# 3D Reconstruction from Multiple 2D Images

Chinasa T. Okolo

CSCI 181P: Image Processing

Pomona College

Fall 2016

## INTRODUCTION

This project explores the topic of 3D image reconstruction from multiple 2D images. The type of 3D reconstruction that will be implemented in this project is called Multiview 3D image reconstruction. The process of Multiview 3D image reconstruction involves taking multiple real-world images from any angle and matching points of subjects and backgrounds across images to form 3D models, representing them in a new image. The goal is to do so without loss of quality or information and keep coloring consistent, even when coloring, shading, lighting, and exposure is not kept consistent across the images. Because this process has been approached previously, our goal is to explore what has worked, what has not, and how it can be achieved in the most efficient way, with minimum loss of information. Algorithms developed in our referenced published papers will make our implementation much easier. We will use these approaches and a set of real-world images, determine exactly which code to implement on these images, and then analyze the results of this implementation. These results will tell us what effect our project has, how well it works, and what can be done moving forward to improve this project.

## PREVIOUS WORK

The reconstruction of 3D images from multiple images involves matching points within the multiple views to ensure that the 3D image accurately represents the original images. An important part of this when taking images of subjects is distinguishing the background parts of the image from the subject. The background sections of the images will have different depth than the parts of the image that are much closer, so it is important to differentiate the two in order to match them in the 3D reconstruction. Goldlucke and Magnor describe an algorithm using graph cuts to assign pixels different depth values to use in the reconstruction. They are able to separate

objects from each other and accurately represent the structure in a 3D image. This directly applies to our project because we are precisely trying to use depth perception to match pixels as necessary in our reconstruction. We can implement this technique in our project to ensure that our image contains the correct information that we are trying to model.

As noted previously, there have been many publications and algorithms produced to solve the challenge of reconstructing a 3D image from multiple images. Many of these methods are accurate, but they all have delicate intricacies and variations in methodologies that can be confusing to those not familiar with them. In this paper, Esteban, Dijk, and Groen standardize the process of 3D image reconstruction in five steps: camera calibration, motion estimation, optimization, reconstruction and modeling. Esteban et al have created a toolbox in MATLAB called FIT3D which will allow users to obtain a complete 3D reconstruction from a set of standardized images. FIT3D provides users with many degrees of flexibility and scope allowing researchers and students to use one particular method with real data without the need for extensive additional programming when implementing 3D reconstruction. Esteban et al's paper is relevant to our project because we aim to implement a method for 3D image reconstruction as efficiently as possible. Image processing is a new field for both of us and there are many methodologies that are very abstract and hard to understand. Being able to use the FIT3D Toolbox will provide us with a valuable resource for completing our project, reducing the time needed for researching 3D reconstruction methods and allowing us to spend more time on implementation.

The automatic reconstruction of 3D planar images has many applications in architecture, mapping, and environment analysis. Baillard and Zisserman describe a new method for automatically constructing 3D models with images taken from different planar views. Current

methods of reconstruction are unable to accurately form 3D representations due to various complexities in environments. Their overall approach is to use the lines and their image neighborhoods over multiple views to construct 3D planar images. This method allows line grouping and adds detail to the reconstruction that was previously missed in other methods. In the paper, the authors demonstrate this method by using a dataset of overlapping aerial images to begin line matching. After this step, has been completed, half-planes are computed, then grouping and completion of 3D lines based on these half-planes, and finally the image is created using plane delineation and verification. Baillard and Zisserman's paper is related to our project because we aim to implement 3D reconstruction from multiple 2D images as outlined in this paper.

Over the past few years, multiple algorithms have been created to solve the problem of 3D reconstruction, but there has been no method developed to test the accuracy of these algorithms. Seitz et al provide a collection of high quality, multi-view images to evaluate these algorithms. In this paper, Seitz et al have work with their own dataset that they created using the Stanford spherical gantry, which is a robotic arm that can capture images within a one-meter spherical radius. The images represent a wide variety of characteristics, including sharp and smooth features, strong and weakly textured surfaces, and complex topologies. Within the paper, these images are used to compare the results of six 3D reconstruction algorithms. Seitz et al's paper is relevant to our project because we need a high-quality data set of images to test our code with. It is possible that we could take pictures of our own, but they would probably not be as high quality as the ones provided by this paper. Having access to these images will provide us with a reliable metric to compare our results with those produced in the paper.

# 3D POINT CLOUD REGISTRATION AND STITCHING

In computer vision and image processing, point cloud registration is essential in creating multi-dimensional representations of data. Point clouds are sets of data points usually defined by a three-dimensional coordinate system that represents the external surface of the object modeled in the image. During registration, the overall goal is to determine the relationship between positions and orientations of images acquired in separate viewpoints and/or orientations (PCL). These viewpoints will be compiled in a global coordinate framework where intersecting areas of each respective point cloud will overlap and align with each other. In compiling these viewpoints, multiple transformations such as translation, rotations, and scaling will be used. Viewpoints of cloud point data can be extracted from images that contain multiple dimensions, such as those from 3D, magnetic resonance imaging, or tomography scans. These types of scans measure multiple points on the surface of an object, creating a point cloud that will be used for things such as animation and 3D reconstruction. The goal of this project is to create a platform that will reconstruct 3D models from multiple 2D images, but to do this we have to start from the basics by finding the respective point clouds of each image and then stitch them to build full point cloud maps that accurately depict the depth needed to construct 3D models. We will be exploring the methods of 3D point cloud registration and stitching and structure from motion from two views to find a suitable method of reconstructing 3D models from 2D images of varying viewpoints. The code created and modified in this part of the project will demonstrate multiple cloud point stitching and single cloud registration.

To begin 3D point cloud registration and stitching, it is necessary to employ a point cloud, a surface mesh, cloud registration, and stitching. A point cloud is the collection of x, y, z coordinates of an image resulting from three-dimensional scanners (IGI Global). In the process

of extracting a point cloud, multiple photographs are analyzed to find the corresponding points between each photo to extract a point that matches closest to the actual surface of the object. In order to get the best representation, it is best if the photos used in the process are all taken from different angles. In the three-dimensional coordinate system, these points represent the external, physical surface of the object being modeled. When multiple points clouds are compiled, the user is able to accurately model a 3D reconstruction of the respective object. The information provided by point clouds are not enough to construct three-dimensional models. To do so, it is necessary to convert the compilations of multiple point clouds into polygon mesh models.

A surface mesh is a collection of vertices and connectivity information where each point is a vertex of a triangulated face (Northeastern University). Mesh models contain more information than point clouds, and thus are used to produce three-dimensional reconstructions. To accurately represent the surface of an object, a triangle mesh is needed which will use the x, y, z coordinates produced from the point clouds (Michigan Tech). The process of converting point clouds into triangle mesh models consists of subsampling which distributes the selection of points from the point cloud and reduces computation time, normal reconstruction which produces the vector cross product of two non-parallel edges of the surface points in an object (MathWorld), and finally surface reconstruction which uses the original point cloud with the previously computer normal vector to build a three-dimensional surface (MeshLabStuff). To ensure that the original color of the modeled object is conserved, each respective color attribute of a point should be stored in the point cloud. After these algorithms are computed, the surface mesh construction will be complete.

Cloud registration is the process of consistently aligning various three-dimensional point cloud data views into a complete model (PCL). Registration finds the relative positions and

orientations of each separate photograph and combines the points into a coordinate framework where the intersecting areas between the points overlap as accurately as possible. When multiple point clouds need to be processed, pairwise registration will be used to register these datasets. From the cloud points given, a set of interest points that best represent the scene will be identified. At each of these points, feature descriptors (such as geometric or color) will be computed into a feature vector. Since each point cloud will output a feature vector, it will be necessary to find the corresponding set based on the similarities between features and point positions. Brute force or nearest neighbor methods can be used to find matching points and features. To create the correspondence set of matching points and features, searches can be made by using the points in one cloud and corresponding these points to another cloud or separately computing the correspondences in each cloud then using the intersection of these respective correspondences (PCL). Next, the data should be filtered to remove invalid correspondences and from this information the structure for motion estimate can be computed, this process will be discussed later in the paper.

## METHODS

To obtain the data needed for three-dimensional point cloud registration and stitching, sensors that can capture color along with depth will be influential in this process. These sensors are called RGB-D, and provide color information along with the estimated depth for each pixel captured, producing RGB-D images. As expected, many of these sensors and scanners can come at a steep price, but at a price of $150 Microsoft Kinect has reimagined what can be done with 3D mapping. The Microsoft Kinect is a structured light sensor that uses infrared laser light to project patterns onto the scene being captured and analyze a speckle pattern to create a depth map (Northeastern University). The infrared speckle pattern is reminiscent of salt and pepper

noise and used for comparison with the known depths of the reference patterns stored by the Kinect (Riverside). The depth map is computed from the infrared projector and sensor contained within the camera. This process is called structured light and is the process that creates point clouds. To obtain images, the Kinect combines structured light with two computer vision techniques: depth from focus and depth from stereo. Depth from focus operates on the assumption that an object that appears blurrier is further away and depth from stereo is a process that will extract 3D information from multiple two-dimensional views of a scenes (MathWorks). The main principles of these techniques allow the Kinect to improve the accuracy when capturing multiple objects in an image.

**Detailed Code Analysis.** This code registers point clouds and then stitches together multiple point clouds that have been captured by the Microsoft Kinect and reconstructs a 3D scene using the Iterative Closest Point Algorithm. This code was referenced from the 3-D Point Cloud Registration and Stitching example from MathWorks. Registering point clouds is an extensive process involving advanced techniques and multiple processing steps. The first step in registering two point clouds is to obtain the data from its respective directory. When RGB-D images are captured by the Microsoft Kinect, a point cloud, along with the image itself is outputted. The point cloud is stored in a mat file and consists of vectors of the image depth data. To extract two consecutive point clouds using the first point cloud as a reference to the second, index into the first and second entries of the mat file.

```
ptCloudRef = livingRoomData{1};
ptCloudCurrent = livingRoomData{2};
```

To ensure that registration is completed at the highest quality, preprocessing is needed to remove noise from the data. In this code:

```
gridSize = 0.1;
fixed = pcdownsample(ptCloudRef, 'gridAverage', gridSize);
moving = pcdownsample(ptCloudCurrent, 'gridAverage', gridSize);
```

the data is processed by downsampling with a box grid filter with a size of ten centimeters. Downsampling reduces the sampling rate of the signals in the image data and will remove noisy points that are not relevant to the data.

To align the two point clouds, the Iterative Closest Point Algorithm will estimate the 3D rigid transformation (a transformation that preserves length) on the downsampled data. The first point cloud will be used as a reference for the second point cloud and these two point clouds will be merged, aligning the overlapping points. The following code finds the rigid transformation to align the second point cloud with the first and will then transform the second point cloud using the resulting coordinate system defined by the first point cloud.

```
tform = pcregrigid(moving, fixed, 'Metric','pointToPlane','Extrapolate', true);
ptCloudAligned = pctransform(ptCloudCurrent,tform);
```

To create the world scene, the aligned region of the overlapped points will be filtered with a box filter to downsample this data and each point cloud will be merged to produce a three-dimensional representation of the input images on a x, y, z graph.

```
mergeSize = 0.015;
ptCloudScene = pcmerge(ptCloudRef, ptCloudAligned, mergeSize);
```

```
% Visualize the world scene.
```

```
subplot(2,2,[2,4])
pcshow(ptCloudScene, 'VerticalAxis','Y', 'VerticalAxisDir', 'Down')
title('Initial world scene')
xlabel('X (m)')
ylabel('Y (m)')
zlabel('Z (m)')
drawnow
```

To stitch a sequence of point clouds, which will create a larger 3D scene, repeat the process of registering two point clouds. As before, the first point cloud will be used to establish the reference coordinate system and each point cloud will be transformed to the reference coordinate system. To begin, the transformation object that has accumulated all transformations should be stored in an alternate variable.

```
% Store the transformation object that accumulates the transformation.
accumTform = tform;
```

Then, to account for all the point clouds analyzed thus far iterate through the code and modify the point clouds at each iteration. Like aforementioned methods, the previous point cloud should be continually used as a reference, the Iterative Closest Point Algorithm should be applied, the current point cloud should be transformed based on the reference coordinate system defined by the referenced point cloud, and the world scene encompassing all the point clouds should be updated and visualized.

```
for i = 3:length(livingRoomData)
    ptCloudCurrent = livingRoomData{i};
    %Use previous moving point cloud as reference.
```

```
fixed = moving;
 moving = pcdownsample(ptCloudCurrent, 'gridAverage', gridSize);
 %Apply ICP registration.
tform = pcregrigid(moving, fixed, 'Metric','pointToPlane','Extrapolate', true);
%Transform the current point cloud to the reference coordinate system
 %defined by the first point cloud.
accumTform = affine3d(tform.T * accumTform.T);
ptCloudAligned = pctransform(ptCloudCurrent, accumTform);
% Update the world scene.
ptCloudScene = pcmerge(ptCloudScene, ptCloudAligned, mergeSize);
 %Visualize the world scene.
hScatter.XData = ptCloudScene.Location(:,1);
hScatter.YData = ptCloudScene.Location(:,2);
hScatter.ZData = ptCloudScene.Location(:,3);
hScatter.CData = ptCloudScene.Color;
drawnow('limitrate')
```

In case there were any discrepancies when capturing the scene with the Kinect, an affine transformation can be done to shift the data to a different viewpoint.

```
ptCloudScene = pctransform(ptCloudScene, affine3d(TRANSFORMATION VECTOR));
```

If not, the data can be displayed as before.

```
ptCloudScene = pctransform(ptCloudScene, affine3d(A));
pcshow(ptCloudScene, 'VerticalAxis','Y', 'VerticalAxisDir', 'Down', ...
```

```
        'Parent', hAxes)
title('Updated world scene')
xlabel('X(m)')
ylabel('Y(m)')
zlabel('Z(m)')
```

## Experiments and Results

Image 1. 3D Point Cloud Registration

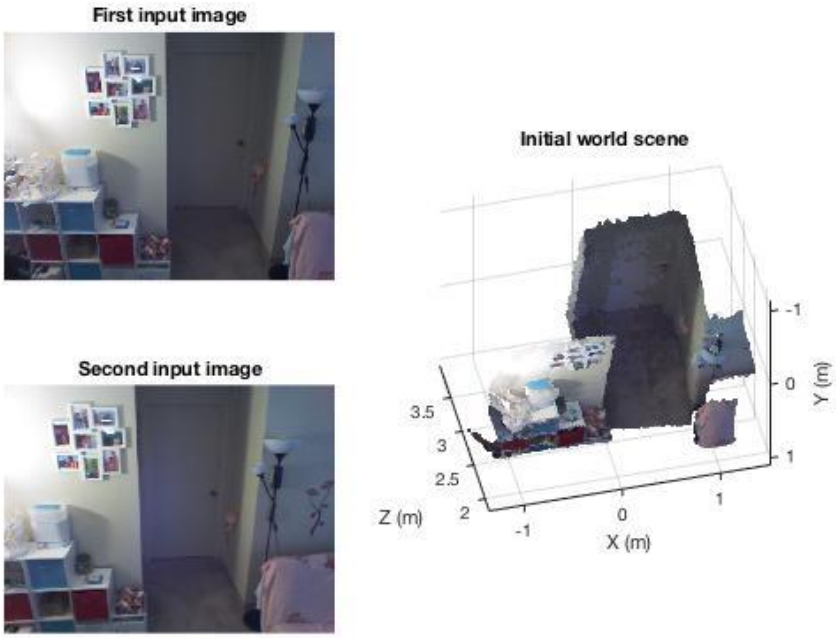
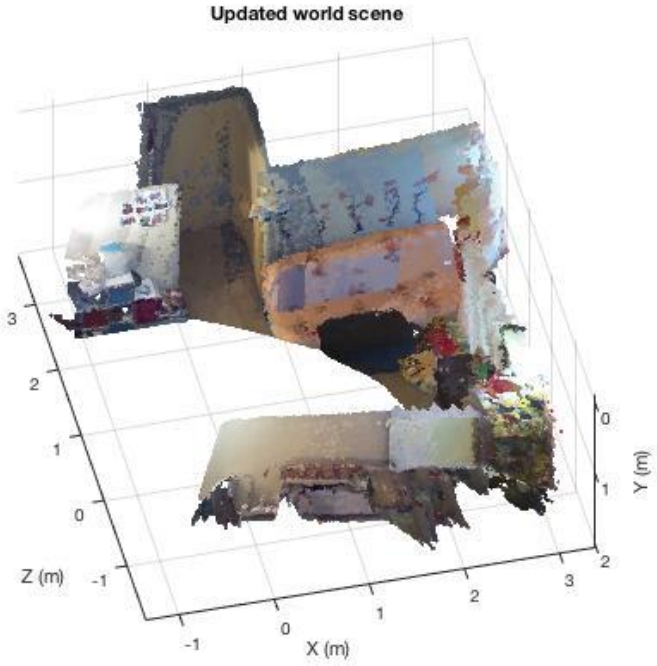



Image 2. 3D Point Cloud Stitching



Works Cited

[1] Joint 3D-Reconstruction and Background Separation in Multiple Views Using Graph Cuts. B. Goldlucke and M. A. Magnor. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003. DOI:10.1109/CVPR.2003.1211419.

[2] High-Resolution Image Reconstruction from Multiple Differently Exposed Images. B. K. Gunturk and M. Gevrekci**.** In *IEEE Signal Processing Letters*, April 2006. DOI:10.1109/LSP.2005.863693.

[3] Automatic Reconstruction of Piecewise Planar Models from Multiple Views. C. Baillard and A. Zisserman. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999. DOI: 10.1109/CVPR.1999.784966.

[4] Normal Vector. E. W. Weisstein, [Online]. Available: http://mathworld.wolfram.com/NormalVector.html.

[5] What is Point cloud. IGI Global. In *What is Point cloud | IGI Global*, 2016. [Online]. Available: http://www.igi-global.com/dictionary/point-cloud/36879.

[6] FIT3D Toolbox: Multiple View Geometry and 3D Reconstruction for MATLAB. Isaac Esteban, Judith Dijk, and Frans Groen. In *Electro-Optical Remote Sensing, Photonic Technologies, and Applications IV, 78350,* October 2010. DOI:10.1117/12.864112.

[7] A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011. https://rgbd-dataset.cs.washington.edu.

[8] MeshLab Stuff: Meshing Point Clouds. P. Cignoni. In *MeshLab Stuff*, Sept 2009. Available: http://meshlabstuff.blogspot.com/2009/09/meshing-point-clouds.html.

[9] Point Cloud Processing. Northeastern University College of Computer and Information Science (CCIS), [Online]. In *rigid motions*, 2016. Available: http://www.ccs.neu.edu/home/rplatt/cs5335_2016/slides/point_clouds.pdf.

[10] The PCL Registration API. The Point Cloud Library, [Online]. *Documentation - Point Cloud Library (PCL)*. Available: http://pointclouds.org/documentation/tutorials/registration_api.php.

[11] A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., & Szeliski, R.. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006. DOI:10.1109/CVPR.2006.19.

## APPENDIX

```
% 3D Reconstruction
% StitchPointClouds Modified.m
%
% Chinasa T. Okolo
% CSCI 181P
% Project
%
% Combine multiple point clouds to reconstruct a 3-D
%scene using Iterative Closest Point (ICP) algorithm
%
% Code referenced from: Math Works StitchPointClouds Example

% Register two point clouds
dataFile = ('meeting_small_1.mat');
load(dataFile);

% Extract two consecutive point clouds and use the first point cloud as
%reference.
depth1 = imread('meeting_small_1_1_depth.png');
[pcloud1, distance1] = depthToCloud(depth1);

depth2 = imread('meeting_small_1_2_depth.png');
[pcloud2, distance2] = depthToCloud(depth2);

ptCloudRef = pointCloud(pcloud1);
ptCloudCurrent = pointCloud(pcloud2);

% Create gridfilter
gridSize = 0.1;
fixed = pcdownsample(ptCloudRef, 'gridAverage', gridSize);
moving = pcdownsample(ptCloudCurrent, 'gridAverage', gridSize);

% Find rigid transformation
tform = pcregrigid(moving, fixed, 'Metric','pointToPlane','Extrapolate', true);
ptCloudAligned = pctransform(ptCloudCurrent,tform);

% Create the 3D scene
mergeSize = 0.015;
ptCloudScene = pcmerge(ptCloudRef, ptCloudAligned, mergeSize);

% Visualize the input images.
figure
subplot(2,2,1)
imshow(ptCloudRef.Color)
```

```
title('First input image')
drawnow

subplot(2,2,3)
imshow(ptCloudCurrent.Color)
title('Second input image')
drawnow

% Visualize the world scene.
subplot(2,2,[2,4])
pcshow(ptCloudScene, 'VerticalAxis','Y', 'VerticalAxisDir','Down')
title('Initial world scene')
xlabel('X (m)')
ylabel('Y (m)')
zlabel('Z (m)')
drawnow
```