UNIVERSITY OF WATERLOO
Faculty of Science

Developing Procedural Machine Vision Intelligence

Work Term Report
Kitchener, Ontario

Prepared by
Carter Minshull
3B Physics and Astronomy
ID 20472922
January 15, 2016

Mackenzie King Village
Waterloo, Ontario
N2L 3G1

January 15, 2016

Jeff Chen
Department Chain and Professor, Faculty of Science
Department of Physics and Astronomy
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

Dear Prof. Jeff Chen:

This report entitled, "Developing Procedural Machine Vision Intelligence" was prepared as my Work Report for my fourth coop term based on software developed by me during my placement. This is my third work term report. The purpose of this report is to show conceptual challenges and innovative solutions demonstrated while completing a specific project during my time on the Christie Digital Research and Innovation Software team.

This report was written entirely by me and has not received any previous academic credit at this or any other institution.

Sincerely,
Carter Minshull

Carter Minshull
ID 20472922

Table of Contents

List of Tables and Figures

Summary

In this report we examine the design and implementation of the partnered "segmentation" and "location" modules added to the "AutoMap" software tool being developed at Christie Digital, and how they affected the ability of the software to function successfully while not deviating from its core concepts and vision. By improving these features of the AutoMap software, there has been an increase in cases where this software is applicable. This report outlines the objectives, challenges and solutions along the way to creating procedural machine vision within AutoMap, as well as steps that can be taken going forward to maximize the benefits these developed modules.

1.0 Introduction

Projection mapping is the use of conventional projectors to project images and video known as content onto a surface that is three dimensional. Typical screens range from buildings to models to cars, and are generally used in the entertainment industry or as a form of demonstration. While producing spectacular visual effects, the process of correctly mapping onto three dimensional screens is a technical and computational challenge, requiring advanced software tools and expert human operators. Projection mapping is a growing industry and has been executed on large scale hundreds of times worldwide. At Christie Digital there has been progress towards creating a tool that could fully automate the projection mapping progress and revolutionize the industry. This software tool is named AutoMap.

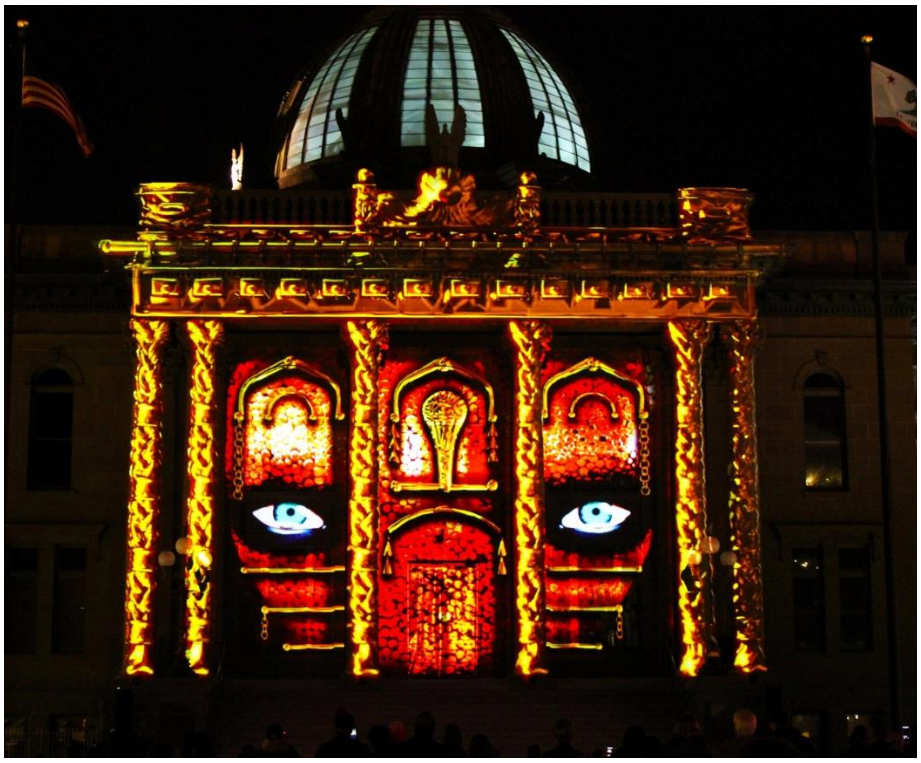

Figure 1 Mapped Building: A building illuminated with projection mapped content can take tens of man hours to properly align. This could be reduced to seconds using AutoMap. [1]

During my coop term at Christie Digital within the Research and Innovation Software group, my most significant contribution was the conceptualization, design and implementation of the Segmentation Module and accompanying revamped Location Module which together improve the functionality of AutoMap. AutoMap, currently in an engineering build, is used to provide a tool which can quickly and effortlessly scan an environment using a visual stereo triangulation measurement into a three dimensional point cloud. Once calibrated using a known image to locate the coordinates of each camera, AutoMap uses projectors to display geometric images over the scene while taking pictures of each frame to triangulate the 3D point cloud. AutoMap then processes these images to identify the location and orientation of a desired object and send the necessary information to a media application which will warp and push content to projectors. This allows to instantly projection map content to the location of the object screen with the push of one easy button and no manual alignment.

In short, AutoMap uses cameras and projectors to locate a screen (usually a 3D printed model in testing cases) in its field of view in order to automatically map projector content onto the screen. AutoMap can be used with physical projectors and cameras or tested in a virtual environment called "Staging Server" with virtual projectors, cameras and screens. As a tool that can turn hours of manual calibration and alignment into seconds of automated calculations at the push of a single button, AutoMap's core vision relies on simple operation and fast performance.

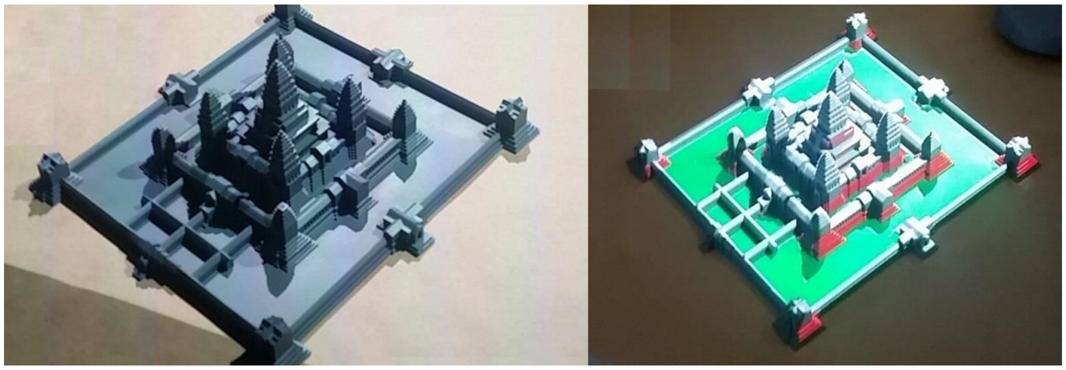

Figure 2 Mapped Screen: A 3D printed screen of Cambodian temple “AnkorWat” (left). The same 3D screen is shown with AutoMap projection mapping content onto the screen (right).

2.0 Initial State

In its initial state, AutoMap was highly effective at doing a certain specific job, but was limited in situations where it’s power as a tool could be leveraged. In order to function successfully, AutoMap required several physical criteria. AutoMap needed a near perfectly flat base for the scanned scene. The base also needed to be in the horizontal plane relative to the calibrated image and at the exact z height where the calibration was executed. Additionally, if any foreign object were to be present in the scanned field of view above the base ($z = 0$) it would throw off the entire point cloud of the screen, making it impossible to locate or map the content. These were restrictions that were easy to accommodate in a demonstrative environment, but failed when we sought to use it as the tool it was designed to be. In such cases, there is not always full control of the situation or environment. Relaying back to the vision of AutoMap, it is intended to be a versatile tool that can handle multiple cases and will save countless hours of human labour, with the simple click of a button.

The idea postulated that began the journey towards the AutoMap that exists today was the simple concept of being able to have two screens present in one scene and the ability to map unique content simultaneously onto each screen in one execution of AutoMap. This simple end goal is what inspired the development of the Segmentation Module.

3.1 Segmentation Module

The idea for the software module developed which was named "The Segmentation Module" was originally believed to be the first step towards multi-screen AutoMapping. The Segmentation Module quickly grew to become the backbone of not only multi-screen mapping, but a vast new generation of possible AutoMap cases, including previously impossible single screen cases. While at its foundation the Segmentation Module is three simple third party algorithms used to manipulate sets of three dimensional points called point clouds, what makes the module so powerful is the way in which these algorithms are implemented. From the third party library "Point Cloud Library" [2], the Plane Segment algorithm, the Euclidean Clustering algorithm and the Region Growing algorithm are used. The Plane Segment algorithm is used to strip a planar point cloud from a larger set. The Euclidean Clustering algorithm is used to separate a larger point cloud into clusters based on a proximity threshold separating each cluster. Finally the Region Growing algorithm takes a large point cloud and clusters it based on an angle threshold for how much a point's normal vector is allowed to vary from a point to its adjacent neighbour. Individually these algorithms are powerful at completing specific tasks, but when used correctly in series, they can effectively segment a very complex environment into something understandable by AutoMap. The module also consists of several linear algebra operations, as well as the three previously mentioned third party algorithms, all of which are tied together in the implementation of one flexible algorithm which I designed and tested myself. This algorithm dictates, based on changing observations, which of the many functions to use next and what parameters to give them. This allows the passing of any point cloud into the algorithm to have it return an intelligently segmented set of clusters needed by AutoMap to locate and map identified screens. The road to this procedural segmentation was not simple and required weeks of testing and innovation, but being set on maintaining the core belief of AutoMap being a hands-off tool instead of something where user parameters were required for functionality drove the motivation.

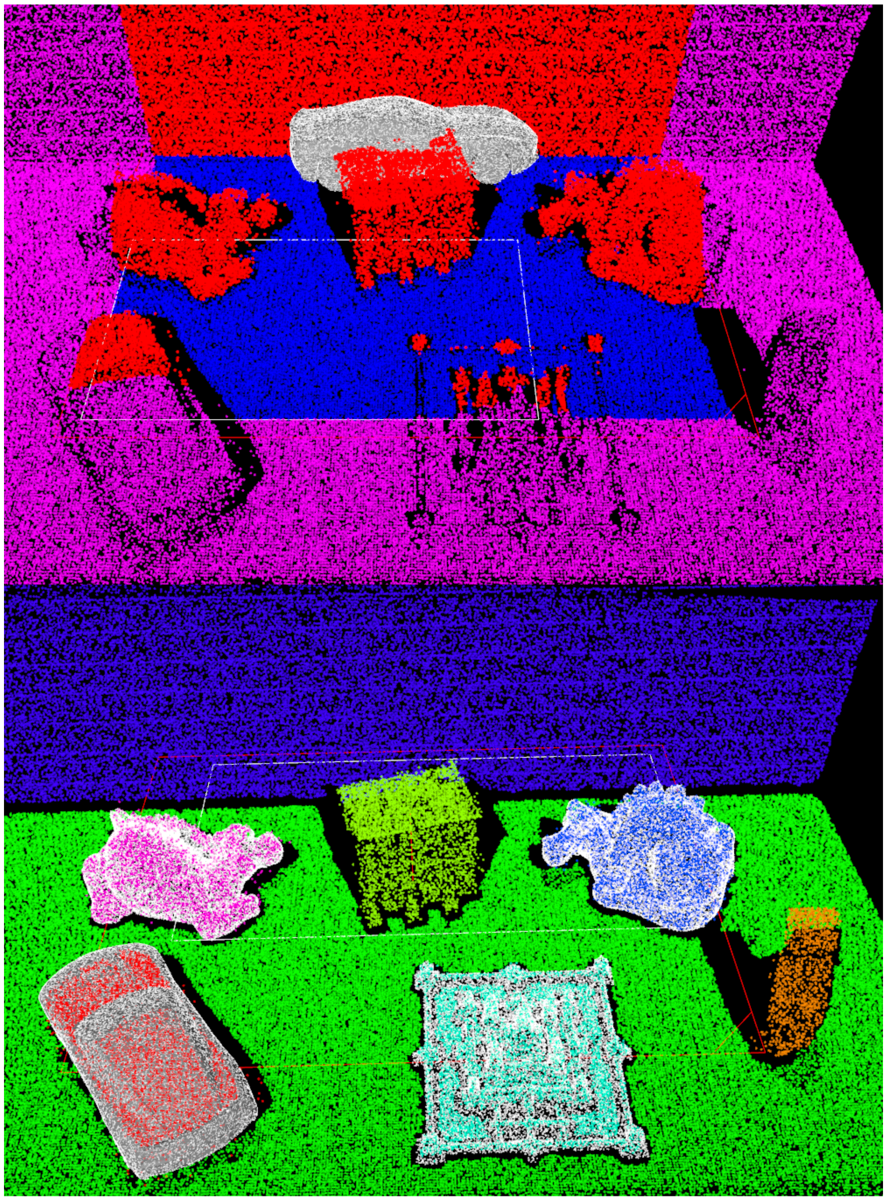

Figure 3 Initial to Final State of AutoMap: A screenshot of AutoMap's initial state trying to locate a car screen in a complex environment (top). The same complex scene is shown being processed by the finished AutoMap's segmentation and improved multi-screen location modules (bottom). Located objects shown as white interpolated point clouds.

3.2 Location Module

While the Segmentation Module neared completion, it was recognized that having an intelligently segmented scene was only as effective as AutoMap's ability to check and identify these segmented clusters as screens. In its initial state, any measured points were considered part of the screen selected by the user and AutoMap needed only to determine the translation and rotation to best align the 3D geometry with the measured point cloud. Now with many point clouds corresponding to potentially multiple screens as well as foreign objects, not only the translation and rotation of the geometry must be determined, but also which screen a point cloud likely is, if at all, must be identified. The solution was to create two methods for locating: single-screen location where the desired screen is given by the user and is much faster, and multi-screen location which tests all known screens against every cluster to find all known geometries and their orientations.

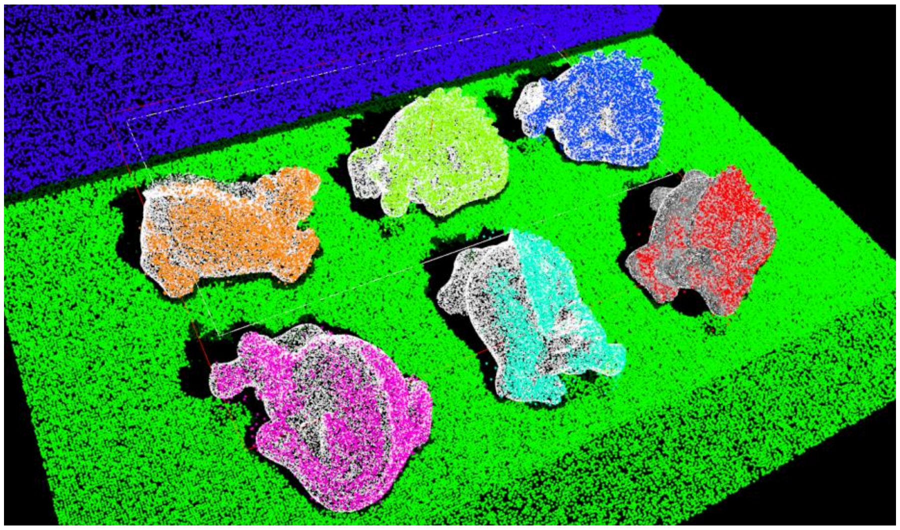
Figure 4 Locate Module: a screenshot of AutoMap using the revamped location module to locate the position and varied orientation of six dragon screens. Successful locates shown with white interpolated point clouds.

4.0 Challenges

While developing these key modules for AutoMap there were many challenges which sought to derail progress. The two main overlaying challenges were maintaining the simple "one easy button" functionality for AutoMap, and being able to execute the program in as little time as possible, ideally staying within the tens of seconds' order of magnitude seen in AutoMap's initial state. The modules would need immense functionality additions without a drop in performance or simplicity of user interface.

These two pillars of simplicity and performance tied in closely to the two modules being assembled. Within the Segmentation Module, it was most difficult to create a method of handling many cases with little to no additions to the one button interface. At the same time, it was required to transition from locating a single known screen to identifying potentially dozens of screens from a library as well as their orientations, while having little impact on the performance.

5.0 Solutions

While there were many specific challenges overcome in this project, three examples where I found innovative and effective solutions brief enough to discuss in this report are described below.

Firstly, in the Segmentation Module, we ran into the issue of screens with planar elements losing subsets of their point cloud to the planar segmentation. In order to remain true to the simple interface, the software needed its own way to determine when to recombine incorrectly separated point cloud clusters. The solution was a combination of some simple property assessments of the point clouds, which would recombine clusters only in the correct cases. As seen below, a combination of centroid and distance to furthest point (shown with a white line trace in FIG 5) within a parent cluster are used to determine if it is appropriate to absorb adjacent child clusters. The result is correct absorbing of child clusters that in reality belong to a specific screen or object.

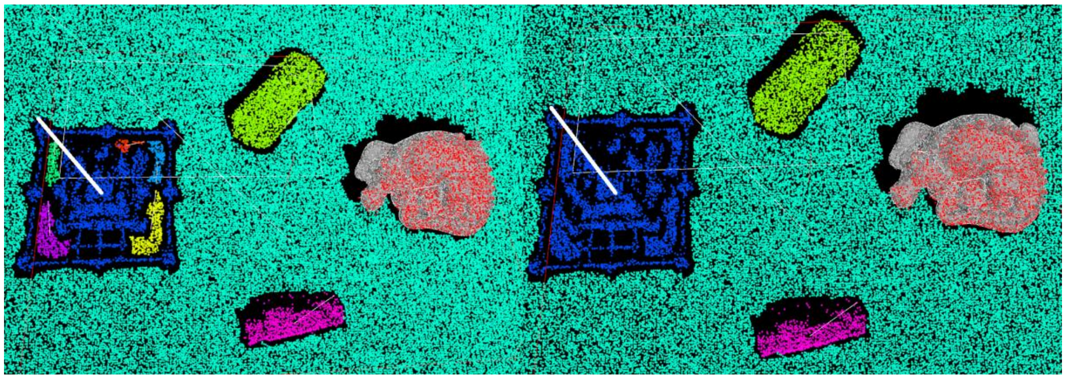
Figure 5 Absorbing Clusters: AnkorWat point cloud is shown broken into several child clusters (left). The AnkorWat parent cluster absorbs suitable child clusters to accurately represent the screen scanned (right).

The second challenge to overcome was the issue of unintentional bridging of clusters. With the early implementation of the segmentation algorithm, it struggled to differentiate separate screens or objects when placed in close proximity. This was due to the default proximity threshold defined in the Euclidean Clustering. No amount of fine tuning this variable could allow it to properly separate close objects without simultaneously segmenting larger screens into several clusters which would be too difficult to recombine. The solution was simple. By adding a fine pass of Euclidean clustering to certain clusters generated from the rough Euclidean clustering of the entire set (minus planes), enough noise and stray points were removed to allow for incredibly close proximity of objects while still being able to separate them.

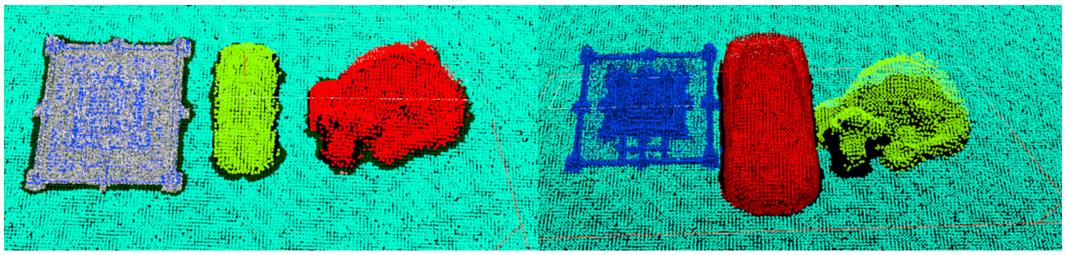
Figure 6 Fine Euclidean Pass: Scene in AutoMap with three screens located as close as possible before bridging of clusters unintentionally occurs (left). Following the fine Euclidean Cluster pass, screens can exist much closer without unintentional bridging (right).

Thirdly, in the Location Module, an incredible challenge was faced by completing 5-20 times as many calculations in a similar amount of time as done previously. With improvements to the identification function, it could now be determined if an object was not in the screen library, saving valuable time doing a fine orientation calculation. Additionally, by leveraging previous knowledge such as the origins of each cluster, many point clouds could be excluded altogether from the location step due to them being planar in nature (from plane segment). After improving the performance by a reasonable factor, the effects of the Segmentation Module could be utilized to further improve the Multi-Screen locate. Using knowledge that the segmentation would produce a set of clusters ordered in decreasing size, our library of screens could be organized with screens decreasing in size as well to increase the chances of testing the correct screen against a point cloud on the first attempt. This, along with a restructuring of the data structures which saved performance elsewhere, allowed the multi-screen locate to exist within the bounds deemed acceptable. In addition, an improved single-screen locate was implemented using strategies developed for the multi-screen locate, which will perform at least the same or better than the previous method used.

6.0 Conclusion

This report examines how to effectively implement improvements to AutoMap which not only allow for the desired case of two screens in a simple environment, but a plethora of new possibilities. By implementing a solution based on logic and growth, the goal was achieved as well as laying the foundations for dozens of other elements to be added in the future and countless cases which are now possible. With the modules added, AutoMap is no longer bound by the strict coordinate system previously required and can now handle virtually any environment tested. AutoMap's segmenting and locating abilities are now at such an advanced stage they can even locate a screen being held in mid-air and correctly segment away the user's hand and locate the partially covered screen. In addition, the possibility for infinite objects to be effectively clustered and to locate as many screens as are present now exists.

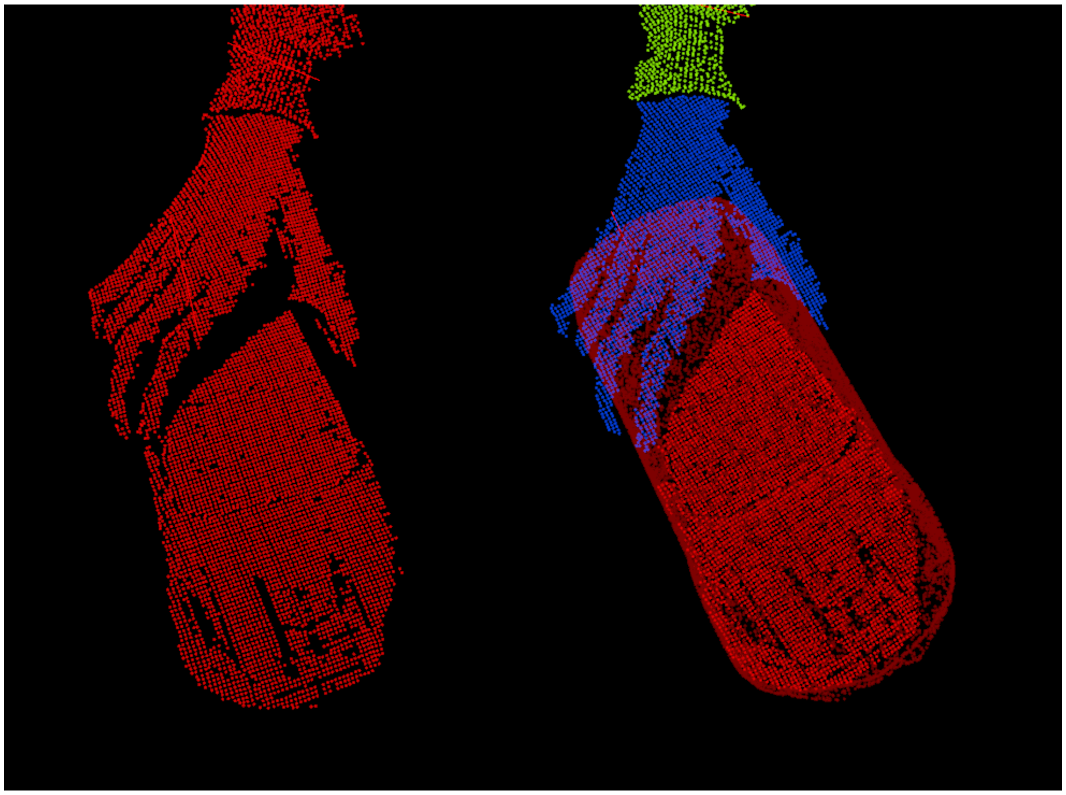

Figure 7 Handheld Car Screen: A car screen being held mid-air by users hand before segmentation (left). Same point cloud post-segmentation with interpolated point cloud of the located screen shown in dark red (right).

7.0 Recommendations

Continued work on AutoMap and the tools created this term will provide many opportunities for improvement in the future.

1. Improvement to AutoMap

Still in its infancy, AutoMap has many directions of possible growth. By continuing to expand upon the work done on the segmentation and location modules, the set of acceptable cases that can be handled will continue to grow. There is also work being done to further improve the single button interface to an even simpler “no button” AutoMap. This requires real-time checking of the camera images, to determine when objects move or are added to the visible scene. Once implemented, the user need not operate any software at all. One would simply need to place screens within the field of view and wait for AutoMap’s procedure to be triggered automatically. This would be the pinnacle of the simple easy interface vision of AutoMap.

2. Leveraging Modules

Another recommendation left with the team following my coop term is to consider leveraging these versatile software modules for applications other than AutoMap. There is a high demand for procedural segmentation software in any platform that uses point clouds. Being intentionally generic in their design, these modules could be easily implemented into other projects at Christie Digital to better allocate resources.

Citations

[1] Christie Digital, http://www.christiedigital.com

FIG 1: from Christie Digital Facebook Page

[2] Point Cloud Library, http://pointclouds.org/

Segmentation Algorithms:

http://docs.pointclouds.org/trunk/group__segmentation.html