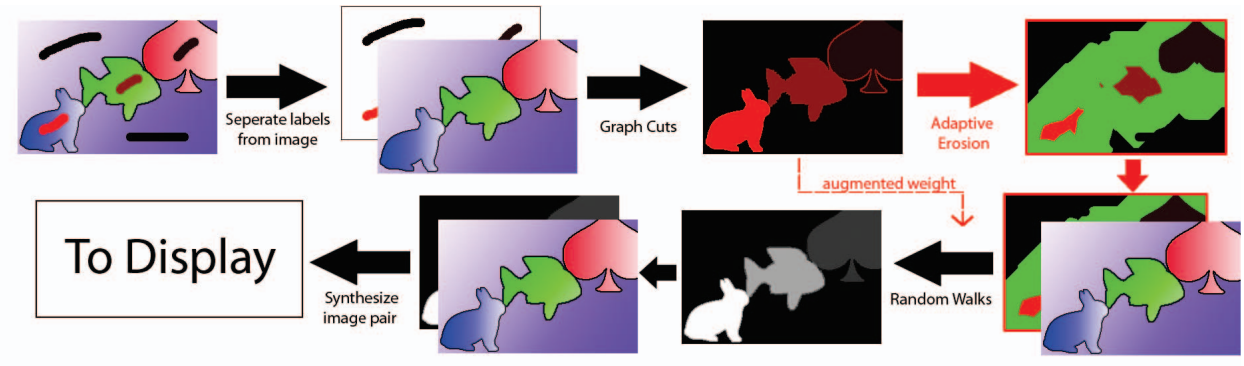


Fig. 1: Adaptive 2D to 3D conversion system with the modifications to the previous system highlighted in red

scaled again when synthesizing the stereoscopic image pair, and this was also demonstrated in [5] and [6]. Using the aforementioned, a multi-label Graph Cuts segmentation algorithm is performed on the image. In the previous work, the Random Walks segmentation algorithm is performed in parallel with Graph Cuts, and the depth map generated from Random Walks is combined with Graph Cuts using the geometric mean and a static weighting. What is different in the proposed work is that the output is a depth prior that provides a rough initial estimate to ensure better accuracy, where the differing intensities of red denote the initial depths. After, each group of labels from this estimate are adaptively eroded, and are fed into the Random Walks algorithm with the image data. The result will leave some pixels unclassified, and so these are set as unknown. The erosion and the depth prior are used to provide more information to Random Walks, as decreasing the number of unknowns significantly decreases computational time. Contrary to the previous work, we determine an adaptive way of merging the two depth maps, and thus creating a final depth map that is more coherent in comparison to the original framework. We thus go through each improvement in more detail.

2.1. Using Graph Cuts to generate a depth prior and the Random Walks labeling

We start by using the user-defined labels along with the original image to generate a depth map using Graph Cuts. We follow the same process as [6] in order to get our initial depth map.

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 labeling. Graph Cuts requires that its labels be integer values. Therefore, to combine our depth prior with the original labels we must first convert the integer labels back to the range $[0, 1]$. To do this, we use a lookup table mapping the user-defined depth values to integer values.

2.2. Converting the Graph Cuts generated depth map into an augmented user-defined label map

The motivation for using the Graph Cuts depth map as a label is to provide more information to the Random Walks algorithm in the form of additional seeds which can allow it to better segment objects, and we thus augment what the user originally specified. The Graph Cuts segmentation ultimately provides a label for every single pixel in an image, and inputting this directly as a label for Random Walks would be meaningless as no pixels would require labeling anymore. Therefore, some preprocessing must be done before we can augment the depth map with the original user-defined label.

The goal of the preprocessing is to transform the Graph Cuts depth map into a label map for Random Walks. This is done by adaptively eroding the different labels in the depth map. Erosion is one of the two basic morphological operations, the other being dilation. Erosion of an image A with a structuring element B is defined in Eq. 1 as:

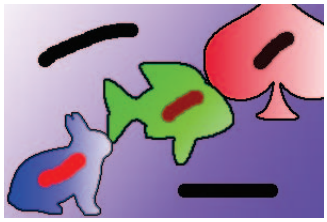
$$A \ominus B = \{z | (B)_z \subseteq A\} \tag{1}$$

Eq. 1 shows that the erosion of A with B is the set of all pixels z such that B, translated by z, is contained in A. For our purpose, we use the 3 x 3 cross structuring element centered at the origin. The amount of times the erosion is performed to a label is adaptively determined by E_n, shown in Eq. 2 below.

$$E_n = \lfloor \alpha(I(\mathcal{L}) + 1) \rfloor \left\lfloor \beta \left(\frac{n_e(\mathcal{L})}{n(\mathcal{L})} + 1 \right) \right\rfloor \left\lfloor \gamma \left(\frac{n_e(\mathcal{L})}{n_i} + 1 \right) \right\rfloor \tag{2}$$

E_n determines how many times the Graph Cuts depth prior is recursively eroded to generate the augmented label map. E_n is composed of three different factors, all of which are expected to contribute some amount of erosion hence we add one to each factor. The first factor takes into consideration the assigned depth of the label being eroded. $I(\mathcal{L})$ is the depth value assigned to the label, normalized between 0 and 1. The higher the value, the more erosion is performed. This is so that the smoothing effect of the Random Walks algorithm is allowed to be more prominent in foreground objects. The second factor deals with texture and is based on a simple check of texture using edge information. $n_e(\mathcal{L})$ is the number of pixels in the image after edge detection is done on label $\mathcal{L}$ and $n(\mathcal{L})$ is the total number of pixels currently covered by label $\mathcal{L}$. The larger the ratio, the more edges were present within the object. Labels for objects with more texture are eroded more. The final factor deals with the size of the objects within the label $\mathcal{L}$, n_i is the total number of pixels in the image. Labels covering more pixels are eroded more. The constants α, β, and γ were chosen experimentally as 5, 1 and 5 respectively. The results of applying Eq. 2 to a set of labels can be seen in Fig. 2.

In Fig. 2(a), the original user-defined labeling is shown, with the style of labeling previously mentioned. The points that are untouched are considered to be unknown, and the system will thus assign depths to these points. In Fig. 2(b), the depth prior generated from the Graph Cuts algorithm is shown. Finally, in Fig. 2(c), the



(a) Labelled Image

Depth Prior generated by Graph Cuts on the left (b) and the Augmented Label Map using Adaptive Erosion on the right (c)

Fig. 2: Augmented Labelling Example

(a) Labelled "Bowling2" Image (from the Middlebury Stereo Database)

Depth Maps: From [6] on the left (b) and our proposed on the right (c)

Fig. 3: Depth map generation example using Bowling2

labels that are to be used for the Random Walks algorithm are illustrated, after applying the adaptive erosion. Here, the same methodology of labeling with respect to the Figs. 2(a) and (b) are used, and green represents the unknown pixels that the algorithm must classify. From Fig. 2, the number of unknown pixels has significantly reduced in size, which the Random Walks algorithm can benefit from as previously mentioned. The overhead from the erosion stage is offset by the speedup gained from the faster Random Walks.

2.3. Random Walks using Graph Cuts augmented label map and modified edge weights

Once the label from the previous stage is generated, we augment the initial user-defined label 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. This new label is used as the input to the Random Walks stage. In [6], they described the method through which Random Walks is used. In this work, we make a modification to the edge weighting. Our weighting function is defined as:

$$d(\vec{c}_i, \vec{c}_j, d_i, d_j|\alpha) = \sqrt{d(\vec{c}_i, \vec{c}_j)^2 + (\alpha(d_i - d_j))^2} \ , \qquad (3)$$

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 d_i and 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 [6], this weight was static, and was empirically determined depending on the image that was used. To circumvent this situation, in this work, we determine α 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, we apply a Sobel edge detector to the grayscale counterpart of the image, and α is generated for each set pixel i using the following formula:

$$\alpha = 1 - \exp(-I_e(i)) \ , \qquad (4)$$

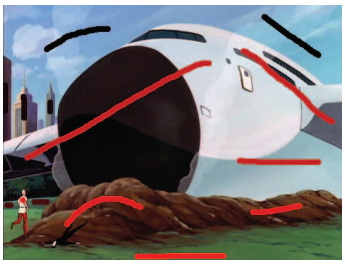
where $I_e(i)$ is the intensity of pixel i in the edge detected image. Fig. 3 shows the differences between a depth map using [6] and a depth map using the adaptive weighting α in Eq. 4. We can see that in Fig. 3(b), the bowling ball, which previously had a portion of it classified as background, is now more uniformly labeled with a single label when observing the same area in Fig. 3(c). Furthermore, the various objects' depth gradients are not as flat. This is important when using depth maps to generate stereoscopic image pairs as objects with purely uniform depth values appear to exist in a single plane when viewed and we get what is known as a "cardboard" effect, where all objects are flat and simply layered at different distances from the foreground as opposed to having volumetric depth.

3. EXPERIMENTAL RESULTS

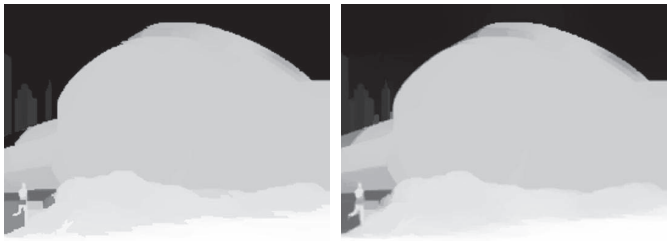
We now present some results which compare the depth maps produced using our modified method and [6]. For full resolution images and depth maps, an online repository has been established at `http://www.rnet.ryerson.ca/%7Emfawaz/icassp2012`.

Fig. 4 demonstrates a sample result using a frame from an animated "Superman" movie. The image used is a single 536 x 404 frame taken at a random point in the clip. We use this image to demonstrate performance with basic cartoon images. Fig. 4(a) illustrates the labelled frame. Fig. 4(c) shows the results obtained using our adaptive method, compared to Fig. 4(b), illustrating the result done by the method in [6]. Here, the edges are still preserved, as well as the detail within the airplane is still maintained. Furthermore, the left side of the airplane wing, which before was misclassified as background, is more accurately represented by a brighter label. The gradual depth in the foreground is also better captured using our new method.

Fig. 5 illustrates the results in the same style as Fig. 4, where the original image was obtained from the Middlebury Stereo Database (`http://vision.middlebury.edu/stereo/data/scenes2006/`) called "Monopoly". The differences in the previous method in comparison to the proposed method are clear. The Monopoly board is

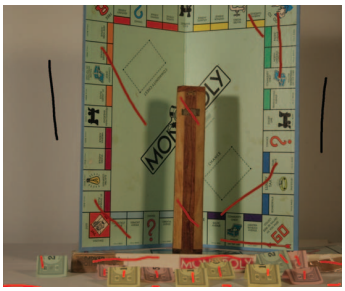


(a) Labelled "Superman" Image



Depth Maps: Using [6] on the left (b) and our proposed on the right (c)

Fig. 4: Depth map generation example using Superman



(a) Labelled "Monopoly" Image (from the Middlebury Stereo Database)

Depth Maps: Using [6] on the left (b) and our proposed on the right (c)

Fig. 5: Depth map generation example using Monopoly

now more uniformly labeled with a single depth and the gradient leading towards the wall on the left is more pronounced. None of the parameters were changed for this image compared to any of our other tests and it is evident that even when dealing with real life scenes, our method can still produce good quality depth maps.

4. DISCUSSION AND FUTURE WORK

The results demonstrate the ability to create good quality depth maps using our adaptive Graph Cuts and Random Walks method. Using the augmented labels along with Random Walks ensures the boundaries of the different objects are respected. Adding information from Graph Cuts into the Random Walks weighting helped clean up edges, and maintaining smooth gradients in Random Walks. Despite the differences in animated images versus real ones, we can see that this method can handle both cases without the need for changing any parameters. This makes our method adaptable to many applications. With regards to future investigations, adding user feedback is being considered. This would allow the user to observe the final depth map and provide further labeling or other information, inevitably allowing improvements. There is ongoing investigation into how this can be applied in 2D to 3D video conversion as well. Current problems are related to adapting the depth map across frames without the need to regenerate a new result each time which involves research into object motion tracking.

5. CONCLUSION

The goal of this work was to expand on the work in [6] adaptively combine the Graph Cuts and Random Walks algorithms in order to produce good quality depth maps. We presented a two stage modification to the existing system, showing consistent results for a wide range of 2D images, including real life scenes and animated scenes, without the need for parameter changing. One venue of investigation we are pursuing is with color edge detection mechanisms to replace the Sobel detector employed. Performing edge detection on grayscale counterparts ignores the correlation between the color channels, and so information that would inevitably aid in the detection process is lost. This could be extended to better texture detection mechanisms as well. Finally, we are investigating adding user feedback, which could ultimately be applied to 2D to 3D video conversion.

6. REFERENCES

[1] W. J. Tam, C. Vazquez, and F. Speranza, "Three-dimensional TV: A Novel Method for Generating Surrogate Depth Maps using Colour Information," *Proc. SPIE Electronic Imaging - Stereoscopic Displays and Applications XX*, 2009.

[2] C-. C. Cheng, C-.T. Li, and L-.G. Chen, "A 2D-to-3D Conversion System using Edge Information," *Proc. IEEE ICCE*, 2009.

[3] J-.I. Jung and Y-.S. Ho, "Depth Map Estimation from Single-View Image using Object Classification based on Bayesian Learning," *Proc. IEEE 3DTVCON*, 2010.

[4] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning Depth from Single Monocular Images," *Neural Information Processing Systems*, vol. 18, 2005.

[5] M. Guttmann, L. Wolf, and D. Cohen-Or, "Semi-automatic Stereo Extraction from Video Footage," *Proc. IEEE ICCV*, 2009.

[6] R. Phan, R. Rzeszutek, and D. Androutsos, "Semi-Automatic 2D to 3D Image Conversion using a Hybrid Random Walks and Graph Cuts based Approach," *Proc. IEEE ICASSP*, 2011.

[7] Y. Boykov and G. Funka-Lea, "Graph Cuts and Efficient N-D Image Segmentation," *Intl. Journal of Comp. Vis.*, vol. 2, no. 70, pp. 109–131, 2006.

[8] L. Grady, "Random Walks for Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–1783, 2006.