iModel: Object of Interest 3D Modeling on a Mobile Device

A Design Project Report

Presented to the School of Electrical and Computer Engineering of

Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering, Electrical and Computer Engineering

Submitted by

ShaoYou, Hsu

Haochen, Liu

MEng Field Advisor: Tsuhan Chen

Degree Date: May, 2012

Abstract

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

Project Title: iModel - Object of Interest 3D Modeling on a Mobile Device

Author: ShaoYou, Hsu
Haochen, Liu
Mentor: Adarsh Kowdle

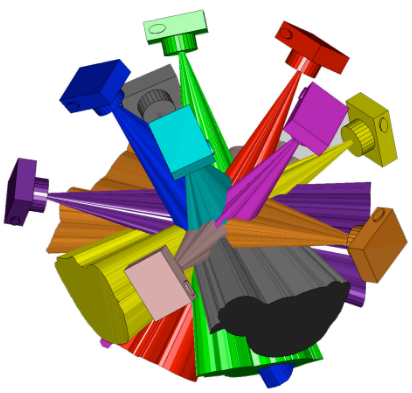
Abstract:

With the advent of 3D technology, 3D content creation is a very popular topic with numerous applications such as augmented reality, gaming, etc. Traditionally, a studio setup with monotonous background or expensive laser scanners are used to obtain the 3D model, which would not work in practice. With mobile devices with cameras available with users, we wish to solve this using an interactive 3D modeling approach via a video of the object captured in its natural environment.

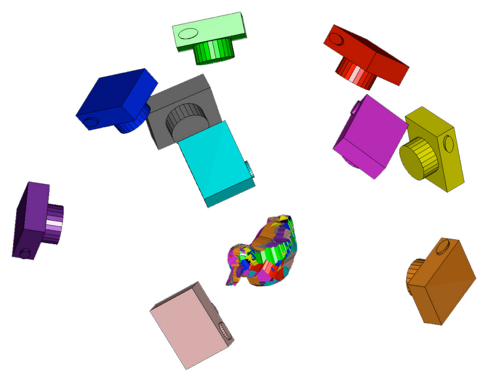
This project is to develop a new-edition app on iPhone and iPad based on the old one to deal with 3D model reconstruction problem using computer vision techniques. This app will use new computer vision techniques to make people restore a 3D model of an object in iPad/iPhone conveniently.

Executive Summary

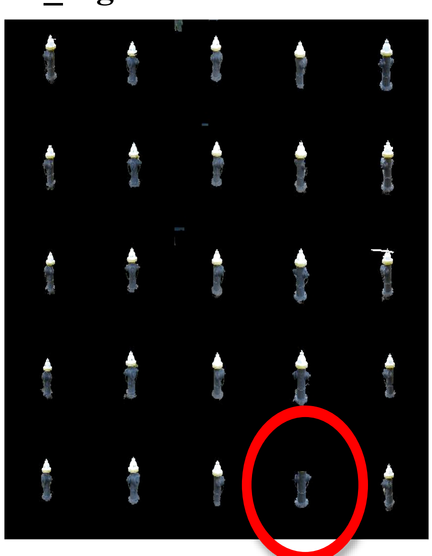
Since we already have the design of segmenting the object from the background in pictures, we work most of time in designing the process of building a 3D model from a video and improving the algorithm, which we use to build the 3D model after we get all the segments from the object of interest. The original algorithm of building 3D model is described as ES_Fig.1 and ES_Fig.2, which show that the model is reconstructed by every 2D segments of the object taken by cameras in different directions. The way to combine those 2D segments is to intersect the projection of these segments with camera parameters(coordinates, focal length…etc) derived by other algorithm called Bundler. The problem happens when there is a segment which loses some information like ES_Fig.3, then we will have a wrong result like ES_Fig.4 which doesn't have the missing part lost in the 2D segments. After amending the algorithm, we successfully derive the correct result like ES_Fig.5 which regain the white head chopped in the 2D segments.
If we choose the proper parameter in our modified algorithm, then we can improve the quality of the result, which even gets the steel chain like ES_Fig.6.



ES_Fig.1



ES_Fig.2



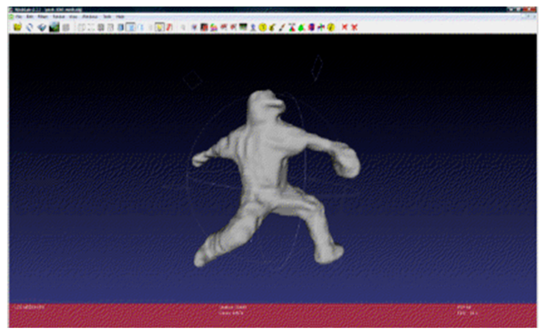
ES_Fig.3

Fig.6

Overview of design and its implementation:

With the advent of 3D technology, 3D content creation is a very popular topic with numerous applications such as augmented reality, gaming, etc. Traditionally, a studio setup as in DP_Fig. 1 or expensive laser scanners are used to obtain the 3D model, which would not work in practice (DP_Fig. 3 and DP_Fig. 4).
With mobile devices with cameras available with users, we wish to solve this using an interactive 3D modeling approach via a video of the object captured in its natural environment.

OD_Fig.1



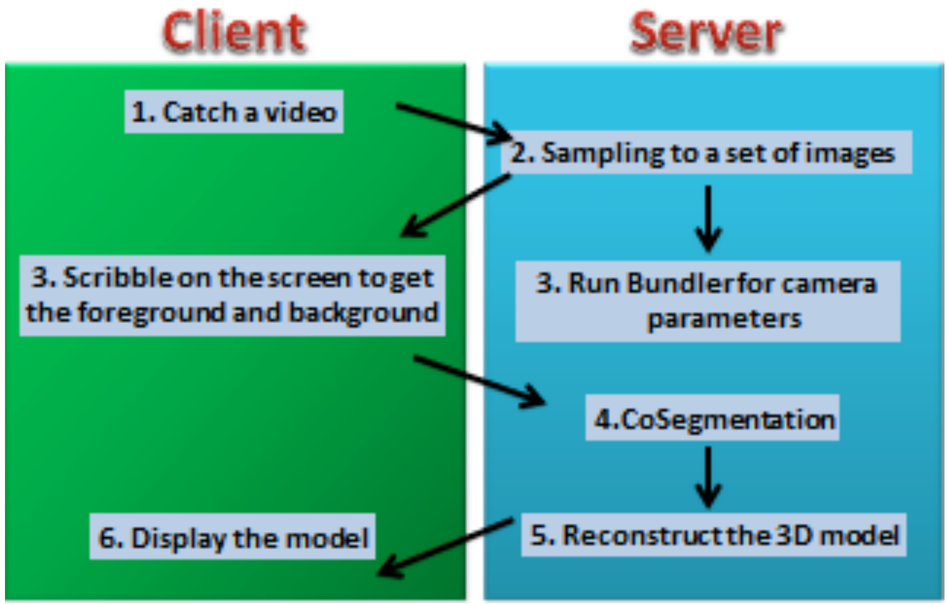
OD _Fig.2(3D model of OD _Fig.1)

OD _Fig.3

OD _Fig.4

This project is to develop a new-edition app on iPhone and iPad based on the old one to deal with 3D model reconstruction problem using computer vision techniques. Shaoyou Hsu will deal with the computer vision techniques to recognize the patterns in video and generate 3D model; Haochen Liu will develop the iOS programming part to make interface of computer vision algorithms and edit 3D model as user needs. We plan to develop a new edition of app with video recording and editing 3D model functions added. This app will use new computer vision techniques to make people restore a 3D model of an object in iPad/iPhone conveniently.

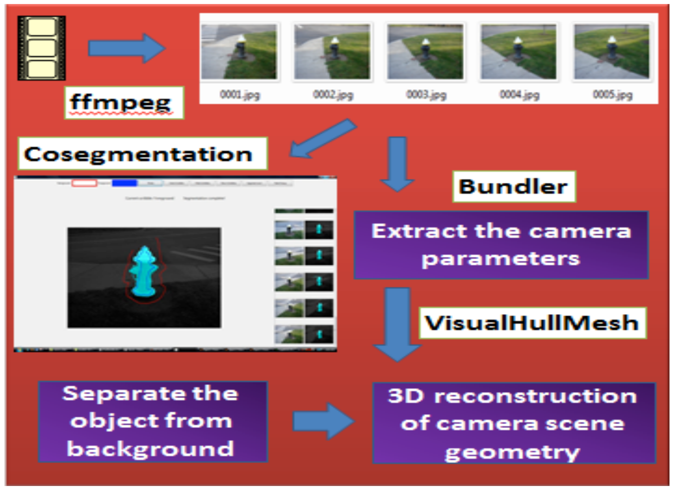
We deign the flow chart of the process executed by our server and client parts as in OD_Fig.5.



OD_Fig.5

The goal is to build an app for the client to execute the system and get the 3D model by their iPhone. In the client part in the flow chart, we plan to let the users take a video around the object they want, and then run our app. After about one minute, the user will receive 8 frames of the video, and then they can scribble on the screen to separate the object and the background by using the program called iCoseg in the server part, which is developed in our lab before. Finally, the user can get the 3D model of the object of interest in 5 minutes from the server and they move or rotate the 3D model as they want.

All the tasks we do can be split into two main structures, which are server part and client part.
In the server part:



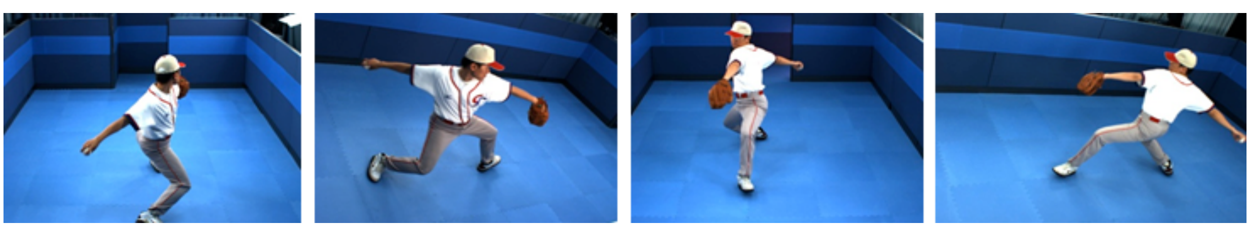
OD_Fig.6

The server part will handle several tasks. The first one is to sample 40 to 50 frames from the video sent by the user. After deriving the frames, the server can extract the camera parameters by using Bundler, which is a program based on structure- from-motion and send 8 frames back to the user to separate the object from the background. The final task is to build the 3D model based on the shape-from-silhouette combining the camera parameters obtained from structure-from-motion and send the result back to the user.

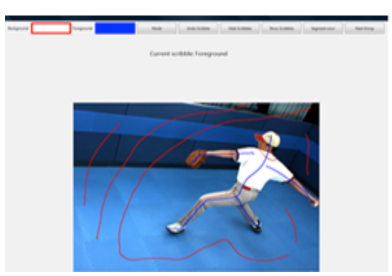
The server part is based on an existing tool, called" Interactive Co-segmentation", which is the first step towards enabling users to create 3D models of an object of interest. The result will be made use to do the construction of 3D model.

For example, we have several frames of a video and choose 4 of them as a sample (OD_Fig.7).

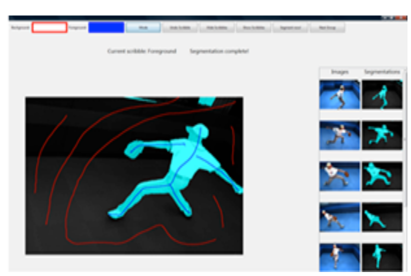
The interactive algorithm uses the scribbles (as red and blue lines shown in OD_Fig.8) from the user indicating foreground and background (red lines are for background and blues lines are for foreground) to extract silhouettes of the objects of interest from multiple views. (OD_Fig.9)



OD_Fig.7



OD_Fig.8



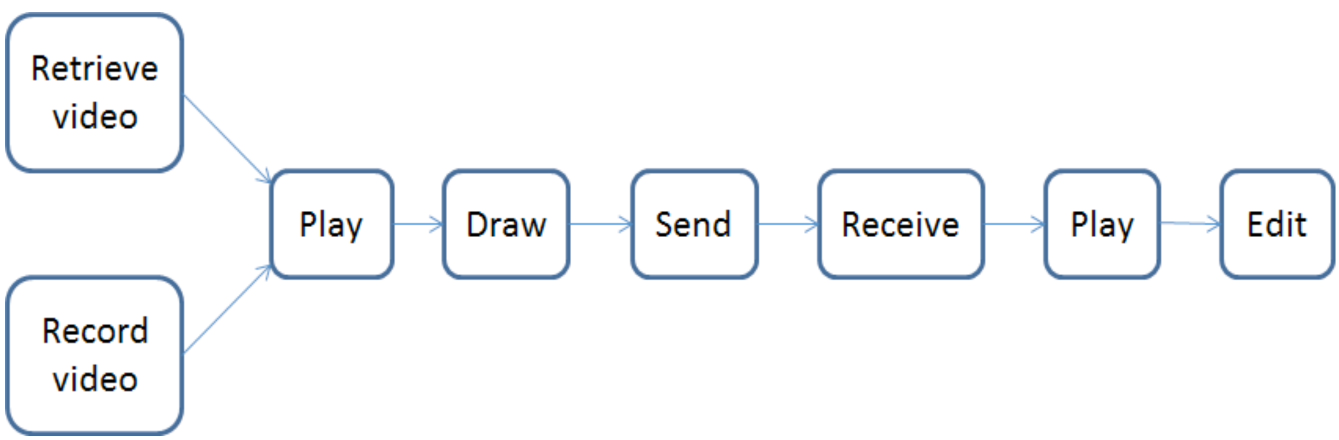
OD_Fig.9

We call it “Co”Segmentation because the user can just scribble on three or four of total frames to separate the object they want instead of doing this on every frames.

In the client part:
In the front-end, we want to build an interface based on iOS. The idea is that while the cellphone-cameras are prevalent, a more accessible approach is to capture images of the object just by these devices. Once we have the interface, we can receive the images as an input from the users, and then transferring to the server, which will do all the algorithms mentioned above, to get the 3D model as an output. Now, server can send the result to the interface and show it on the screen. When the 3D model is derived successfully, that means users can make any of their own things into 3D models and personalization in the virtual world is then feasible.

Senario of the iOS app including:

1. To retrieve a video or record a new video
2. Play this video in this app, find the images to be drawn
3. Draw lines on these images
4. Save these lines to files and then send the video and files to server
5. Receive 3D model file from the server
6. Decode this 3D model file, play this on device
7. Get image from 3D model in a certain angle, put this image onto other images if user requires



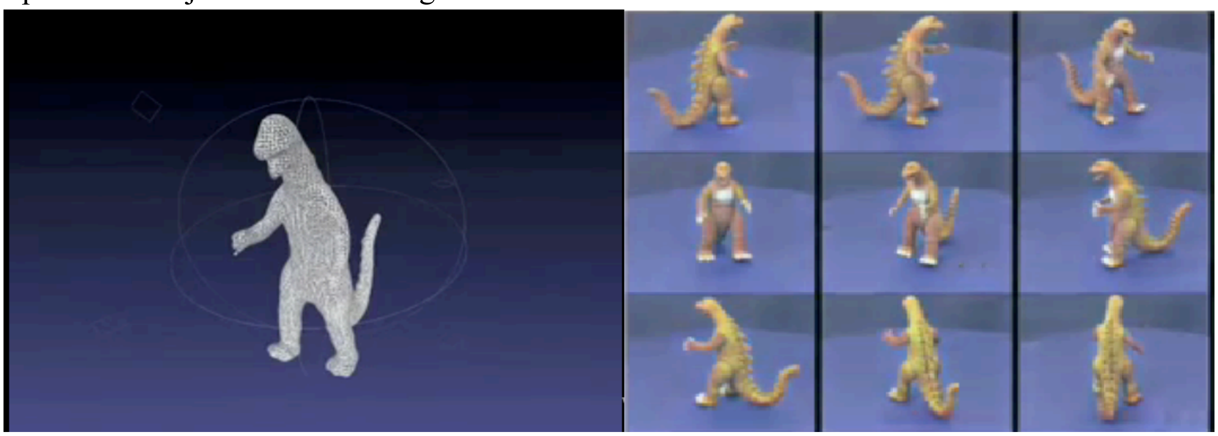
The final accomplishment for this project is the iModel app on iPhone. In order to use this iModel app, the user should take a video around the object first. Then choose the video and send it to the server. The server side contains several php files. After the server receives the video, the server will sample a set of images from the video to get the different views of the object. And then send them back to the iOS device.

After the iOS device receives the images from the server, the user may choose one of them and then scribble on it. It's not necessary to scribble on all of them, so that it is convenient for the user to use.

Then the user can press the "send" button to send the scribbles data to the server. The server runs the Bundler and CoSegmentation to generate a 3D model file(.obj type) under the specific folder. The iOS device can access the 3D model file and then display this 3D model on an iOS device.

Design problem and the solutions:

In the previous work, we take pictures around the object of interest, which is put into the place with simple color in the background as in DP_Fig.1. Now the goal is to take the picture in the natural environment, which has more features in the background. The problem is after we use complicated background in the pictures, it's not easy to do the segmentation when we want to separate the object from the background.

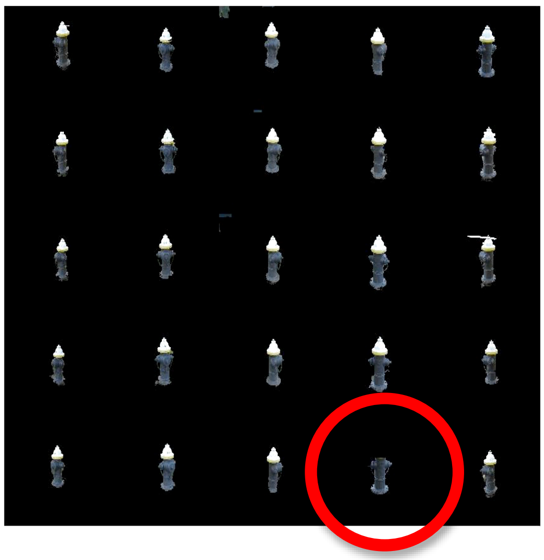


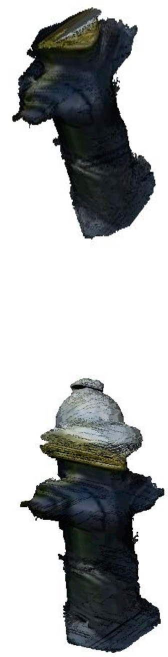
DP_Fig.1

Because some colors will appear frequently in the background as well, we have to make a clean segment by repeatedly doing the algorithm for segmenting many times and this will make the user bored of using our app. Another problem is that since we use the different algorithm for complicated backgrounds, we need the backgrounds of the object of interest always be complicated, or our algorithm won't work.

Problems that have been solved in algorithms:

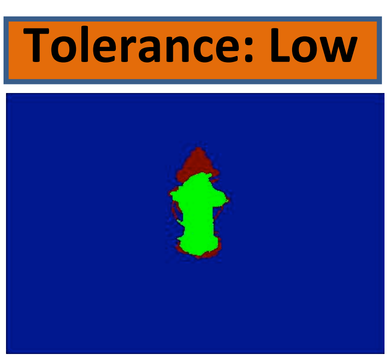
As we need to extract the camera parameter of the images of the object, the algorithm can't make it if each image has too long interval between them. That means every image of the object should be similar and only has difference caused by slight movement of the camera. Hence, we use frames sampled in a short interval from a video to achieve this. Besides, the features of the background have to be abundant to let the algorithm have enough information to track through the set of images and estimate the camera parameters. As for the process of 3D-reconstruction, The problem happens when there is a segment which loses some information like DP_Fig.2, then we will have a wrong result like DP_Fig.3 which doesn't have the missing part lost in the 2D segments. After amending the algorithm, we successfully derive the correct result like DP_Fig.4 which regain the white head chopped in the 2D segments.

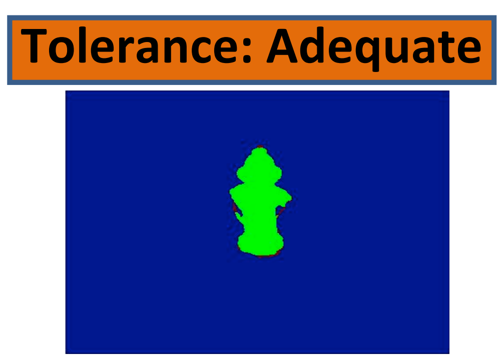


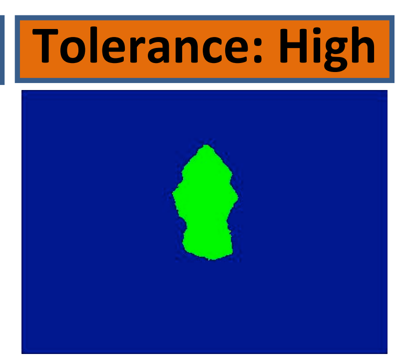


DP_Fig.2 **DP_Fig.4**

To solve this problem, we add an additional parameter called "tolerance". We check every pixel to see how many frames has this pixel, and set the "tolerance" to decide if this pixel should show up in the result or not. For example, we can see in the DP_Fig.2 there is one segment which has his head chopped. If we choose the tolerance to be low as in DP_Filg.5, the missing part is regarded as missing even though the rest segments have it. The result is shown in DP_Fig.8. If we choose an adequate tolerance, then the missing part is added back to the result which is shown in Fig.9. But if we choose the tolerance to be high, the result will have noise which shouldn't appear in the result due to we "tolerate" it to show.







DP_Fig.5 **DP_Fig.6** **DP_Fig.7**

DP_Fig.8

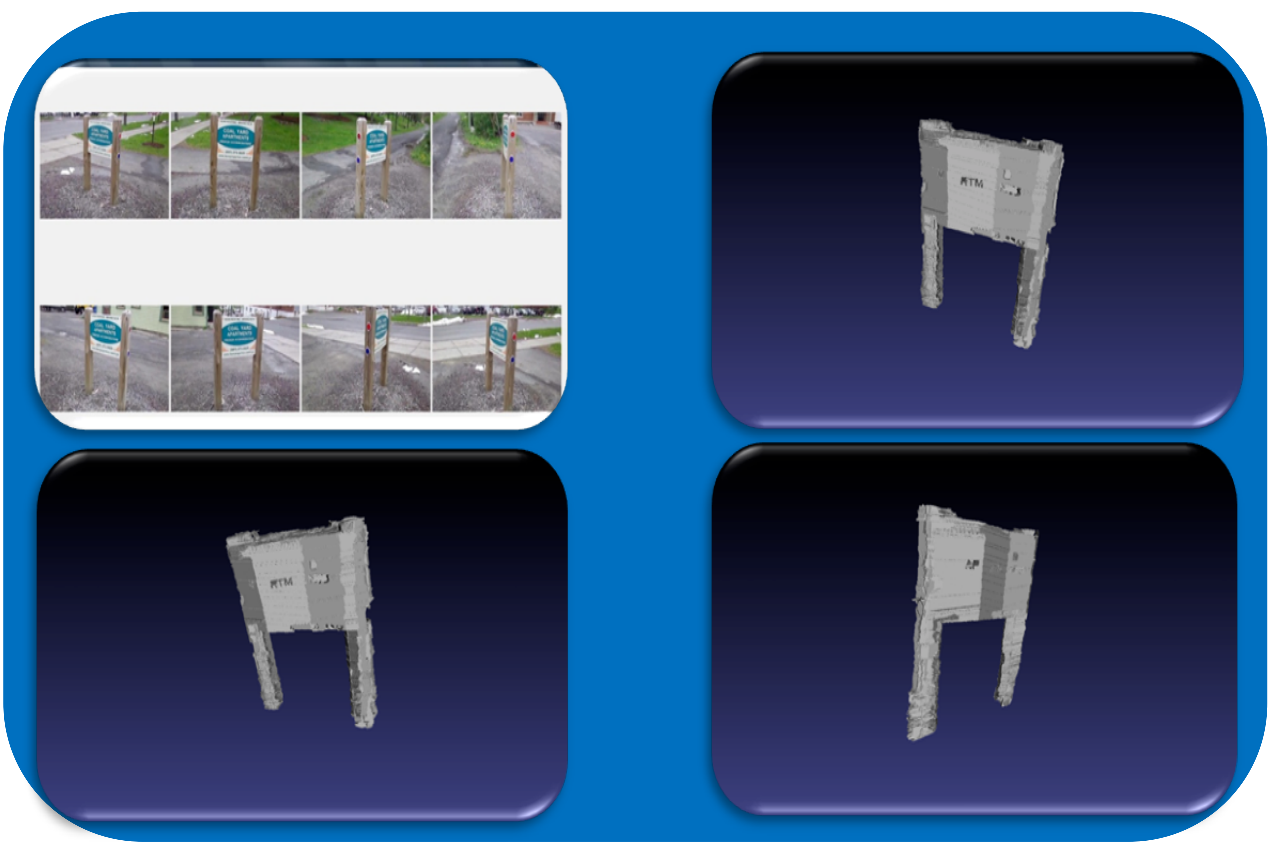
DP_Fig.9

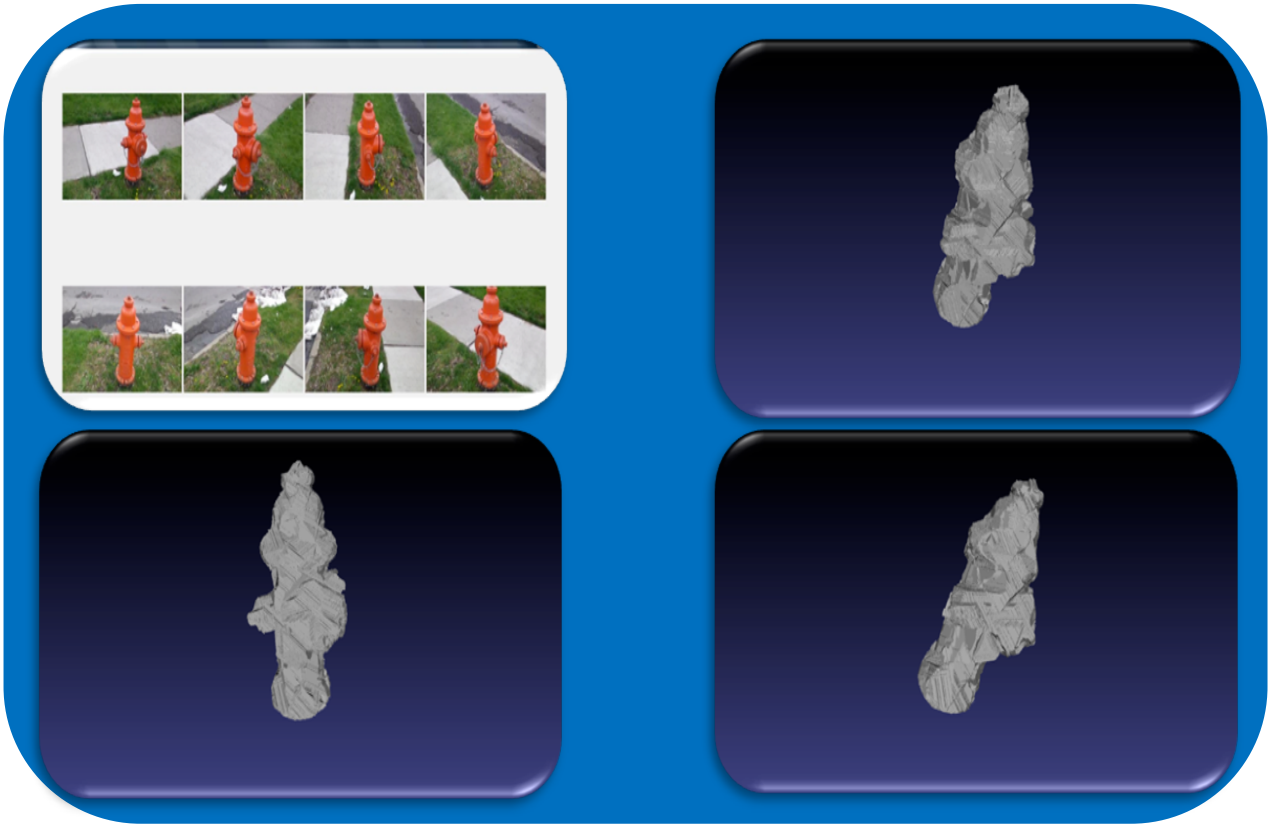
DP_Fig.10

Problems that have been solved in catching images and lines from a video:
Generally, the file format of video recorded by iPhone/iPad is MOV, the image file format of image is JPEG. When we receive the line drawn by a user, we need an image printed on the device. This requires this app should have the ability to retrieve an image from the certain video. And then we implement the "get gesture" part to receive the lines drawn by users, save the lines by recording the coordinates of every point into txt files.

Problems that have been solved in decode 3D model file:
After receiving the 3D model file from server, this app should decode this file to a 3D model presented on the device. This will call for 3D model interface in objective-C programming language. This 3D model could be rotated in any direction and we should also extract the image from it in any angle. The methodology of this part is introduced in "iPhone 3D Programming".

Results:



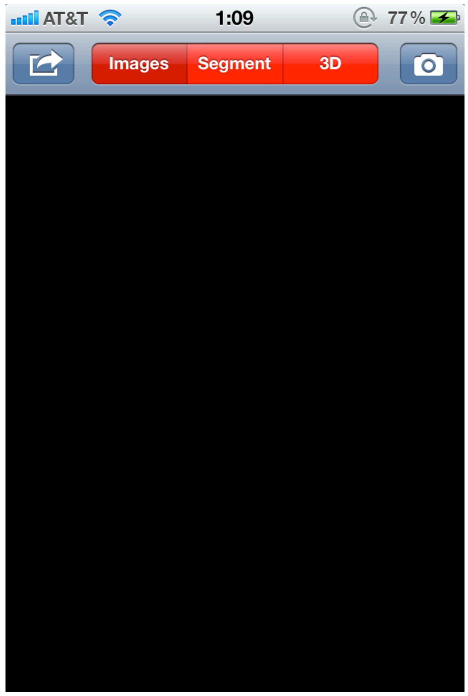


Conclusion

There are plenty of apps in App store about image processing. However, most of these apps provide the function of dragging, cutting, rotation and so on, very few apps have 3D model in them. This app distinguishes with others since we have computer vision algorithms in it, so that we can catch the object 3D model automatically and use this model into the image editing. We will improve our work by applying correspondences or 3D positions not only camera parameters to help obtain better reconstruction.

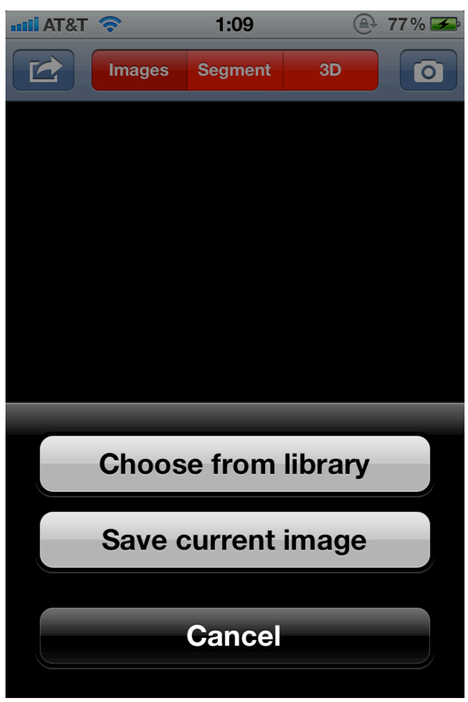
User Manual

When the user firstly opens the iPhone app, the user interface is as below:



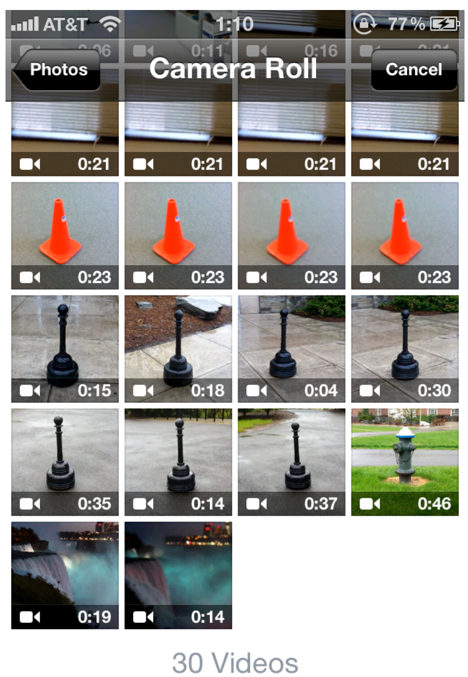
UI Fig 1. The original user interface

Firstly we choose a video from the camera roll:

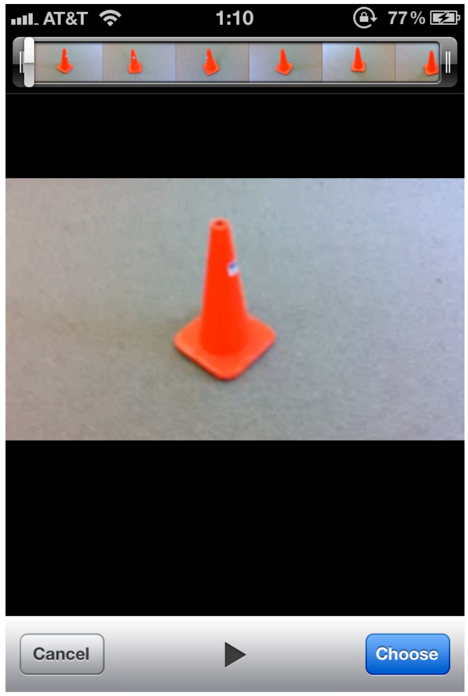


UI Fig 2. User Interface to choose the video

UI Fig 3. User Interface to choose the video

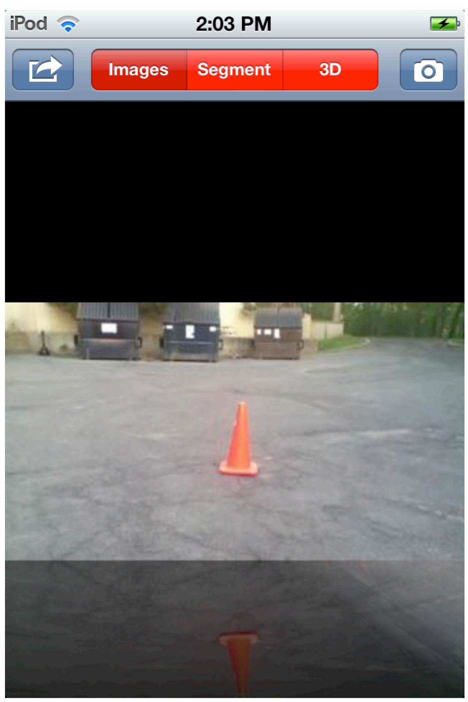


UI Fig 4. User Interface to choose the video

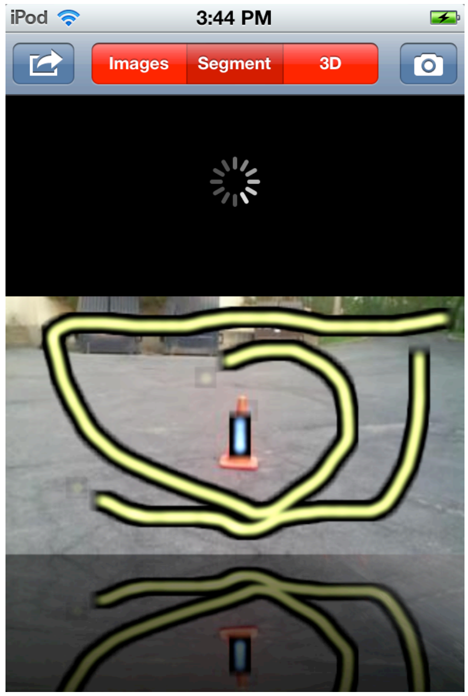


UI Fig 5. User Interface to choose the video

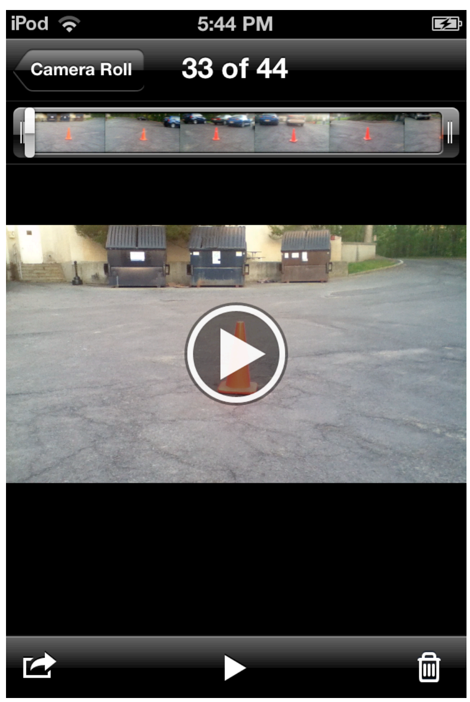
Then we press the button "choose", the app will send this video to server and get the sampled images back to the iOS device.



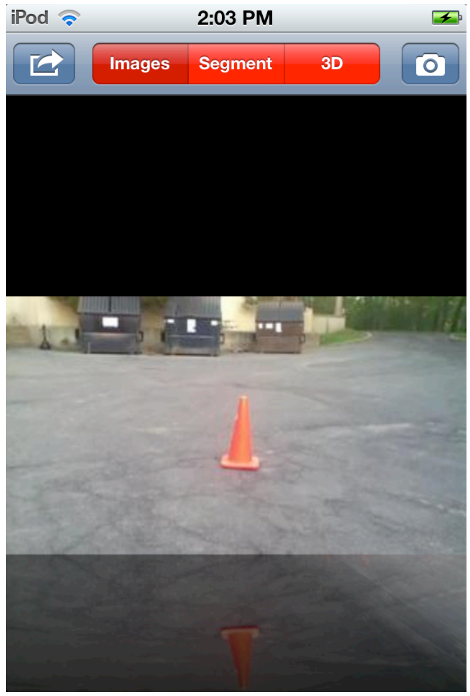
UI Fig 6. The image sampled by the server



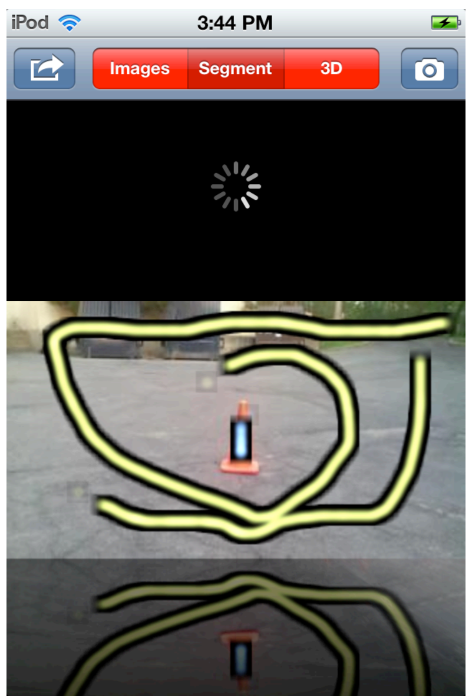
UI Fig 7. The scribble data on images by the user



UI Fig 8. Choose the video from the camera roll

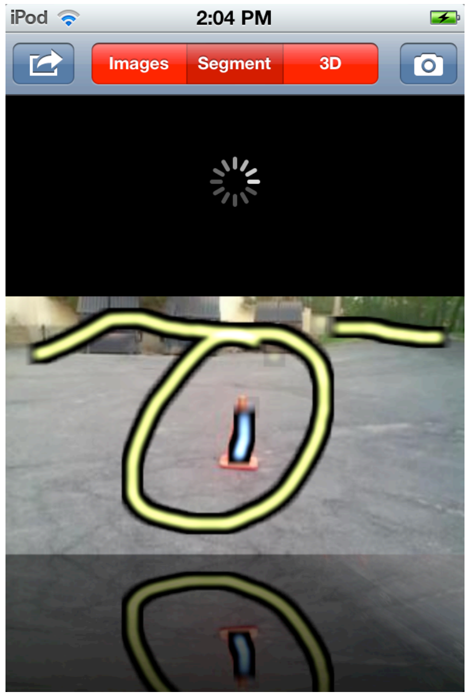


UI Fig 9. the image sampled by the server

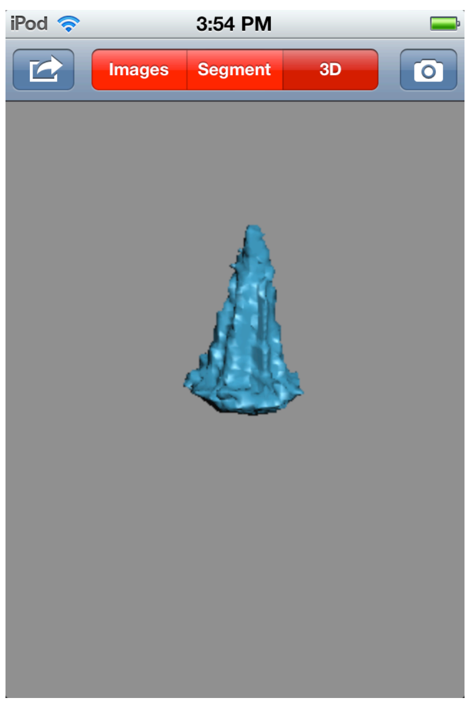


UI Fig 10 . the scribble data on images by the user

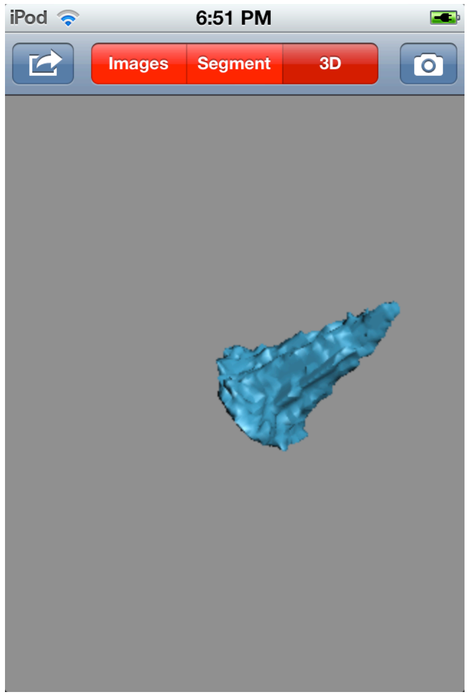
Then the user scribbles on it and press the top left corner button. The scribble datum will be sent to server and the iOS device get the 3D file.



UI Fig 11. the scribble data on images by the user



UI Fig 12. the 3D model displayed on iOS device



UI Fig 13. the 3D model displayed on iOS device

UI Fig 14. the 3D model displayed on iOS device

Reference:

"**iModel: Interactive Co-segmentation for Object of Interest 3D Modeling** ", Adarsh Kowdle, Dhruv Batra, Wen-Chao Chen and Tsuhan Chen. Workshop on Reconstruction and Modeling of Large-Scale 3D Virtual Environments, European Conference on Computer Vision, 2010 (ECCV '10).

"**Calibration, Recognition, and Shape from Silhouettes of Stones** " Keith Forbes, 2007