# CS231A Course Project Proposal
# Fully automated image matting with Kinect

Karen Cheng
kycheng@gmail.com

Buu-Minh Ta
bmta@stanford.edu

## Abstract

## Future Distribution Permission

## 1. Introduction

Digital matting is a process that extracts the foreground from the background. Bayesian algorithm is a commonly-used method that performs matting by expressing each pixel value as a function of foreground color, background color, and the opacity of the foreground object. This approach requires a trimap to provide an initial guess for foreground, background, and unknown regions. The limitation of this approach is that the trimap is often generated manually, and would be highly inefficient if one were to implement matting on a video stream.

This problem could be circumvented with depth map from a 3D camera – the Kinect, for example. In this paper, we propose different methods that we will evaluate to compute trimaps with the Kinect. The simplest approach applies a threshold to the depth values in order to differentiate the foreground from the background. The second approach applies segmentation to the depth map. The segmentation methods we consider applying are K-means and Ncuts. In this paper, we will demonstrate that depth data could provide us with a more robust method to automate the generation of trimap.

## 2. Data collection

The data will be collected using a Kinect. Our data collection will consist of images that are easy to interpret and images that are more complex. An easy-to-interpret image would have a uniform background and a well-defined foreground, whereas a complex image would have a sophisticated background and a fuzzy foreground.

### 2.1. Calibrating the Kinect

Before the mapping of depth values to RGB pixels take place, the cameras were calibrated to obtain the intrinsic and extrinsic parameters. The intrinsic parameters define the optical and geometrical properties of both cameras. The extrinsic parameters are what define the relationship between the camera coordinates. Calibration is done based on Zhengyou Zhangs flexible calibration method implemented in OpenCV, utilizing images of a checkerboard pattern at 8 different orientations. To increase OpenCVs cvFindChessboardCorners accuracy in locating the corners, the checkerboard pattern was uniformly illuminated by a floodlight, and the IR projector was covered to block the speckles it projects to the scene. The results obtained were comparable to findings from an online source [4].

### 2.2. Correspondence between RGB and depth

We determine the correspondence between depth and RGB pixels by the first transforming 2D image points from the depth camera, $[u, v]^T$, to 3D depth coordinate points $[X_c, Y_c, Z_c]^T$ :

The 3D depth points then undergo rotation and translation to transform into RGB space, and are then projected onto the RGB image plane. Finally, the depth value at $[u, v]_{RGB}$ are matched with the RGB values at $[u, v]_{RGB}$. Depth values are obtained by converting the raw 11-bit disparity values to distance values in centimeters.

In a case where the foreground can be easily separated from the background, as on figure 1, we can see that the correspondence depth is not correct. We can see that the shape of the extracted foreground (using k-means, as explained in next section) is correct, but is not at the correct location…This is an issue we have to inspect further.

## 3. Algorithms

### 3.1. Naive approach: simple threshold

The easiest approach is to use a threshold on depth to separate the foreground from the background. However, scenes can be very different, there can be a great distance between foreground and background, or they can be close. Another problem is what happens if the depth of the background is continuous and overlaps the depth of the foreground. This can be the case if the picture is taken in a angle, and the background is the two walls of the angle. This shows that this simple approach can't really work, and is too simple.

### 3.2. Segmentation with k-means

Another approach is to segment our points into two clusters, depending on their depth. This solves the first issue of the threshold, because now the threshold is apdative and depends on the data. Figure 1 shows a result where the foreground is easy to separate from the background, as their depth don't overlap. However, if that's the case, then we have the same problem.

What we can do is to project the clusters onto the image space. In the problematic situation described above, the cluster corresponding to the foreground would be discontinuous, and separated by the background. We can assume that the foreground is at the center of the image, and that it is continuous. That way, we can correctly adjust the foreground cluster by taking only the part that satisfy those conditions.

Once we have 2 clusters, we have to build the trimap, and thus define the confidence we have in each of the clusters. K-means doesn't really allow us to do that, and a simple way would be to define a border of incertitude on the border of the two clusters. This is kind of arbitraty, and probably not very precise.

### 3.3. Segmentation with N-cuts

K-means has the advantage of being simple to implement, and fast. However, it might not be perfect for our problem, since we can't directly treat clusters in image space, because basically, the distance that matters is the distance to the center of the clusters. N-cuts has a more local property, as the disparity is set between neighbors. Furthermore, N-cuts can give us the level of confidence we have in the clusters, at the discretization step. That way, we can have the unknown zone for the trimap.

We could also combine a segmentation on the color and a segmentation on the depth, and set the unknown zone where they contradict each other.

## 4. Status

We have collected some data with the Kinect, and calibrated it to get the intrinsics parameters. However, the correspondence is not correct, so we have to investigate about this. We thought the Kinect would give us an easy and good correspondence, but it doesn't seem to be the case, so we had to work, and keep working on this part.

The segmentation has been done on the depth to get 2 clusters. On simple cases, where the foreground is well separated from the background, this works perfectly. We have to implement and test the idea where they overlap.

We are investigating some n-cuts implementation to understand exactly how we can modify them for our purpose.

We have implemented the image matting from the trimap following the algorithm presented in [1]. We are still facing some issues, because the paper doesn't present all the details, and probably because of the initial conditions.

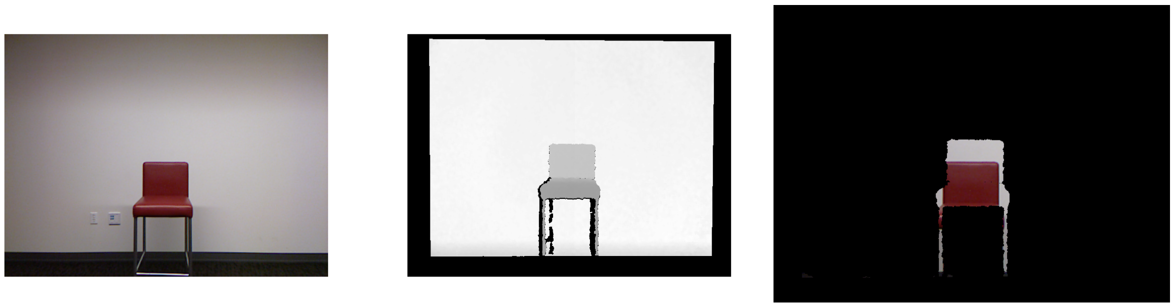

Figure 1. Foreground and background are easy to separate. From left to right: RGB picture, corresponding depth, foreground extracted using depth with kmeans

## References

[1] A Bayesian approach to digital matting, B. Curless, D. Salesin, R. Szeliski

[2] Video matting from depth maps, J. Finger, O. Wang

[3] A closed-form solution to natural image matting, A. Levin, D. Lischinski, Y. Weiss

[4] `http://nicolas.burrus.name/index.php/Research/KinectCalibration`

[5] `http://openkinect.org/wiki/Imaging_Information`