Danfoss Visual Inspection System

PROJECT PLAN

Team Number	**Dec1704**
Client	Radek Kornicki
Adviser	Alexander Dogandzic
Team Members / Roles	Evan Woodring -Team Lead Nicholas Gerleman – Key Concept Holder Joseph Elliott – Communications Cory Itzen – Webmaster
Team Email	Dec1704@iastate.edu
Team Website	Dec1704.com
Revised	3/31/2017 – v2.0.0

Contents

1 Introduction

1.1 PROJECT STATEMENT

Our goal with the Danfoss Visual Inspection System is to pick up defects in assembly line products. The general flow goes as follows: scan a product with a camera, generate a 3D model of the product, compare the generated 3D model with the product's corresponding CAD model, and determine if there is an error in the product.

1.2 PURPOSE

The driving purpose of this project is to eliminate waste. An incorrectly configured product that has been shipped off to a client will always be shipped back. This is wasteful for the sellers, consumers, and shipping companies. Providing an accurate means of automated error detection will be beneficial to all involved parties because it saves money.

1.3 GOALS

Regarding some current goals, we would like to achieve the following:

- Successfully generate accurate 3D models of products.
- Successfully compare two 3D models to determine the location of errors, or lack thereof.
- Build a platform to scan products without human interaction.

Overall, our goal is to have a fully functional system on a test assembly line that can accurately report errors for any given product. We envision a photo booth-like area where products can be scanned with consistent lighting and rotation. We also imagine having the software run quickly, under 20 seconds.

2 Deliverables

Module to Generate Useful Point Clouds from All Incoming CAD File Formats

We expect to receive JT files representing the products we will be scanning. To progress, we need to convert these JT files into a mesh representation such as an OBJ file for easy point cloud to point cloud comparisons.

Module to Generate 3D Models from Scanned Products

We are using the RealSense 3D camera to scan objects. The SDK for this camera allows an easy generation of OBJ files representing the scanned objects.

Module to Align Multiple Point Clouds

It is necessary to align the two point clouds before calculating error between them. This process requires finding optimal scale, translation, and rotation to compensate for differences in orientation and physical positioning.

Module to Compare Two Point Clouds

The crux of the visual inspection system is the comparison of the two point clouds. This module will ensure that any two point clouds can be compared, and how probability that an error exists in the product.

End-to-End Prototype Built from the Previous Modules

These four modules allow us to build an end-to-end prototype. The prototype will be a simple linking of the four modules, tied together behind a user-friendly interface. This prototype can be used to identify pain-points in our methodologies and reduce risk for our final design.

Prototype Testing Results

Danfoss has happily granted us a facility to test our system. We will use this facility to get accurate reports on how well our prototype is working for real-world scenarios.

Revised System

With testing follows errors. We do not expect to have a completely bug-free system when we first reach the testing phase. We do, however, expect to expel all unexpected behaviors from the system before the deadline is met.

3 Design

3.1 PREVIOUS WORK/LITERATURE

- The KD Tree is a data structure used for spatial indexing. It allows for fast computation in multidimensional sets such as finding a nearest neighbor by Euclidean Distance. This data structure can be useful for determining localized error. [1]

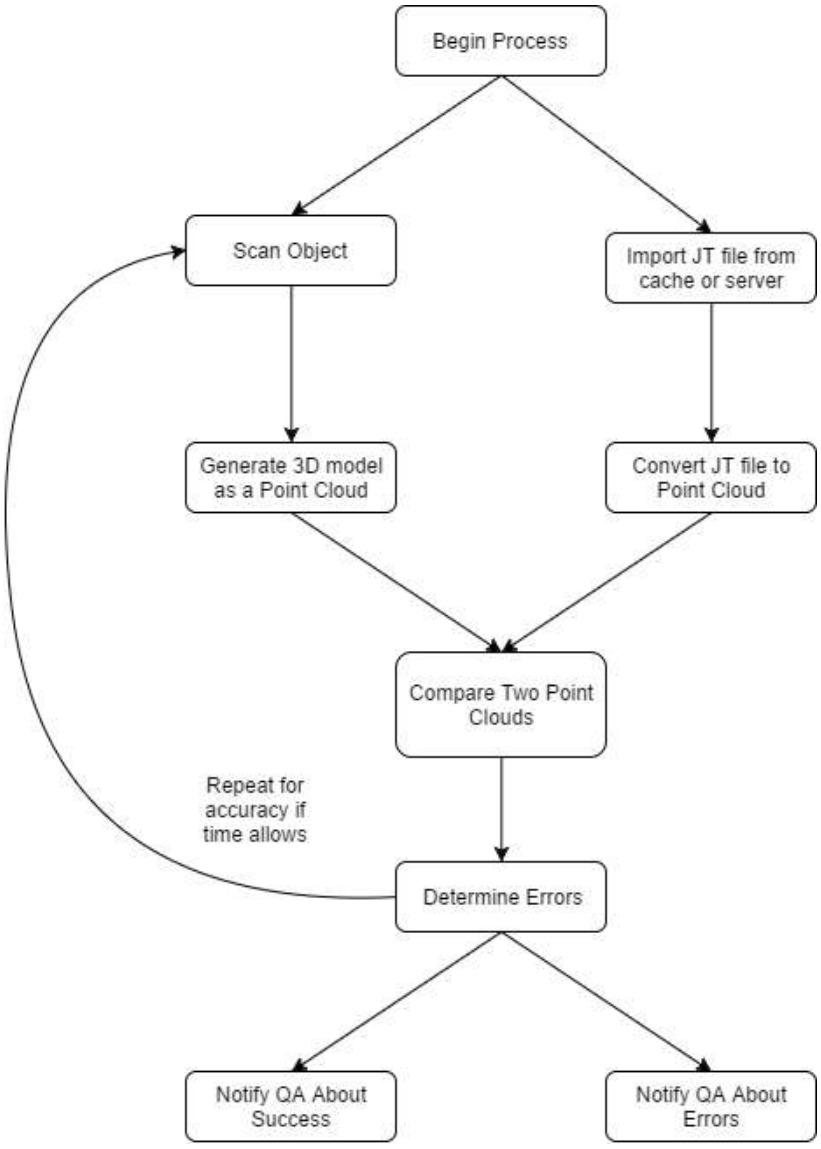
- Iterative Closest Point is a method developed for point set registration. The algorithm aims to minimize differences between point clouds by applying linear transformations. It has been successfully used for 3D alignment and reconstruction. [2]

- KinectFusion is an algorithmic process and tool that allows 3D reconstruction from multiple low resolution depth maps (such as those provided by a depth camera). This tooling has been integrated into the SDK of the currently used RealSense camera. [3][6]

- The JT file format is used for object visualization. It provides many different forms of geometric primitives and it the format used in the CAD files we are provided. The specification is openly available and there are some prebuilt tools that allow usage of the files. [4]

3.2 PROPOSED SYSTEM BLOCK DIAGRAM

The process to achieve our goals can be broken up into several discrete steps. The first phase involves obtaining point clouds from both the physical object and CAD model. These point clouds are then compared to determine optimal alignment. We use these aligned point clouds to determine possible error in the system. The scan is repeated if there is low certainty of the existence of an error. The system finally alerts the user to either the absence of error or shows where it believes an error might have occurred.



3.3 ASSESSMENT OF PROPOSED METHODS

Several different methods exist for each process in the system. The methods chosen are those which promise the greatest chance of success as well as ease of implementation.

Object Scanning and Mesh Generation

The evaluated scanning hardware provides a wealth of built-in software functionality. The Intel RealSense camera software development kit for Windows provides built in functionality for mesh generation of a physical object [5][6]. Preliminary evaluations of this tooling showed promising results and low implementation difficulty. A mesh is the converted to a full point cloud by extracting captured vertices.

We have also tested the Occipital Structure Sensor. Our conclusion thus far has been that the RealSense is the superior sensor. The RealSense is both easier to use, and has generated better scans. We will be moving forward with the RealSense camera.

Converting JT File to Point Cloud

Several differing methods are currently under evaluation for this process. Existing open source tooling and libraries were unable to open provided JT files for unknown reasons. Manually opening the file in the commercial Autodesk Inventor and exporting to a mesh was tested as correctly working. We are currently communicating with the client to determine if using pre-converted mesh files is feasible for their targeted use case. A custom tool may need to be created using the JT specification if this is not.

Converting a mesh from a CAD model to a point cloud presents several challenges not present in creating a point cloud from a physical object. Simple geometry such as a large flat surface may be expressed using a small number of vertices. This differs from a mesh obtained by sampling the surface at fixed points. This can be alleviated by the process of tessellating the CAD mesh. Points may be added in ideal surface positions by interpolating between points in each face. This allows the CAD mesh to be expressed as a point cloud of arbitrary density.

Point Cloud Alignment

The point clouds representing the physical and ideal objects must be accurately scaled and aligned to detect error. A rudimentary method to do this is ensuring alignment during capture and manually accounting for the offset in distance and model sizes. This method lacks robustness and increases the manual work necessary to use the system. This process is instead done algorithmically using the Iterative Closest Point method. The method approximates an optimal linear transformation to minimize mean squared error between point clouds. It has many pre-existing implementations, reducing difficulty in implementation.

Error Detection

Several methods have been proposed for error detection but testing is required to determine the best. A simple method for finding localized error is to map vertices in the physical point cloud to the nearest vertex in the CAD model point cloud. If the entire physical scan matches the model and no errors exist, the set of these distances should be low. Errors or defects are determined by detecting clusters of vertices that are mapped with long

distances. This method does not however account for the possibility of large missing parts in the defective product. It is possible that multiple methods may be used in conjunction to increase accuracy in this step.

3.4 VALIDATION

Validation will occur for each component of our process as well as the process as a whole. A test bench is currently being created allowing us to visualize and interact with the result of each component independently. This approach allows us to find design defects early in the design of each component rather than later after integration.

Our process will be validated against first synthetic, then real examples of correctly and incorrectly manufactured products. A large sample of real word 3D scans will be created to ease testing of components that do not directly interface with the camera.

Testing will be done to understand the constraints of the system. This includes factors such as capture environment, capture distance, object size, object material, and object size. Deliverables will be presented to the client in order to determine acceptance of their requirements.

4 Project Requirements/Specifications

4.1 FUNCTIONAL

The system shall be able to use a CAD model as a reference object

The system shall be able to scan an object of the size 3'x3'

The system shall be able to determine whether a product is generally defective

The system shall be able to determine the area of a defect

The system shall be able to generate 3D models available for later viewing.

4.2 NON-FUNCTIONAL

The system shall be able to operate for extended periods of time without failure.

The system shall be able to determine the error status of a product in under 30 seconds.

The system shall be able to be used by employees without specialized knowledge

The system shall be able to operate securely. All data will remain locally to Danfoss.

The system shall be able to reliably determine the error status of a product.

4.3 STANDARDS

Standard industry practices will be followed to reduce defects in code as well as improve overall quality. All changes made to the production system must be reviewed by other members of the team. Automated tests on these components will be created in the form of unit and integration tests. Regression testing will be performed to prevent adding defects to already working components. A strict set of coding conventions has been laid out to increase understandability of the code and prevent common types of errors.

5 Challenges

There are a few challenges specific to each module:

GENERATING POINT CLOUDS FROM PRODUCTS

Determining the right kind of camera proves difficult as each camera has their own SDKs to learn, platforms to run on, and devices that can power them. For example, the Intel RealSense camera is specific to Windows, while the Occipital camera, another sensor we tested, is primarily designed for use on the iPad.

The accuracy of point clouds may not be high enough to accurately determine errors in products. A product that has a special configuration that only differs from the original by a few centimeters may not be "picked up" by the camera we are using. This would lead to two different products, albeit minutely different, being interpreted as the same.

The limited view of a camera may lead to holes in the created mesh in areas such as their top in bottom. These holes need to be correctly stitched together or ignored as they may be misinterpreted as error.

GENERATING POINT CLOUDS FROM JT FILES

JT files themselves pose problems in generating point clouds. There are many software solutions available for users to view and construct JT files, but few that allow for programmatically manipulating JT files. Currently, this does not seem to a significant issue, as a user needs to select the configuration. This component is not really possible to fully automate, but we would like to automate as much as possible.

COMPARING POINT CLOUDS

There are several methods for computing error between two point clouds. Tools are available in the Point Cloud Library [5] that will prove useful here. It may be difficult to determine the best method for out purposes without extensive testing.

Solely detecting if an error in the object is present is not sufficient for the purposes of this project. The overall goal is to determine where errors are. Finding errors and relaying these locations to quality assurance employees may prove to be much more difficult.

KNOWLEDGE OF THE AREA

Many techniques and methods used in this field are conceptually new to this team. Large amounts of prior research will be required in implementing areas of the project.

6 Timeline

The Danfoss Visual Inspection System will be developed in two semesters. The project begins in the Spring 2017 Semester and concludes in the Fall 2017 Semester. See Appendices in Section 9 Figures 1 and 2 to see a detailed Gantt Chart outlining individual goals throughout the two semesters.

6.1 FIRST SEMESTER

Development in the Spring Semester consists of fully implementing and testing the first prototype. The first prototype requires the implementation of four separate modules. The modules will be individually worked on to create the first working prototype.

- Module to Generate Useful Point Clouds from All Incoming CAD File Formats
- Module to Generate 3D Models from Scanned Products
- Module to Align Multiple Point Clouds
- Module to Compare Two Point Clouds

The four modules are then linked together with a simple UI to build an end-to-end prototype. The prototype is not expected to be completely bug-free, but any and all unexpected behavior should be expelled. The remainder of the semester is spent testing the prototype, documenting issues and bugs, and preparing semester plans for the Fall semester.

6.2 SECOND SEMESTER

Development begins on the second prototype at the start of the Second Semester. The primary goal of this prototype is to focus on making the system production ready, while still ensuring the system remains bug-free and all behavior runs as expected. This prototype concludes with testing, where issues and recommendations are documented for the final implementation.

The final implementation is a fully functional system that achieves all of the initial goals we set out to achieve. Development of the final implementation is guided primarily from recommendations and issues that were documented from second prototype. All errors in the system shall be fixed and the Danfoss Visual Inspection System will be production ready.

7 Conclusions

Our goal with the project plan is to show that it is possible to compare a real-world object to a 3D model. The plan revolves around generating a proof of concept. We can achieve this by the following plan, laid out in detail in the Deliverables section. To give a quick summary, we will do the following:

- Show that we can make a point cloud out of the provided CAD models.
- Show that we can make point clouds out of real world objects.
- Show that we can compare the previous two point clouds to determine the presence of errors.

This plan allows us to achieve the proof of concept we are striving for. It further allows us to pursue a production-quality system.

8 References

[1] "KD Tree". Data structure used for spatial indexing.

https://en.wikipedia.org/wiki/K-d_tree

[2] "Iterative Closest Point". Algorithm developed for point set registration.

https://en.wikipedia.org/wiki/Iterative_closest_point

[3] "KinectFusion". Algorithmic process and tool that allows 3D Reconstruction.

https://events.ccc.de/congress/2011/Fahrplan/attachments/1969_kinectfusion-uist-comp.pdf

[4] "JT file format". File format used for object visualization.

https://www.freecadweb.org/tracker/file_download.php?file_id=503&type=bug

[5] "PCL – Point Cloud Library". Contains documentation on the modular point cloud libraries, software installations, and a comprehensive list of tutorials covering various aspects of the PCL.

[6] "RealSense SDK". Documentation pertaining to the camera we plan to use.

https://software.intel.com/en-us/intel-realsense-sdk-support

9 Appendices

Task Name	Start	End	Duration (days)	Plan
Generate Point Cloud from CAD file	1/9/2017	3/10/2017	60	Convert JT files representing the products into OBJ files for easy point cloud comparisons.
Test RealSense 3D Camera	1/9/2017	3/10/2017	60	Determine viabiliy of the Intel RealSense 3D Camera
Test Occipital Structure Camera	1/9/2017	3/10/2017	60	Determine viabiliy of the Occipital Structure Camera
Generate 3D Models from Scanned Products	1/9/2017	3/10/2017	60	Generate OBJ Files representing scanned objects
Implement Test Bench	2/1/2017	4/10/2017	68	Implement the Test Bench which allows for a Point Clouds to be taken in as input. Mutators are applied to each stage of the test bench.
Align two point clouds	2/1/2017	4/10/2017	68	Using the Iterative Closest Point Algorithm, rotate, transform, and scale two cloudsto put them in a similar position allowing for accurate error detection
Error Detection Algorithm	3/10/2017	4/10/2017	31	Algorithm to compare two point clouds and detect major differences between the two. Clusters of different vertices indicate errors in the scanned object.
3D Scanning	3/10/2017	4/10/2017	31	Programmaticaly scan an object with RealSense and output a JT file
Tessalation Algorithm	3/10/2017	4/10/2017	31	Alorithm to distribute vertices along the faces of an object.
Implement first prototype	4/10/2017	4/25/2017	15	Build a prototype to link the previous three modules. The prototype will consist of a linking of the previous modules with a simple UI.
Test protype in a real world scenario	4/26/2017	5/5/2017	9	Begin testing the first prototype. Document issues and bugs. Report on how well our prototype is working for the real-world scenario.
Summer Break	5/6/2017	8/20/2017	106	n/a
Develop Second Prototype	8/21/2017	9/30/2017	40	Implement bug fixes, issues, and recommendations gained from testing the first prototype. Investigate unique use cases and modify algorithms to handle these.
Test Second prototype	10/1/2017	10/11/2017	10	Test second prototype. Report bug, issues, and recommendations. In depth study of the algorithms and how they are working on actual scans.
Prepare Final Implementation	10/12/2017	11/30/2017	49	Implement bug fixes, fixes, and recommendations gained from Prototype Two. Prepare for the final implementation. Zero bugs, and all unique cases should be recognizable by the algorithm.
Final Implementation	12/1/2017	12/10/2017	9	Fully Funtional System. All errors in the system shall be fixed, and a fully funcitonal system will be available for production.

Figure 1

Senior Design Timeline

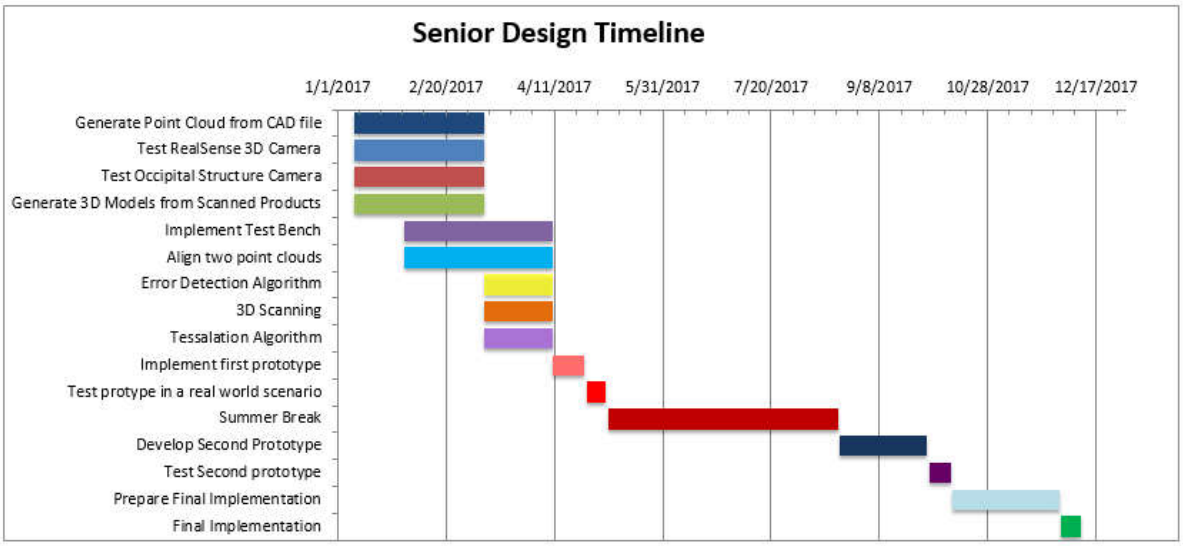


Figure 2

Gantt Chart for Senior Design Timeline