Steelcase Answer Component Verification Software Manual

University of Alabama in Huntsville

Department of Electrical and Computer Engineering

Jacob Brooks, Devon Eastin, Elaine Boyd, Lee Seemann

Table of Contents

[Software Overview 3](#_Toc449603040)

[C# Wrapper 3](#_Toc449603041)

[C++ Dynamic Link Library (DLL) 3](#_Toc449603042)

[C++ Verification Software 3](#_Toc449603043)

[Project Files 3](#_Toc449603044)

[C# Wrapper 3](#_Toc449603045)

[C++ Dynamic Link Library (DLL) 4](#_Toc449603046)

[C++ Verification Software 4](#_Toc449603047)

[Function and Variable Descriptions 5](#_Toc449603048)

[C# Wrapper 5](#_Toc449603049)

[C++ Dynamic Link Library (DLL) 7](#_Toc449603050)

[C++ Verification Software 7](#_Toc449603051)

[**Profile Class** 7](#_Toc449603052)

[**Color Class** 7](#_Toc449603053)

[**Length Class** 8](#_Toc449603054)

[**Quantity Class** 8](#_Toc449603055)

[**Database Class** 8](#_Toc449603056)

[**Driver Class** 9](#_Toc449603057)

[**Main Class** 11](#_Toc449603058)

[Graphical User Interface 11](#_Toc449603059)

[Software Environment Requirements 12](#_Toc449603060)

# NOTE: This document will be updated as future modifications to the software are completed.

# Software Overview

This software project contains three separate software projects created in Visual Studio Community 2015. These three projects are described below.

## C# Wrapper

This code is written in C# and is intended to serve as the interface between the existing Steelcase software and the order verification software written in C++. All of the data needed from the Microsoft Kinect (depth data and order image) is retrieved by this code and passed on to the C++ software for processing. This wrapper also contains a graphical user interface (GUI) that serves as a way of indicating the user to any issues with the order.

## C++ Dynamic Link Library (DLL)

This code is written in C++ and creates a dynamic-link library (DLL) out of the primary C++ verification software. This software bridges the gap and allows communication between the C# wrapper and the C++ verification software.

## 

## C++ Verification Software

This code is written in C++ and is the primary verification software for this project. All data processing (edge detection, depth data analysis…) needed for order verification is performed by this code. An array containing information about the success/failure of each order is returned to the C# wrapper, where this information is processed.

# Project Files

The significant files contained within each of the three software projects are described below. Additional files are present within each project, but should not need to be manually edited by future developers.

## C# Wrapper

|  |  |
| --- | --- |
| File Name | Purpose |
| Form1.Designer.cs | C# file that contains GUI code generated by Visual Studio |
| Form1.cs | C# file that contains the underlying functionality of the GUI |
| Program.cs | Primary C# file that contains: 1) code needed to interact with the Microsoft Kinect, 2) code needed to communicate with the C++ DLL, and 3) code needed to update the GUI upon completion of the verification process |
| Wrapper.csproj | C# project file that contains a group of project settings needed by Visual Studio |

## C++ Dynamic Link Library (DLL)

|  |  |
| --- | --- |
| File Name | Purpose |
| Steelcase\_Answer\_Verification\_DLL.sln | Visual Studio solution file |
| Steelcase\_Answer\_Verification\_DLL.vcxproj | C++ project file that contains a group of project settings needed by Visual Studio |
| dll.cpp | C++ source file that is responsible for 1) creating a DLL from the primary C++ software and 2) bridging the gap between the C# wrapper and primary C++ software |

## C++ Verification Software

|  |  |
| --- | --- |
| File Name | Purpose |
| Steelcase\_Answer\_Verification.sln | Visual Studio solution file |
| Steelcase\_Answer\_Verification.vcxproj | C++ project file that contains a group of project settings needed by Visual Studio |
| profile.cpp | C++ source file for the profile class, responsible for performing profile verification including edge detection |
| profile.h | C++ header file for the profile class |
| color.cpp1 | C++ source file for the color class, responsible for performing color verification |
| color.h1 | C++ header file for the color class |
| length.cpp1 | C++ source file for the length class, responsible for performing length verification |
| length.h1 | C++ header file for the length class |
| quantity.cpp1 | C++ source file for the quantity class, responsible for performing quantity verification |
| quantity.h1 | C++ header file for the quantity class |
| database.cpp | C++ source file for the database class, responsible for establishing communication with an Oracle database |
| database.h | C++ header file for the database class |
| driver.cpp | C++ source file for the driver class, controls the execution of the verification software when run in conjunction with the C# wrapper (it is bypassed when the verification software is run in isolation)\*\* |
| driver.h | C++ header file for the driver class |
| main.cpp | C++ source file for the main class, controls the execution of the verification software when run in isolation (it is bypassed when the software is run in conjunction with the C# wrapper)2 |

1These files were created, but there intended functionality was not implemented due to time constraints

2When run in isolation, main() controls the execution of the C++ verification software

# Function and Variable Descriptions

## C# Wrapper

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Purpose** |
| NUM\_FRAMES | const short | the number of depth frames to average before processing |
| frame\_count | int | tracks the number of depth frames already retrieved for the current order |
| depth\_frames | List<short[]> | holds the frames of depth data until they can be averaged together |
| sensor | KinectSensor | represents the connected Kinect sensor |
| depth\_data | short[] | array to hold the depth data provided by the Kinect |
| color\_data | byte[] | array to hold the color data provided by the Kinect |
| color\_retrieved | bool | used to determine if a color image has already been retrieved for the current order |
| color\_bitmap | WriteableBitmap | bitmap to hold rgb data from Kinect |
| file\_path | string | the path to the location where the Kinect color image is stored |
| dll\_ptr | IntPtr | pointer used to retrieve the results of the verification from the DLL |
| potential\_Sensor | var | temporarily represents the Kinect sensor during initialization |
| depth\_frame | DepthImageFrame | depth frame provided by the Kinect |
| color\_frame | ColorImageFrame | color frame provided by the Kinect |
| average\_depth | short[] | temporary array to hold average depth data |
| GUI | Form1 | instance of the GUI |
| instance | Answer | instance of the Answer class |
| results | int[] | holds the results of the verification process returned by the DLL |

**Function Name: Main()**

Purpose: controls the flow of execution for the C# wrapper

Return Type: void

Functions Called: processResults(), initKinectSensor(),

retrieveKinectDepth(), retrieveKinectColor(), processResults()

Input: N/A

**Function Name: initKinectSensor()**

Purpose: connects to an available Kinect sensor

Return Type: void

Functions Called: N/A

Input: N/A

**Function Name: retrieveKinectDepth()**

Purpose: initializes the Kinect depth sensor and sets up the depth event

handler

Return Type: void

Functions Called: N/A

Input: N/A

**Function Name: retrieveKinectColor()**

Purpose: initializes the Kinect color sensor and sets up the color event

handler

Return Type: void

Functions Called: N/A

Input: N/A

**Function Name: depthFrameReady()**

Purpose: event handler that executes each time a depth frame is available

for retrieval from the Kinect

Return Type: void

Functions Called: averageKinectDepth()

Input: object, DepthImageFrameReadyEventArgs

**Function Name: colorFrameReady()**

Purpose: event handler that executes each time a color frame is available

for retrieval from the Kinect

Return Type: void

Functions Called: N/A

Input: object, DepthImageFrameReadyEventArgs

**Function Name: averageKinectDepth()**

Purpose: averages multiple depth frames together to remove anomalies

Return Type: short[]

Functions Called: N/A

Input: List<short[]>

**Function Name: processResults()**

Purpose: uses array received from DLL to update the GUI and alert the

user to the success/failure of the verification process

Return Type: void

Functions Called: N/A

Input: int[], Form1

## C++ Dynamic Link Library (DLL)

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Purpose** |
| project\_driver | driver | starts the verification process in the C++ DLL |
| results | int\* | stores the results of the verification process provided by the C++ DLL |

**Function Name: Steelcase\_Answer\_Verification()**

Purpose: bridges the gap between the C# wrapper and C++ DLL, allows

verification process to begin by calling driver.startup()

Return Type: int\*

Functions Called: N/A

Input: short[]

## C++ Verification Software

### **Profile Class**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Purpose** |
| profile\_verified | bool | true if profile of the order is verified to be correct, false otherwise |
| depth\_data | short[] | depth data provided by the Kinect |
| PROFILE | enum | contains the possible results of the profile verification software |
| width | constant | the width of the Kinect frame |
| height | constant | the height of the Kinect frame |
| src | Mat | color image of the order |
| src\_gray | Mat | grayscale image of the order |
| thresh | int | threshold used for edge/curve detection |
| max\_thresh | int | threshold used for edge/curve detection |
| thresh2 | int | threshold used for edge/curve detection |
| max\_thresh2 | int | threshold used for edge/curve detection |

**Function Name: initialize()**

Purpose: controls flow of execution of the profile verification software

Return Type: bool

Functions Called: N/A

Input: N/A

**Function Name: edgeDetection()**

Purpose: primary edge detection source code

Return Type: void

Functions Called: N/A

Input: N/A

**Function Name: thresh\_callback()**

Purpose: executes each time an edge/curve detection threshold is modified

Return Type: void

Functions Called: N/A

Input: int, void\*

**Function Name: thresh\_callback2()**

Purpose: executes each time an edge/curve detection threshold is modified

Return Type: void

Functions Called: N/A

Input: int, void\*

**Function Name: cornerHarris\_demo()**

Purpose: executes each time an edge/curve detection threshold is modified

Return Type: void

Functions Called: N/A

Input: int, void\*

**Function Name: shiftDepthArray()**

Purpose: perform bit shifting on depth data to obtain depth info in

millimeters

Return Type: void

Functions Called: N/A

Input: short[]

**Function Name: detectProfile()**

Purpose: utilize depth data and results of the edge detection software

to determine profile

Return Type: PROFILE

Functions Called: N/A

Input: int, int, int, int, int, int, int, int, short[]

### **Color Class**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Purpose** |
| color\_verified | bool | true if color of the order is verified to be correct, false otherwise |

**Function Name: initialize()**

Purpose: controls flow of execution of the color verification software

Return Type: bool

Functions Called: N/A

Input: N/A

### **Length Class**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Purpose** |
| length\_verified | bool | true if length of the order is verified to be correct, false otherwise |

**Function Name: initialize()**

Purpose: controls flow of execution of the length verification software

Return Type: bool

Functions Called: N/A

Input: N/A

### **Quantity Class**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Purpose** |
| quantity\_verified | bool | true if quantity of the order is verified to be correct, false otherwise |

**Function Name: initialize()**

Purpose: controls flow of execution of the quantity verification software

Return Type: bool

Functions Called: N/A

Input: N/A

### **Database Class**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Purpose** |
| env | oracle::occi:Environment\* | Oracle environment variable |
| conn | oracle::occi::Connection\* | Oracle database connection variable |
| query | oracle::occi::Statement\* | Oracle statement variable used to form queries |
| results | oracle::occi::ResultSet\* | Oracle variable used to store the results of a query |

**Function Name: database()**

Purpose: database class constructor

Return Type: N/A

Functions Called: N/A

Input: N/A

**Function Name: connectDatabase()**

Purpose: creates a connection to an Oracle database

Return Type: oracle::occi::Connection\*

Functions Called: N/A

Input: string, string, string

**Function Name: queryDatabase()**

Purpose: creates and executes a query on an Oracle database

Return Type: oracle::occi::ResultSet\*

Functions Called: N/A

Input: oracle::occi::connection\*, string

**Function Name: ~database()**

Purpose: database class destructor

Return Type: N/A

Functions Called: N/A

Input: N/A

### **Driver Class**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Purpose** |
| file\_path | string | default location where Kinect color image is stored |
| verify\_profile | profile | instance of profile class |
| verify\_length | length | instance of length class |
| verify\_color | color | instance of color class |
| verify\_quantity | quantity | instance of quantity class |
| options | enum (int) | used to indicate the success/failure of a verification operation |
| results | int[] | array used to hold the results of the verification processes, this is returned to the C# wrapper for processing |
| is\_profile\_correct | bool | true if profile of the order is correct, false otherwise |
| is\_length\_correct | bool | true if length of the order is correct, false otherwise |
| is\_color\_correct | bool | true if color of the order is correct, false otherwise |
| is\_quantity\_correct | bool | true if quantity of the order is correct, false otherwise |
| userName | string | user name associated with the Oracle database used for the connection |
| password | string | password associated with the Oracle database used for the connection |
| connectString | string | used to create a connection to an Oracle database |
| query\_string | string | used to form an Oracle database query |
| conn | oracle::occi::Connection\* | Oracle database connection variable |
| result\_query | oracle::occi::ResultSet\* | Oracle variable used to store the results of a query |
| query | oracle::occi::Statement\* | Oracle statement variable used to form queries |
| data\_file | ofstream | used to open and write to a file containing the depth data |

**Function Name: driver()**

Purpose: driver class constructor

Return Type: N/A

Functions Called: N/A

Input: N/A

**Function Name: startup()**

Purpose: controls the flow of execution of the entire order verification

process, returns the results of the verification process to the C# wrapper

Return Type: int\*

Functions Called: N/A

Input: short[]

**Function Name: printDataToFile ()**

Purpose: used to print depth data to a file for viewing

Return Type: void

Functions Called: N/A

Input: short[]

**Function Name: ~driver()**

Purpose: driver class destructor

Return Type: N/A

Functions Called: N/A

Input: N/A

### **Main Class**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Purpose** |
| driver | driver | instance of the driver class, used to control program execution from main() |

**Function Name: main()**

Purpose: controls execution of verification software when run in isolation

from the C# wrapper

Return Type: int

Functions Called: N/A

Input: N/A

# Graphical User Interface

The following images show the graphical user interface (GUI) of the project. Figure 1 shows the GUI before the verification software has run, and figure 2 shows the GUI after the verification software has completed. The status labels at the bottom left of the GUI show the success/failure of each verification component by turning green (success) or red (failure). The “Start” button begins the verification process and the “End” button terminates the software.

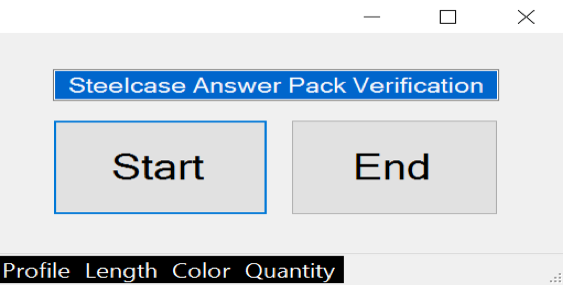


Figure 1

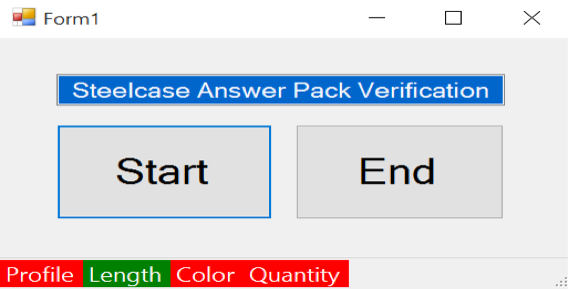


Figure 2

# Software Environment Requirements

The following requirements must be met by the environment where this software will run. It is possible to run the software without all of the items listed below, but certain sections of code will need to be bypassed and full functionality will not be possible.

1. OCCI (Oracle C++ Call Interface) – this API is provided by Oracle to allow applications to interact with Oracle databases seamlessly and can be downloaded directly from Oracle at the following website: <http://www.oracle.com/technetwork/database/occidownloads-083553.html>
2. Oracle database – if an Oracle database connection is not created, the software will fail to execute properly
3. Microsoft Kinect SDK V 1.8 – this software development kit is provided by Microsoft and allows applications to interact with the Microsoft Kinect, version 1.8 is the final release of the SDK for the Microsoft Kinect model used for this project and can be downloaded at the following website: <https://www.microsoft.com/en-us/download/details.aspx?id=40278>
4. Microsoft Kinect V 1.0 – an original Microsoft Kinect (Version 1.0) must be plugged in to a usb port of the computer where this software will run in order to execute the C# wrapper software. The C++ software can be run without a Kinect if sample data is provided.
5. OpenCV – this download provides the open source software needed by the profile verification software and can be downloaded from the following website: <http://opencv.org/>

In addition to downloading the necessary tools, the project settings of each C++ Visual Studio project must be updated to match the current computer’s file system. All of the paths of the files in the following locations must be updated to match the location of the necessary files on the current computer:

1. Properties 🡪 C/C++ 🡪 General 🡪 Additional Include Directories
2. Properties 🡪 Linker 🡪 General 🡪 Additional Library Dependencies

When all the above requirements are met, the software can be run using Visual Studio Community 2015.