1415 N CHERRY AVE
CHICAGO, IL 60642
(312) 281-6900
DMDII.ORG
DMDII@UILABS.ORG

DMDII
a UI LABS Collaboration
DIGITIZING AMERICAN MANUFACTURING

DMDII FINAL PROJECT REPORT

Authoring Augmented Reality Work Instructions by Expert Demonstration	
Principle Investigator / Email Address	Prof. Eliot H. Winer, ewiner@iastate.edu
Project Team Lead	Iowa State University
Project Designation	DMDII-15-04-03
UI LABS Contract Number	0220160010
Project Participants	Purdue University John Deere Company The Boeing Company Design Mill, Inc. Daqri, LLC
DMDII Funding Value	$1,006,912
Project Team Cost Share	$1,006,912
Award Date	Mar 22, 2016
Completion Date	Feb 28, 2018

DISTRIBUTION STATEMENT A. Approved for public release.

This project was completed under the Cooperative Agreement W31P4Q-14-2-0001, between Army Contracting Command – Redstone and UI LABS on behalf of the Digital Manufacturing and Design Innovation Institute. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Army.

TABLE OF CONTENTS

I. EXECUTIVE SUMMARY

The challenge of maintaining the flexibility of human workers with the efficiency and quality necessary to compete globally is exacerbated by the archaic nature of work instruction delivery, which has not changed substantially in more than a century. Workers conducting manual processes on the shop floor still typically refer to written instructions, which sometimes include reference drawings. Whether delivered on paper or a computer monitor, this method induces a substantial cognitive load on the worker. In an assembly process, for example, a worker is required to read an instruction, remember part numbers, retrieve necessary parts, position them properly, and finally conduct the assembly operation. Depending on the complexity of the task and their familiarity with it, a worker may feel compelled to refer to work instructions numerous times throughout the process. This can lead to increased completion time and assembly errors.

A promising solution to these challenges is the adoption of augmented reality (AR) technology for work instruction delivery. For an assembly operation an AR application can be used to superimpose video of work area with computer-generated visual features that provide instructions on assembly operations, such as which part to pick next, where to assemble a particular part, or which tool to use. However, despite promising AR research results, the broad adoption of AR in manufacturing industries has been hampered by a procedural technology gap, namely: how to facilitate the authoring of AR content.

The goal of this project was to develop the Augmented Reality Expert Demonstration Authoring (AREDA) product to provide a simple and intuitive method for rapidly authoring AR work instructions. This was achieved through tracking and recording the actual part manipulations of an expert using 3D depth cameras with advanced image processing, computer vision and point cloud matching algorithms. The system was based on existing successful research results from the proposers. Through cost-sharing partnership with aerospace (Boeing) and heavy equipment (Deere) sectors, functional requirements for AREDA were developed. Product requirements based on market opportunities were gathered from other project partners (DAQRI and Design Mill). Inter University collaboration (Purdue) afforded the opportunity to implement cutting edge depth camera sensors for use within AREDA.

AREDA was developed at a level commensurate with TRL 6 in an attempt to bring these new methods to commercialization faster. This is a key innovation that will accelerate AR adoption in manufacturing, which in turn will reduce time to author work instructions, enhance their communication to the shop floor, reduce process errors, and decrease manual process time – all of which will substantially improve US manufacturing productivity and competitiveness.

II. PROJECT REVIEW

Project Motivation, Scope, and Objectives

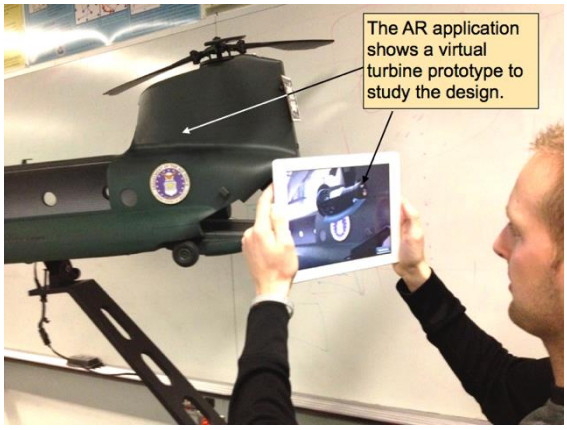
Despite decades of investment in increasingly advanced automation, manual processes are still common on the factory floor in many manufacturing industries. Global competition on cost and quality, as well as advances in technology continue to drive automation, especially for simple and repeatable manufacturing processes. However, for the foreseeable future, market demands for mass customization and

manufacturing agility will continue to require the judgment afforded by human labor – and those tasks remaining for people on the factory floor will continue to increase in complexity. The challenge of maintaining the flexibility of human workers with the efficiency and quality necessary to compete globally is exacerbated by the archaic nature of work instruction delivery, which has not changed substantially in more than a century. Workers conducting manual processes on the shop floor still typically refer to written instructions, which sometimes include reference drawings. Whether delivered on paper or a computer monitor, this method induces a substantial cognitive load on the worker (Nakanishi & Sato, 2012; Tang et al., 2003). In an assembly process, for example, a worker is required to read an instruction, remember part numbers, retrieve necessary parts, position them properly and finally conduct the assembly operation. Depending on the complexity of the task and their familiarity with it, a worker may feel compelled to refer to work instructions numerous times throughout the process. This can lead to increased completion time and assembly errors.

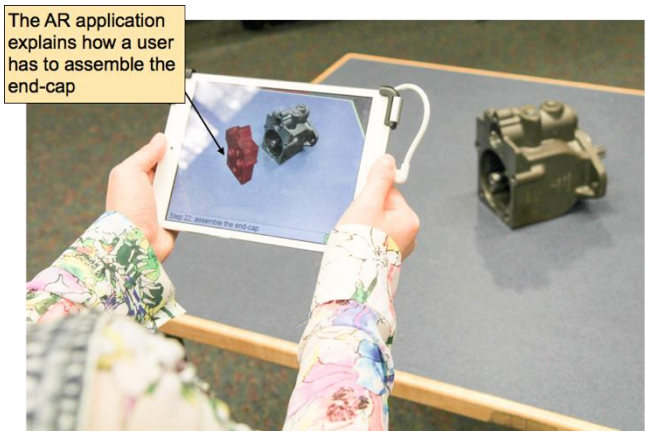
Additional inefficiencies exist in the time consuming and tedious process of authoring shop floor work instructions. For an assembly process, manufacturing engineers use their expertise to determine the order of parts (or subassemblies) to be assembled, the path, approach or alignment of each part, as well as its fastening or attachment method. The manufacturing engineer must access data from multiple sources to reference shop floor fixtures, necessary tools, CAD models of parts, and part locations relative to the shop floor. This information must be aggregated and scripted into a comprehensible process plan. Regardless of the method used to convey the work instructions this tedious authoring process makes it difficult to efficiently communicate manufacturing changes.

A promising solution to these challenges is the adoption of augmented reality (AR) technology for work instruction delivery. AR is a human-computer interaction (HCI) technology that superimposes the natural visual perception of a human user with computer-generated information (e.g., 3D models, annotation, and texts) (Azuma, 1997). AR presents this information in a context-sensitive way that is appropriate for a specific task, and typically, relative to a user's physical location. From a user's point of view, AR provides a representation of the physical world that has been "augmented" with virtual objects. Sample AR applications previously developed by the proposers are shown in Figure 1. In Figure 1a, an AR system is shown depicting a virtual engine model that is spatially registered and scaled relative to the physical model of a helicopter. Figure 1b shows an AR application for assembly. The (red) virtual part is shown in proper alignment relative to the physical subassembly.

One of the earliest applications of AR research is work instruction delivery for manual assembly. In these situations, an AR application is used to superimpose physical parts with computer-generated visual features that provide instructions on assembly operations. Since the computer-generated information presented is context-sensitive, spatially registered with, and superimposed on the physical part, the information is easier to comprehend (Neumann & Majoros, 1998; Sausman et al. 2012). Caudell & Mizell (1992) introduced the first application of this type. Since then, many studies, have been conducted, including several from the proposers, which indicate advantages of AR in comparison to typical instruction media (Tang et al., 2003; Wiedenmaier et al., 2003; Richardson et al. 2014; Radkowski et al. 2015). Most of the research compares AR applications with computer monitors (instructions shown on a display) and/or paper manuals. In general, the results show that the number of assembly errors, the time to identify and locate parts, and the overall assembly time can be reduced significantly using an AR system. In addition, hand-eye coordination tasks and mental workload can also be minimized.



a) AR Design Review



b) AR Assembly Instructions

Figure 1 Sample AR systems for a) design review, and b) assembly instructions

Despite these promising research results, the broad adoption of AR in manufacturing industries has been hampered by a procedural technology gap, namely: how to facilitate the authoring of AR content. In this context, authoring of AR content refers to specification and scripting of a variety of interaction phenomena, including: 1) virtual part representation, 2) 3D part animation paths, 3) pointers, symbols or glyphs, and 4) text annotations.

In most manufacturing organizations, access to the digital thread to obtain 3D product models is relatively straightforward, as is their conversion from native CAD into the polygonal formats required in AR systems. However, creating the four components of AR content listed above is a significant challenge. In addition to all the typical (non-AR) work instruction data a manufacturing engineer must accumulate, creating AR instructions requires expertise in at least one other software package to create the appearance and movements of parts necessary to convey a manual assembly process. Although some vendors are beginning to provide tools specifically aimed at authoring AR work instructions (e.g., Daqri and Metaio), most AR authoring is done with general 3D modeling and animation packages like AutoDesk's Maya. While the AR-focused products are better suited to the specific requirements of AR authoring, all of these systems require an additional learning curve for the manufacturing engineer. In addition, regardless of the tool learned, part representation and 3D path generation are not intuitive. They are generated either incrementally, by 3D manipulators (see Innovation section below), or via key-frame animation. In order for AR to make substantial inroads into industrial manufacturing, automated methods must be commercially available to: 1) identify parts, 2) track part movements, 3) identify and track proper assembly motions, and 4) easily allow text annotations and other graphical symbols to be introduced, if desired.

This goal of this project was to develop the *Augmented Reality Expert Demonstration Authoring* (AREDA) product to address the aforementioned objectives by providing a simple and intuitive method for rapidly authoring AR work instructions by tracking and recording the actual part manipulations of an expert. The system was based on existing successful research results from the proposers (Bhattacharya & Winer, 2015; Zhang & Huang, 2005; Zhang, 2011). The project solicited additional functional requirements from potential users, namely, project partners representing the aerospace (Boeing) and heavy equipment (Deere) sectors. In addition, product requirements based on market opportunities and commercialization potential was gathered from project partners DAQRI and Design Mill. All of the project partners have

experience working with several branches of the US Department of Defense (DoD) and other government agencies (e.g., Department of Energy, National Science Foundation). Requirements from these sectors was also accumulated into commercialization activities. These requirements guided the development of additional functionality to broaden the capability of the research prototype, optimize parameters to enhance its robustness, and simplify its integration into a commercial product. While the research was directed towards the manufacture of complex engineered systems, it has the potential to impact other market segments including education and small businesses. The breadth of experience of the project team, spanning users, providers, and researchers in AR and wearables represented the necessary skill set for project success.

This project addressed the Advanced Analysis (AA) thrust identified by the DMDII and the DoD as critical to significantly improve US manufacturing capabilities and reduce acquisition costs for the DoD. DMDII support was critical as there was no other source of funds that can be accessed to support such a broad team of experts in this critical emerging area. This project intended to overcome one of the last remaining barriers to widespread adoption of AR in manufacturing and enable increased productivity and competitiveness across a spectrum of manufacturing industries.

Methodology

The AREDA product aims to create AR work instructions by capturing an expert "demonstration" of an assembly procedure. As an expert carries out an assembly, the AREDA system observes the procedure, analyzes the parts and their locations, and transforms the observation into AR authoring instructions. The goal of the project was to develop the AREDA software application and prepare it for integration into a commercially viable product. Another project partner, Design Mill, an AR systems integrator, will begin using AREDA to produce customized AR content for its customers and also evaluate its effectiveness. The AREDA software was designed to comply with commercial standards for a technology readiness level (TRL) of seven (Department of Defense, 2011). AREDA has the ability to identify parts and track their movements. It also has limited ability to identify and track proper assembly motions. The ability to allow text annotations and other graphical symbols into the final work instructions was not completed as this was not one of the primary research challenges. It can easily be completed as the product is prepared for commercialization.

III. KPI'S & METRICS

An unbiased evaluation of AREDA's efficiency compared to the current state of AR usage within manufacturing industries is non-trivial. Given that the AREDA tool was developed in an academic setting, studies measuring assembly times will not be representative of the manufacturing sector. Also, time studies performed in a specific assembly line do not necessarily represent a broad range of industries either. These studies must be performed at a number of manufacturing facilities to arrive at a statistically significant and quantifiable KPI metric (e.g., AREDA is at least 30% more efficient than manual AR instructions). Adhering to the scope of the project proposal and time limitations, time studies in a number of industries have not been performed. An alternative is therefore provided to qualitatively describe AREDA's KPIs and metrics compared to manually creating AR work instructions for a specific assembly.

Figure 2 below represents a typical workflow of AR technology usage within manufacturing industries. A subject matter expert, or SME, (e.g., worker, supervisor, manufacturing engineer) relays component

assembly instructions to a modeler or a CAD designer. The modeler then creates AR based work instructions customized for the specific assembly. This approach has the following ramifications in terms of time delays and possible errors:

- A modeler or a CAD designer might not be readily available due to other work commitments, incurring variable delays in creating AR work instructions. This time delay is denoted by 'A1'. This metric (i.e. Availability metric) is an important for consideration because AREDA automates much of the traditional delays in an assembly line setting.
- A SME's expectations and the CAD designer's interpretations do not necessarily match up thereby resulting in human errors, as indicated by 'Error source 1'. Resolving these errors will require multiple meetings and iterations, resulting in time delays. This time delay is denoted by 'A2'.
- The modeler or the CAD designer manually creates CAD models and assembly animations, again subject to errors and iterative time delays. The errors are denoted by 'Error source 2' and time delays by 'A3'.
- Merging a proprietary CAD model into an AR work instruction requires conversions into mesh-based formats suitable for AR. For example, a Unity game engine for viewing AR assembly instructions does not automatically recognize geometry created in a CAD program (e.g., Solid Works, PTC Creo, etc.). This is also an iterative process because the SME and the modeler must reach a consensus before an AR work instruction is approved and finalized. This process constitutes additional time delays denoted by 'A4' and any errors caused are indicated by 'Error source 3'.

Therefore, in the current state, the overall time taken by a modeler or a CAD designer for creating usable AR work instructions for a specific assembly is given by $T_{present}$ = A1 + A2 + A3 + A4.

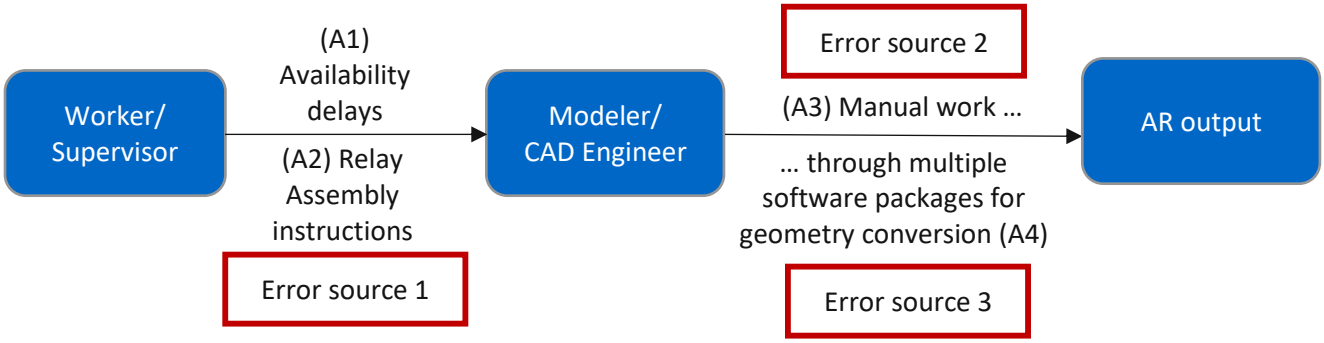


Figure 2 Current state of AR based assemblies in industries

Figure 3 shows AR work instructions generated by AREDA. A SME authors an assembly within AREDA equal to the actual assembly work time, denoted as 'B1'. Note that 'B1' is not a time delay, but a value-added assembly authoring time, and replaces the non-value-added time delays 'A1 + A2 + A3' as described in **Figure 2**. This means that any human errors and time delays introduced because of a third-party modeler or CAD designer have been eliminated.

AREDA captures these assemblies using its depth camera and computationally processes them to automatically recognize CAD parts, generate assembly steps and animations. This is the only step that subjectively incurs significant time because a computer algorithm attempts to automatically recognize parts in an assembly. Consequently, the computational time is dependent on the number of parts in the assembly. The time taken for this process is indicated by 'B2'.

The algorithms that recognize parts or assembly orientation occasionally fail and requires the SME to correct it in the refinement phase while authoring AR work instructions. This step once again does not require a third-party personnel intervention and can be accomplished within the AREDA interface. The time taken for error correction is denoted by 'B3'.

The total time taken for authoring and generating AR work instructions is given by T_{AREDA} = B1 + B2 + B3.

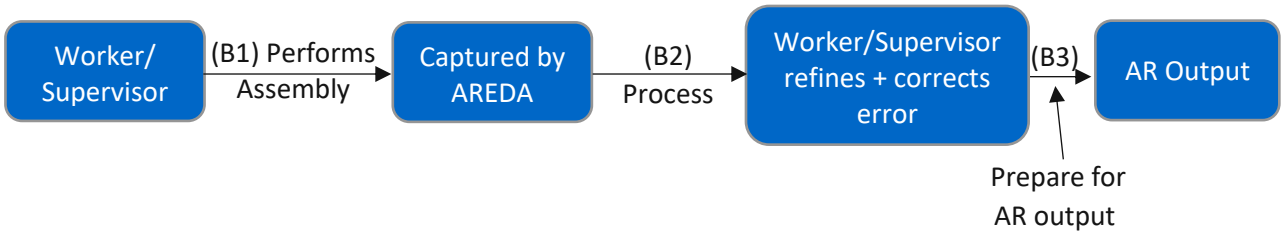


Figure 3 Assemblies using AREDA

It can be seen that the incurred time in the current state of custom building AR work instructions is substantially affected by human errors, miscommunications and misinterpretations. AREDA was designed to naturally overcome these delays, and time spent in using the interface is value-added beginning with work area calibration through AR work instruction generation. Since AREDA is an AR authoring tool, it can be used in computationally creating work instructions for any generic assembly process. The time savings achieved when using AREDA are hence characterized to be much greater than 30%. The performance improvement metrics for AREDA are summarized in Table 1 below.

Table 1 AREDA Performance Improvement Metrics

Metric	Baseline	Goal	Results	Validation Method
Enter Metric	Enter Baseline	Enter Goal	Enter Results	Enter Validation Method
Average time to create a complete set of AR work instructions	Time taken using current process of CAD and 3D modeling software without AREDA	Time decrease > 30%	Goal achieved	See description above on time delays in current state AR work instructions vs AREDA's time incurred
Average number of errors putting together an assembly using AR work instructions	Number of errors requiring disassembly and reassembly to correct using instructions not created by AREDA	Reduction to no significant difference between errors from AREDA and non-AREDA work instructions	Goal achieved	See description above on how human errors are resolved in current state AR work instructions vs AREDA's architecture naturally lending to minimizing errors
Correct AR instruction output provided by AREDA following demonstration	File formats and exchange mechanisms used to output instructions from one software and	Reduction to no significant difference between errors from AREDA and non-AREDA	Goal achieved	AR work instructions computationally generated and can be viewed in the refinement phase of the AREDA interface

	input to another for viewing in AR	produced work instructions		

IV. TECHNOLOGY OUTCOMES

Overview of AREDA

Error! Reference source not found.2 shows the hardware setup for AREDA. Authoring by demonstration relies on computer vision and image processing since the assembly process of the user is visually observed. Given the novelty of an AR authoring tool using a number of technologies, the development focused on a typical one-person work area. AREDA functionality can easily be extended to larger work cells with a suitable hardware setup (e.g., depth camera, lighting conditions, etc). Parts on the workbench are automatically identified and their position/orientation is tracked. Mis-identified parts/orientations are easily modified via the AREDA interface. A depth camera/camera-projector unit was required, above or behind the work area to capture the work area and assembly processes. This unit projects a structured light pattern (visible or infrared spectrum), records the reflection, and outputs a high-fidelity point cloud to facilitate user and object recognition and tracking.

To use AREDA an expert selects the parts that he or she intends to assemble and places them on the bench top area. AREDA automatically registers the parts with a digital CAD library and keeps track of their positions and orientations during assembly recording, steps generation and refinement phases. The expert performs the assembly steps moving required parts to their final locations, properly orienting them, and confirming the start/end of each step through AREDA interface. AREDA translates the observation of this procedure into formal assembly steps. Using the refinement phase, a user can modify steps, parts identified, or their orientations within the interface. By matching parts with their true CAD representations, AREDA integrated into, and significantly extended, the digital thread of product design in large manufacturers. AREDA hence is able to expand CAD models outside of traditional uses (i.e., simulation or computational analysis) and allows rapid access to the data on the shop floor in a meaningful way.

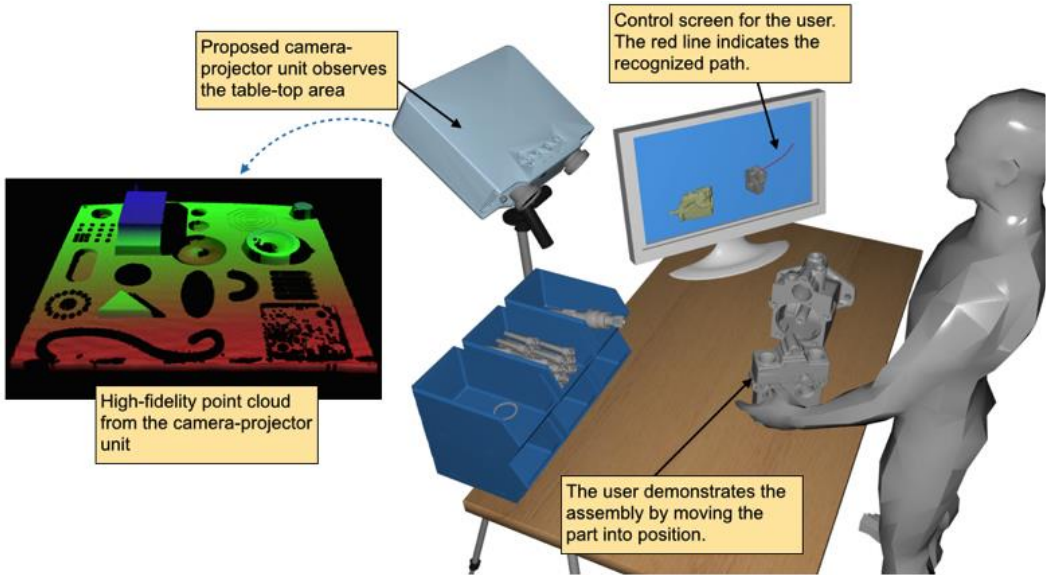


Figure 4 The AREDA system hardware setup showing point cloud generation

AREDA Architecture

A research software prototype of this authoring application has initially been developed and tested by the proposers (Bhattacharya & Winer, 2015) before this project commenced. At the time however, the architecture and functionality of the research prototype was at a TRL of three. **Error! Reference source not found.**3 shows the software architecture diagram for the AREDA system to move it to a TRL of seven. The architecture concept has been developed in close cooperation with project partners Daqri and Design Mill, from a software development perspective as well as Boeing and John Deere to specify data exchange interfaces for industry, DoD, and other government entities. A major part of this effort focused on implementing this software architecture. The architecture incorporated four major components: 1) point cloud generation from the depth camera, 2) object detection and tracking, 3) user detection and tracking, and 4) authoring instruction exchange interface. These represented the major technical task areas that were implemented within AREDA system to become commercially viable.

As shown in **Figure 4**, from the observation of the bench top by the camera-projector unit, a 3D point cloud representation of the work area is generated. The object detection and tracking component identifies the parts on the bench and tracks their position and orientation. The user detection and tracking component tracks the position and orientation of the hands of the expert. The position and orientation information from both modules are combined to compile the overall assembly instructions. The authoring instruction exchange interface prepares an output file that partners Daqri and Design Mill require to incorporate into their AR guided assembly viewers. Specifically, the AREDA system provides the six outputs listed below (Each of these outputs is represented in one of the four technical task areas identified in **Figure 5**):

- Automatic identification of all parts on a bench top work area.
- Recording the user's hand movement and part handling.

- Recording the assembly path and orientation of each part.
- Assembly sequence specification by expert input (expert will designate the beginning and ending of each step using AREDA interface.
- Recording the assembly time per step.
- Output of digital representation of assembly instructions suitable for commercial AR authoring and viewing environments.

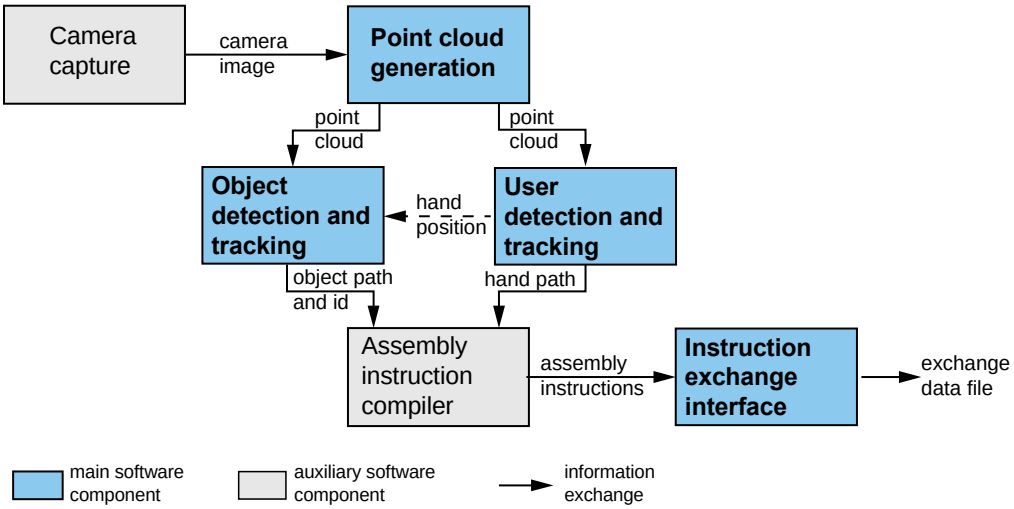


Figure 5 Commercial software architecture for AREDA

AREDA Features and Attributes - Point cloud generation

A point cloud is a set of 3D points representing a real or virtual environment. It is a very well understood technique that facilitates object detection and tracking (Schnabel et al., 2007; Rusu & Cousins, 2011). The AREDA system uses point clouds to represent the parts, assembly, and the expert during a demonstration. A high-fidelity 3D point cloud generation technology (Zhang & Huang, 2005) has already been developed, implemented, and utilized in several research projects (Mehta et al., 2008; Laughner et al. 2012). The method, digital fringe projection (DFP), is a special collection of structured light techniques. These techniques have advantageous features such as higher accuracy, resolution and less sensitivity to ambient light (Zhang, 2011). The system uses invisible near infrared (NIR) light for 3D sensing so as not to disturb the expert during the assembly demonstration (Ou et al. 2013). To mature the current capture and point cloud generation technology two sub-tasks were completed. First, the existing DFP system was encased in a simple to handle enclosure. Second, a calibration routine that facilitates installation at different physical locations of the project partners were implemented within the AREDA interface.

The existing hardware prototype incorporates an infrared camera, an infrared projector, a color camera, and timing hardware (Ou et al., 2013). The entire system was a lab prototype, assembled on an optical table or “breadboard”. These components were integrated into a housing, which can be mounted above the workspace on a factory floor. This development was conducted in close collaboration with all of the research partners to ensure requirements such as portability, temperature, lighting conditions, and vibration were accounted for in the final design. Project partner Daqri, having both AR hardware and software products, is ideally suited to lead this commercialization effort with Purdue the originator of the camera-unit technology. An automatic calibration routine was developed to allow easier setup within AREDA system.

AREDA Features and Attributes - Object detection and tracking

The object detection and tracking component is responsible for recognizing and tracking all parts in the assembly operations. The core functions are part identification, part alignment detection, part path recording during assembly actions, and measurement of assembly time. AREDA relies on point clouds generated by the depth camera to perform these core functions. During the assembly process, point clouds from the entire work area and all parts that are being assembled are filtered out except the one(s) being manipulated during the current step of the assembly sequence. This approach allows individual part identification and minimizes mismatching. Before the expert begins to demonstrate the assembly procedure, sections such as the bench top surface, walls, and other parts are easily removed using Random Sample and Consensus (RANSAC)-like object matching algorithms (Schnabel et al., 2007). Object matching algorithms based on RANSAC are well known, highly effective, and fast.

The AREDA system uses the DFP hardware system to capture 3D frames of the assembly environment. All CAD parts and their corresponding point clouds are uploaded within the AREDA interface. These are saved in an internal SQL database. The DFP hardware system captures the scene as viewed by the depth camera, which includes the work area, the surroundings, and the parts that are brought into the work area for assembly. Object recognition requires that all extraneous details except the part(s) to be assembled are filtered out. Using RANSAC and clustering algorithms, only parts that are within the work area are saved and the rest of the data is discarded. Every new part that the expert introduces to the scene is considered an assembly component candidate. The point cloud that is associated with this candidate is then stored and matched to reference objects prepared in advance from CAD models of the parts using the Iterative Closest Points (ICP) method (Besl & McKay, 1992). ICP solves a least-squares problem to align two sets of points in an iterative optimization process. Upon completion, an ICP match generates a fitness score to indicate how good the camera-projector point cloud matches against a CAD part. A lower fitness score indicates a better match.

The camera unit is fixed to a certain position and orientation to capture the work area and parts. As such, only a section of the part(s) volume being assembled faces the camera directly, and the camera does not necessarily capture 100% of the part(s) volume. This results in a partial point cloud generated by the camera that will be attempted to match against a point cloud from the entirety of the CAD part. Because of the missing volume, ICP matching and alignment could result in errors and an incorrect part will likely be identified by the process. To minimize such errors, ICP matching was implemented with camera point clouds oriented at different incremental orientation angles.

Once a fitness score is calculated by an ICP match, the camera point cloud is slightly rotated about a vertical axis. This new cloud is then matched against the CAD cloud. This sequence is performed at 10 different angles to maximize the camera point cloud exposure to the CAD part's point cloud. Every part in the generated steps undergoes this process, and the part that returns the least fitness across all parts in the library is designated as the part identified in that corresponding step. While this approach is not completely error proof, AREDA was able to minimize incorrect part matching by a significant margin.

Figure 6 shows the matching and alignment process for a certain step. **Figure 4**a shows the point cloud of a CAD part within a bench-vise assembly. **Figure 4**b shows the point cloud generated by the camera-projector unit for the base after filtering surrounding noise. **Figure 4**c shows both the clouds ICP matched

and aligned, with the denser points in blue representing the CAD part's cloud, and the blue-green points representing the camera-projector unit's point cloud.

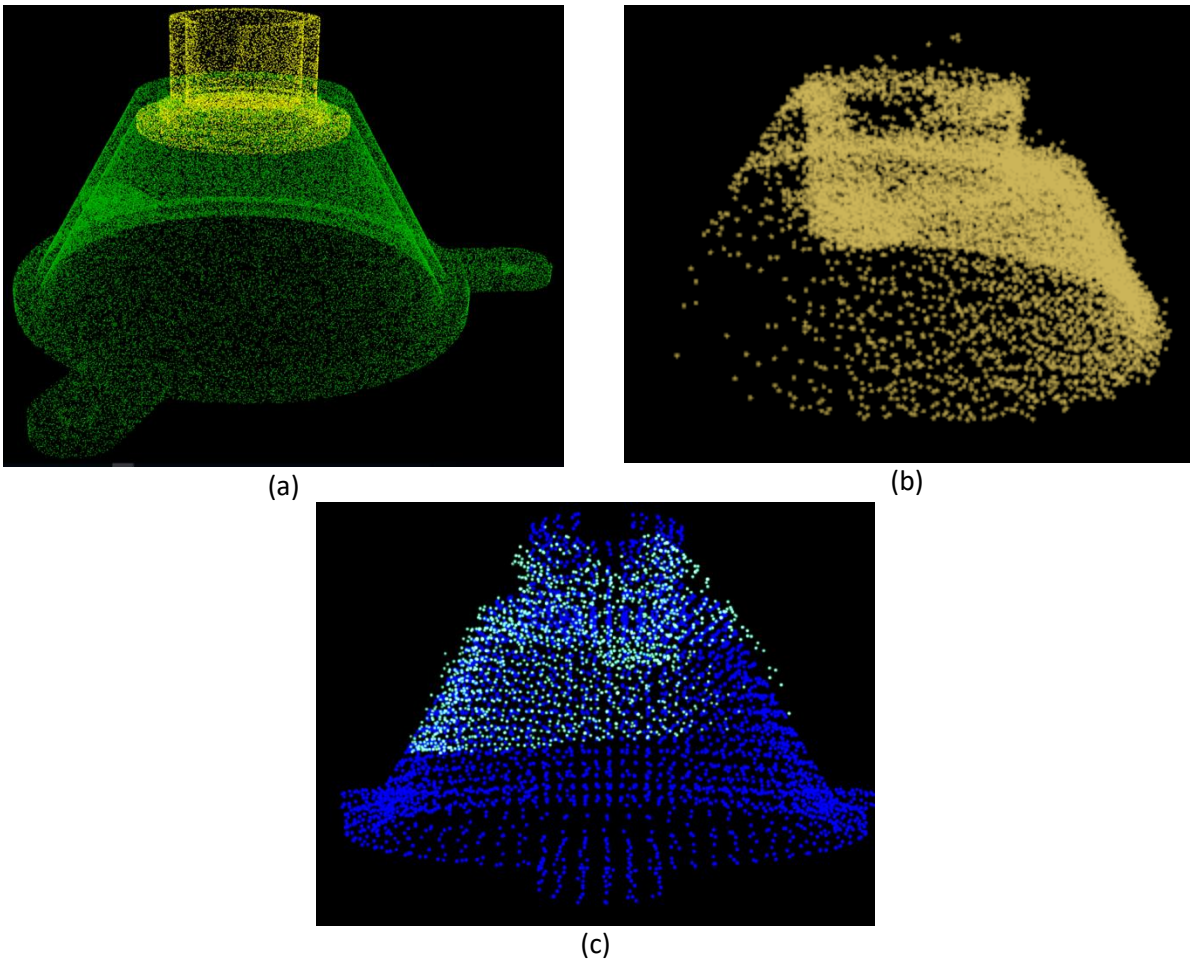


Figure 6 Representations of a bench vise part (a) Point cloud of the CAD part, (b) Depth camera generated point cloud for the part after noise filtering, (c) Merged and aligned CAD and depth camera cloud

AREDA Features and Attributes - User detection and tracking

The expert's handling of parts and start/stop commands for each assembly step must be identified and tracked. The first function of this component utilizes color recognition to identify the hand positions, orientations, and movement paths (Kakumanu et al., 2007). A pre-defined color was used as a seed to initially match the user's hand color in the point cloud output. In order to track multiple users with different skin colors, gloves, or long sleeves, multiple colors are represented as "hypotheses". Each hypothesis was modeled as a Gaussian in order to incorporate noise and uncertainty. All hypotheses were solved using a newly developed Gaussian Mixture Model (GMM) coordinated by using particle swarm optimization (PSO). This PSO-GMM method was developed and tested by the proposers (Bhattacharya & Winer, 2015; Kalivarapu et al., 2009; Kalivarapu & Winer, 2014) and allows multiple Gaussian distributions to be created and solved in real-time, eliminating the need for user input. Results from the method are shown in **Figure 7**a & b. Refinement of this method was required to overcome a potential obstacle in object detection. During assembly operations, the expert's hands will typically cover significant portions of a part or assembly. In this case, object tracking cannot be maintained by ICP techniques alone, as not all of the

part is “visible” in the point cloud. By using the PSO-GMM, additional position and orientation information is available. When merged with the ICP tracking data, overall tracking of parts was maintained, whether occluded by the expert’s hands or not (**Figure 7**c & d). This was one of the primary applied research contributions of the AREDA system.

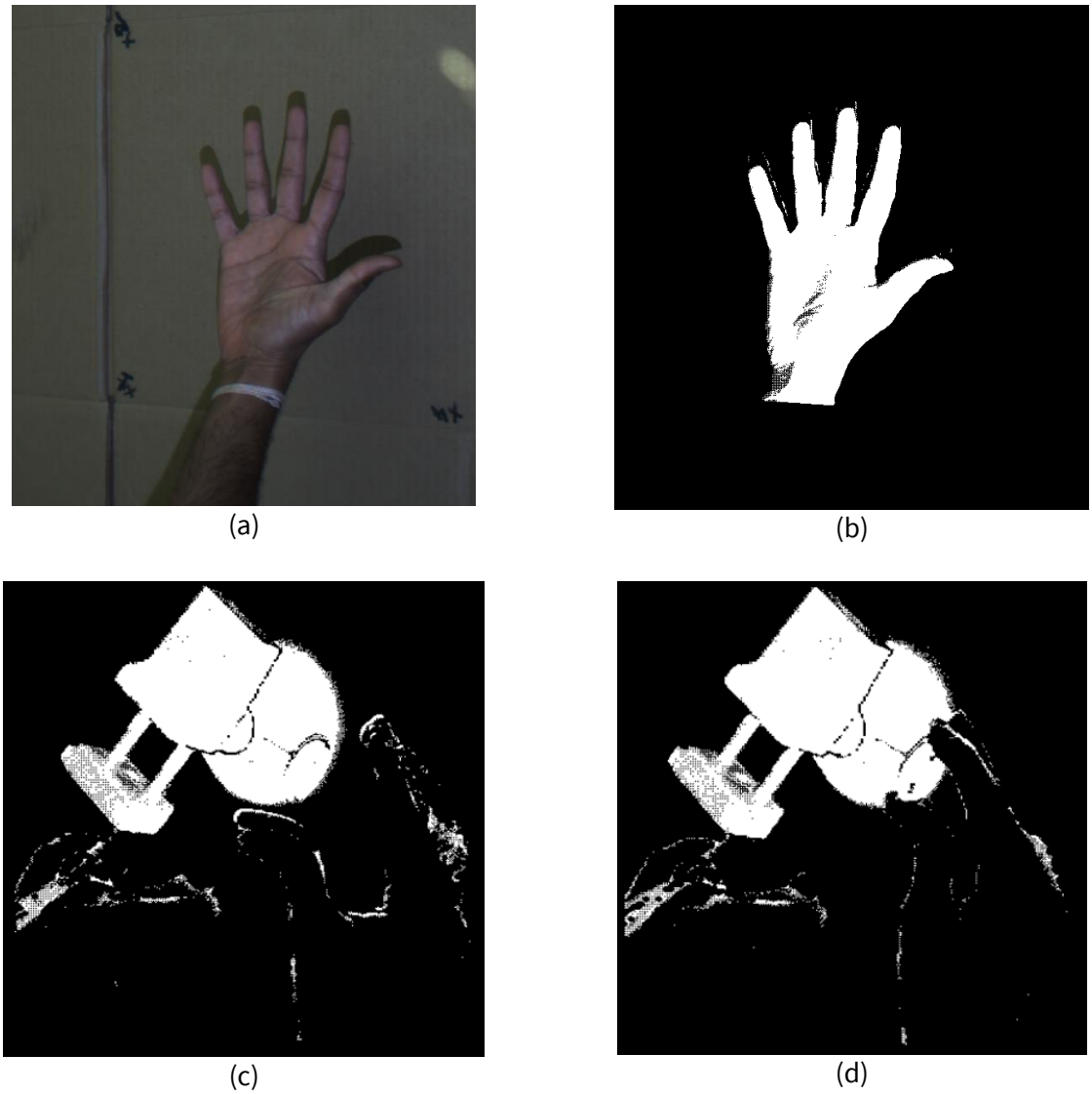


Figure 7 Hand detection example: a) Hand capture footage by the depth camera, (b) Skin tone processed, (c) Skin tone identified and filtered from an assembly using PSO-GMM, (d) Assembly components tracked with skin filtered out in the foreground

AREDA Modes of operation & Software Development Document

This is described in detail in the developer documentation attached in the Appendix section.

AREDA Users and Use Cases

Through prolonged discussions and iterative modifications, the project partners arrived at a consensus on the intended users of AREDA. With user personas ranging from beginner assembly operators through experienced SMEs and IT administrators, a number of possible use cases were explored. Five user

personas were identified, each having skills in certain areas and were motivated to improve the assembly operations in their line of work. The development of user personas was addressed early on in the project and played a significant role in driving the design and development of AREDA and described below.

1. Tech Evaluator

Persona details:
Name: Tom Atkins
Age group: 22 – 25
Job Title: Engineering Analyst
Education: B.S. Mechanical Engineering
Primary Proficiency: Technical blueprints and assemblies

Bio:
Tom is a 24-year-old Engineering Analyst who works in a testing and training group that evaluates technology for near-term implementation. He has 2+ years of work experience. He is mainly involved in developing work instructions to be used by the factory workers. In school, Tom had a brief exposure to AR and is excited about it because he believes that AR technology could benefit his goals. His company wants to explore AR and its potential uses.

Technology Experience:

a. Microsoft Office and Office like products
b. Movie Maker
c. CAD Solid Works

Goals:

a. Create efficient work instructions that are used and understood by the factory workers
b. Improve performance and assembly time

Frustrations:

a. Inability to transfer assembly knowledge easily to factory workers
b. Factory workers not using developed work instructions appropriately

Attributes:
Interested in Augmented Reality applications

2. Process Leader

Persona details:
Name: Diana Wong
Age group: 35 – 50
Job Title: Product Manager
Education: B.S. in Mechanical Engineering, M.S. in Mechanical Engineering
Primary Proficiency: Cutting edge assembly processes and content development

Bio:
Diana is a 41-year-old Product Manager of a large international manufacturing company. She has worked with different types of 3D modeling software and PLM visualization tools for over 15 years. Diana, as well as her company, has just begun to notice new technologies like VR and AR. Diana has reserved excitement

about the possibility of bringing AR into manufacturing and its potential to improve the manufacturing process.

Technology Experience:

a. Microsoft Office and Office like products
b. Siemens PLM Software
c. CAD Solid Works
d. Autodesk Fusion Lifecycle
e. EnSight

Goals:

a. Improve overall efficiency and performance of product lifecycle without quality degradation

Frustrations:

a. Constant knowledge conflict between generated assembly instructions and factory worker.

Attributes:

a. Content expert
b. Requirements gathering

3. Assembly Expert

Persona details:
Name: Bob Carpenter
Age group: 55 - 65
Job Title: Senior Assembly Engineer
Education: A.S. in Engineering Technology
Primary Proficiency: Product assembly

Bio:
Bob is a 60-year-old Senior Assembly-Engineer that is part of a well-known manufacturing company. He is an expert in his setting (assembly space) and is regularly called upon to instruct new members.

Technology Experience:

a. Microsoft Office (maybe)

Goals:

a. Make sure assemblies are done within the safety and quality guidelines provided
b. Be able to pass on knowledge to the next generation of factory workers

Frustrations:

a. Repetitive instruction, no easy way to show correct ways of assembly without physical demonstration
b. Lot of time wasted in trying to learn new ways to do assembly and has no faith in new technology

Attributes:

a. Low technology acceptance

4. IT Manager

Persona details:
Name: Greg Gale
Age group: 30 - 45
Job Title: System Administrator
Education: B.S. in Information Technology
Primary Proficiency: Information Technology implementation

Bio:
Greg is a 44-year-old System Administrator who is in charge of installation/configuration, operation, and maintenance of hardware and software related infrastructure. When his company wants to implement new software or hardware, he will ensure that all the digital information and networking are available and have the proper permissions. New technology is less likely to be implemented within the company without Greg's approval.

Technology Experience:

a. Knowledge of SAN and Cloud technologies
b. SSL certificate encryption key management and generation
c. In-depth understanding of operating systems

Goals:

a. Implement new technologies within the company while avoiding a decrease in production

Frustrations:

a. Legacy software and hardware is being utilized within the company and is hindering potential productivity increase

Attributes:

a. Technology expert
b. Early adopter

5. Content Consumer

Persona details:
Name: Cindy Robinson
Age group: 18 - 23
Job Title: Production worker
Education: A.S. in Engineering
Primary Proficiency: Factory floor production and maintenance

Bio:
Cindy is 23-year-old Production Worker who is responsible for executing the assembly process. She has to take complex technical drawings, that are given to her, and produce products that fit the specifications. Cindy also performs the first quality check on each product. She has about 3 years of experience and has some knowledge about her duties but she is still relatively new.

Technology Experience:

a. Skilled with assembly equipment

b. Operated various types of welding equipment
c. Worked with various types of sheet metal equipment

Goals:

a. Make sure assemblies are done within the safety and quality guidelines provided
b. Make sure products are correct and accurate in order to maintain or improve productivity and meet daily quota

Frustrations:

a. Changes to an assembly procedure are often hard to understand on paper
b. New technology in the past has had a high learning curve and has not resulted into any gains

Attributes:

a. Optimization focus

V. ACCESSING THE TECHNOLOGY

- No background IP was used in this project.
- A detailed description on the requirements for running AREDA is included in the developer documentation, attached in the Appendix section.

VI. INDUSTRY IMPACT & POTENTIAL

The Gartner market guide for Augmented Reality describes that 80% of the market will be immersive displays by the year 2030. Augmented Reality is on course to become a $120 billion market by the year 2025 as viewed by many leading experts. Current practices in manufacturing companies dominated by shop floor assembly processes are stymied by error rates and warranty costs. The use of advanced AR technologies and computer vision algorithms is a preliminary yet significant step in improving US manufacturing competitiveness.

Given that AREDA is an authoring tool and not limited to a specific product or assembly process, it has the potential to be embraced by any company interested in adopting AR technologies. The fact that other companies and organizations expressed interest in learning about or adopting AREDA during DMDII meetings provides evidence that the developed technology is ready to be embraced on a larger scale. For example, Northrop Grumman expressed enthusiasm in the AREDA product and development more than halfway through the course of the project completion. Commercialization efforts from Design Mill and Daqri beyond the project end date will garner attention from many other industries.

VII. TECH TRANSITION PLAN & COMMERCIALIZATION

AREDA was designed to be a commercially viable product. It started as a TRL 3 at the beginning of the project and closed at the proposed TRL 6-7. To become a fully commercialized TRL 9 product, AREDA will require additional development for a smooth customer experience. This will include working with several commercially available depth cameras (currently AREDA works with two, one research-based and the

Microsoft Kinect), extensive testing of the code for error identification, and optimization for increased efficiency. While much of this was done during the project, the code needs to be tested and refined by a commercial software developer. These steps are critical to increase the chances of commercial success. While the source code for AREDA was developed using C++ programming language, the key technologies used in its development are cross platform compatible and can be adopted within another product with relatively minimal issues.

Project partner DAQRI is an early leader in the commercial AR hardware domain. Design Mill, another industry project partner, is a AR system integrator and services the manufacturing market sector. Both industries have contributed significantly in the development of AREDA tool and have committed plans to commercialize the full AREDA system.

One of the primary barriers to adopting AREDA into a manufacturing process is the lack of awareness of Augmented Reality. Companies deeply vested into traditional manufacturing practices tend to oppose adoption of newer technologies because they do not have the time or resources. This is typically a steep barrier and can be overcome only by a structured dissemination of AR knowledge and their applications. Demonstration of AREDA upon its development to TRL 9 by project partners and other technology companies is hence the best recommended path in helping companies adopt the technology.

VIII. WORKFORCE DEVELOPMENT

While the AREDA system brings AR to the daily workflow of designers, engineers, and manufacturers, it also offers exciting potential educational content to be produced. The development of AREDA now facilitates automatically capturing and processing expert instructions, so it can be extended to the shop floor. The lessons learned in the development of this TRL 7 tool from what started as a laboratory prototype is invaluable to the DMDII partners and the broader industrial base in several ways:

Transitioning funded research to practice – Many DMDII partners fund their own university research projects. While many of these projects show tremendous promise, they often fall short of full implementation into daily workflows. The unique workings of AREDA offered lessons for how university research can be developed simultaneously with commercialization plans. Project meetings and quarterly presentations fostered collaboration among DMDII partners and outside industries.

Expansion beyond shop floor assembly – Expert demonstration to author AR guided instructions on a shop floor addresses a critical need of industry. However, there are other use cases that are closely related that could also benefit from the proposed AREDA system. Small appliance repair shops, auto repair businesses, and even franchised businesses are other examples where consistency in process is important. Whether it is fixing a washing machine or having employees at several franchised locations setting up displays, the use of expert demonstration in an AR environment will dramatically speed up the efficiency of these activities and subsequently lower their costs. While none of the project partners were in this marketspace, the sample assemblies used to test AREDA have the same types of complexities seen in many of these industries. With some additional development effort, AREDA could easily serve these markets.

Authoring by expert instruction for training – From workers to students, training on advanced tools and processes is critical to future individual and corporate success. The AREDA system, and development of

its components, now offers capabilities to produce AR training resources rapidly. Students in a senior design class, for example, in an engineering discipline could use the AREDA system to demonstrate how their product comes together and even functions. Alternatively, professors and professional trainers could produce expertly demonstrated materials covering topics such as fluid flow, basic mechanics, or even learning human or animal anatomy. AREDA's architecture can be leveraged to implement future advancements in object recognition, point cloud matching, and machine learning within AREDA as academic research projects as well.

Documentation, reports, and technical publications detailing the development of AREDA are made available via digital means to DMDII partners and other organizations. Short videos were produced to highlight important educational objectives such as: a) What is Augmented Reality, b) What are point clouds, c) How these technologies are used within AREDA and how they work.

IX. CONCLUSIONS/RECOMMENDATIONS

An Augmented Reality Expert Demonstration Authoring (AREDA) tool was developed to provide a simple and intuitive method for rapidly authoring AR work instructions. This was achieved through tracking and recording the actual part manipulations of an expert using 3D depth cameras with advanced image processing, computer vision and point cloud matching algorithms. The system was based on existing successful research results from the proposers. This project addressed the Advanced Analysis (AA) thrust identified by the DMDII and the DoD as critical to significantly improve US manufacturing capabilities and reduce acquisition costs for the DoD.

Through cost-sharing partnership with aerospace (Boeing) and heavy equipment (Deere) sectors, functional requirements for AREDA were developed. Product requirements based on market opportunities were gathered from other project partners (DAQRI and Design Mill). Inter University collaboration (Purdue) afforded the opportunity to implement cutting edge depth camera sensors for use within AREDA.

AREDA was developed at a TRL 7 to facilitate quick commercialization. This is a key innovation that will accelerate AR adoption in manufacturing, which in turn will reduce time to author work instructions, enhance their communication to the shop floor, reduce process errors, and decrease manual process time – all of which will substantially improve US manufacturing productivity and competitiveness.

X. LESSONS LEARNED

One of the challenges in developing AREDA was the use of an appropriate depth sensor for capturing point clouds of assembly work area and the assembled components. AREDA was developed for use with two depth sensors:

a. Microsoft Kinect
b. Point Grey Camera custom built for use as a Structured Light System

The Microsoft Kinect was fully supported due to its use with the Xbox game console. However, their product offering packaged with the game console ended recently. As such, the camera can now primarily

be purchased as an after-market or a third-party accessory product. Although Microsoft continues to support and provide software development kits (SDK) for use with the Kinect, it is likely this product line and support may end. In addition, the Kinect camera's resolution is less than current competitors in the market. So, while the Kinect was good for implementation and testing, it is not recommended in a commercial version of AREDA.

Point Grey camera solves some of the issues posed by the Kinect. For example, the resolution is much higher and is a commercially supported camera. Because it was custom built with a projection system to use structured light. Thus, the camera's hardware and software were modified. On the other hand, the structured light system has its own deficiencies for general use. This, combined with the modifications made for the project, make it not suitable for a commercial version of AREDA.

Depth camera technology is still in infancy and nowhere close to saturation. Choosing a good depth camera and a supported SDK is critical in further developing AREDA. AREDA was designed with modularity to integrate new cameras into the framework, and the developer documentation attached in the Appendix section has more information about how to integrate a new camera.

AREDA in its current form can be slow in processing and recognizing parts. This is primarily because point clouds generated by the depth camera for each identified assembly step are matched against every part uploaded to the AREDA part library in a serial manner. That means that the camera point cloud is matched against only one part at any instance during part processing stage. Implementing parallelization schemes, where camera point clouds can be matched against multiple parts in the library, can improve the speed of part matching.

Some depth cameras, such as the Kinect sensor, are light sensitive. As such it is important that an appropriate lighting condition be identified early on and assembly work area be adjusted accordingly.

Other minor issues during the course of AREDA development are listed in the developer documentation, attached in the Appendix section.

XI. DEFINITIONS

What follows are a set of definitions, terms, and acronyms used in this document. These definitions were gathered from various source including the internet, reference papers, standards organizations, and the authors of these document.

API: Application Programmer's Interface
AR: Augmented Reality
AREDA: Augmented Reality Expert Demonstration Authoring
CAD: Computer Aided Design
HCI: Human-Computer Interaction
SDK: Software Development Kit
SME: Subject Matter Expert
TRL: Technology Readiness Level

XII. REFERENCES

Azuma, R., 1997, “A Survey of Augmented Reality,” *Presence: Teleoperators and Virtual Environments*, Vol. 6, pp. 355–385, August

Besl, P.J. and McKay, N.D., 1992, “A method for registration for 3-D shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No.2, pp. 239-256

Bhattacharya, B. and Winer, E., 2015, “A Method for Real-time Generation of Augmented Reality Work Instructions via Expert Movements.” *Proceedings of the 27th IS&T/SPIE Electronic Imaging Conference*, San Francisco, CA, February

Caudell, T. and Mizell, D., 1992, “Augmented reality: An application of heads- up display technology to manual manufacturing processes.” *Proc. Int. Conf. on System Sciences*, Vol. 2, pages 659–669, Hawaii

Chen, Y. and Medioni, G., 1991, “Object modeling by registration of multiple range images,” *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 2724–2729

Davidson, K., “Manufacturing Technology and the Aging Workforce”, Automation World, http://www.automationworld.com/workforce-development/manufacturing-technology-and-aging-workforce, accessed March 2015

DeLong, D.W., 2004, *Lost Knowledge: Confronting the Threat of an Aging Workforce*, Oxford University Press, New York, NY

Department of Defense, *Technology Readiness Assessment (TRA) Guidance*, April 2011

Hagman, E., “The Gray Shift Part 2: Using Ergonomy To Retain Valuable Older Workers”, Manufacturing.net, http://www.manufacturing.net/articles/2013/09/the-gray-shift-part-2-using-ergonomy-to-retain-valuable-older-workers, accessed March 2015

Kakumanu, P., Makrogiannis, S., and Bourbakis, N., 2007, “A survey of skin-color modeling and detection methods,” *Pattern Recognition*, 40(3), Marxh, pp. 1106–1122

Kalivarapu, V., Foo, J.L., and Winer, E.H., 2009, “Improving Solution Characteristics of Particle Swarm Optimization using Digital Pheromones”, *Journal of Structural and Multidisciplinary Optimization*, 37(4), 415-427

Kalivarapu, V., and Winer, E., 2014, “A Study of Graphics Hardware Accelerated Particle Swarm Optimization with Digital Pheromones,” *Journal of Structural and Multidisciplinary Optimization*, 36(4), 692-702

Li, B. and Zhang, S, 2014, “Structured light system calibration method with optimal fringe angle,” *Applied Optics*, 53(13), 7942-7950

Laughner, J. I, Zhang, S., Li, H., Shao, C. C. and Efimov, I. R., 2012, “Mapping cardiac surface mechanics with structured light imaging,” *American Journal of Physiology: Heart and Circular Physiology* 303(6), H712-H720

Mehta, R. P., Zhang, S., and Hadlock, T.A., 2008, “Novel 3-D video for quantification of facial movement,”

Otolaryngol Head Neck Surg., 138(4), 468-472

Neumann, U. and Majoros, A., 1998, "Cognitive, performance, and systems issues for augmented reality applications in manufacturing and maintenance. *Proceedings or Virtual Reality Annual International Symposium*, Atlanta, GA, USA, March

Nakanishi, M. and Sato, T., 2012, "Application of Digital Manuals with a Retinal Imaging Display in Manufacturing: Behavioral, Physiological, and Psychological Effects on workers," *Human Factors and Ergonomics in Manufacturing & Service Industries*, Vol. 25, No. 2, pp. 228-238

Ou, P., Li, B., Wang, Y. and Zhang, S., 2013, "Flexible real-time natural 2D color and 3D shape measurement," *Optics Express*, 21(14), 16736-16741

Radkowski, R., Herrema, J. and Oliver, J.H., 2015, "Augmented reality based manual assembly support with visual annotations for different degrees-of-difficulty," *Journal of Human-Computer Interaction*, published online January 2015, DOI:10.1080/10447318.2014.994194 [in press]

Richardson, T., Gilbert, S., Holub, J., Thompson, F., MacAllister, A., Radkowski, R., Winer, E., Davies, P., and Terry, S., 2014, "Fusing Self-Reported and Sensor Data from Mixed-Reality Training," *Proceedings of Interservice/Industry Training, Simulation & Education Conference* (I/ITSEC), Orlando, FL, December 2014, Paper no. 14158

Ridene, T. and Goulette, F., 2009, "Registration of fixed-and-mobile-based terrestrial laser data sets with dsm," *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation* (CIRA 09), pp. 375–380.

Rusu, R.B. and Cousins, S., 2011, "3D is here: Point Cloud Library (PCL)," *IEEE International Conference on Robotics and Automation (ICRA),* vol. 1, no. 4, pp. 9-13, Shanghai, CN, May

Sausman, J., Samoylov, A., Regli, S. and Hopps, M., 2012, "Effect of eye and body movement on augmented reality in manufacturing domain." *IEEE International Symposium on Mixed and Augmented Reality 2012*, pages 315–316, Atlanta, GA, USA, November

Schnabel, R., Wahl, R. and Klein, R., 2007, "Efficient RANSAC for Point-Cloud Shape Detection", *Computer Graphics Forum*, 26, pp. 214–226

Tang, A., Owen,C., Biocca, F. and Mou, W., 2003, "Comparative effectiveness of augmented reality in object assembly." *Proceedings of the Conference on Human Factors in Computing Systems (CHI '03)*, Fort Lauderdale, FL, USA, April

Wiedenmaier, S., Oehme, O., Schmidt, L. and Luczak, H., 2003, "Augmented reality (ar) for assembly processes design and experimental evaluation." *International Journal of Human-Computer Interaction*, 16(3): 497–514

Zhang, S., 2010, "Recent progresses on real-time 3-D shape measurement using digital fringe projection techniques," *Opt. Laser Eng.*, 48, 149-158

Zhang, S. and Huang, P.S., 2006, “High-resolution real-time three-dimensional shape measurement”, *Opt. Eng.,* 45(12), 123601

XIII. APPENDICES

Appendix A – Developer Documentation

Augmented Reality Expert Demonstration Authoring (AREDA)

Developer Documentation

Iowa State University
Virtual Reality Applications Center
537 Bissell Rd.
Ames, IA 50011-1096
POC: Eliot Winer
ewiner@iastate.edu
(515) 450-1077

Additional Project Participants:
Purdue University
The Boeing Company
Deere & Company
DAQRI
Design Mill, Inc.

Table of Contents

1. What is AREDA?

AREDA stands for Augmented Reality Expert Demonstration Authoring. The goal of AREDA is to provide a simple and intuitive method for rapidly authoring AR work instructions in a manufacturing assembly process. Traditionally, assembly line operators follow written instructions in performing assembly steps and tend to have a high rate of errors leading to expensive re-work or warranty costs. AREDA aims to reduces such errors by visually generating step-by-step work instructions on a display through the use of current day technology such as 3D depth cameras, advanced image processing and computer vision algorithms.

For an assembly operation, AREDA aims to superimpose video of work area with computer-generated visual features that provide instructions on assembly operations, such as which part to pick next, where to assemble a particular part, or which tool to use. An expert trains AREDA with an assembly operation, which includes steps such as: a) Background-area-skin calibration, b) CAD and point cloud part library uploads, c) Record a sequence of assembly steps, d) Automatically generate assembly steps while matching the parts recorded by the camera against the uploaded CAD parts. In addition, the application also allows modifying automatically detected parts if there were errors and animation sequences in assemblies. Once the expert generates these assembly instructions within AREDA, they can be passed on to an assembly operator who can visually see and follow the steps for assembly operations.

2. Compiling and Building AREDA

A. Obtaining Source Code

The source code for AREDA is available for download as two zip files:

- AREDASrc.zip (source code)
- AREDADeps.zip (dependencies)

Both the zip files are available on cyBox at:

https://iastate.box.com/s/vims64ow77wmghlrfa4ftu3ls0hjm658

B. Development Environment Requirements/Recommendations

Requirements

a. Operating System: Windows 10, 64-bit
b. Integrated Development Environment (IDE): Microsoft Visual Studio 2013 (C++)
c. USB 3 Port: Depth cameras require plugging into a USB 3.0 port on the host PC.

Notes about IDE:

- Microsoft Visual Studio 2013 was used in the development of AREDA. Earlier versions of point cloud libraries (PCL, a dependency for AREDA), had incompatibilities with newer versions of Visual Studio. Newer releases of PCL solved the incompatibility issues and using them along with a newer version of Visual Studio should work. However, this was not tested at ISU. All the AREDA dependencies listed on cyBox share above were built for Visual Studio 2013. Therefore, all the dependencies must either be downloaded or compiled again if a newer version of Visual Studio were to be used.

- Visual Studio 2013 is available only as a 32-bit IDE, but it supports both 32-bit and 64-bit compilation. AREDA code and all dependencies must be built with a 64-bit compiler tag at the initial phase while generating visual studio project files using CMake. More will be described in detail in the 'Compiling AREDA' section.

Recommendations

d. CPU: Best and fastest available CPU highly recommended. Depth processing is computationally intensive. AREDA usage and interactivity is directly dependent on how fast the host CPU is.

e. Solid State Storage: SSDs are indispensable for compilation and execution. AREDA code was developed using at least a 512GB SSD. 1TB SSD is recommended for development and testing.

f. Video Card: A high-end video card is recommended (e.g., Nvidia GeForce GTX 1080). AREDA has never been tested on AMD based GPUs.

Recommended Software Utilities

g. Git Bash: Latest release of Git bash available here: https://git-scm.com/downloads. This is useful if version controlling is used for further code development. SourceTree GUI is recommended for git version control and KDiff3 diff tool/merge tool along with it for code conflict resolution.

h. CMake-gui: Latest release of CMake-gui available here: https://cmake.org/download/.

i. Image Watch Utility within Visual Studio: Helpful tool to view OpenCV image types (e.g., `cv::Mat`) during run time in Debug mode.

j. Sqlite database connectivity for visual studio 2013: Simple utility that shows how and where intermediate files are stored within an sqlite database. Useful in debug mode.

C. Environment Variables

Environment variables for various dependencies need to be setup before AREDA's Visual Studio project files can be generated using CMake. The software dependencies are described in detail in the next section. Please set the following environment variables.

```
AREDADEPS: C:\AREDADeps
ARTOOLKIT_DIR: %AREDADEPS%\ARToolKit-2.72
OpenCV_DIR: %AREDADEPS%\OpenCV-3.1.0
OSG_DIR: %AREDADEPS%\OpenSceneGraph-3.2.1
FBX_LIB: %AREDADEPS%\FBX\2017.1\lib
PCL_DIR: %AREDADEPS%\PCL 1.7.2
PCL_ROOT: %AREDADEPS%\PCL 1.7.2
OPENNI2: %AREDADEPS%\OpenNI2
OPENNI2_INCLUDE64: %AREDADEPS%\OpenNI2\Include\
OPENNI2_LIB64: %AREDADEPS%\OpenNI2\Lib\
OPENNI2_REDIST64: %AREDADEPS%\OpenNI2\Redist\
QTDIR: %AREDADEPS%\QT-5.7.0\5.7
VTK_DIR: %AREDADEPS%\PCL 1.7.2\3rdParty\VTK
POINTGREY_DIR: C:\Program Files\Point Grey Research\FlyCapture2
PATH:
%PATH%;%OpenCV_DIR%\x64\vc12\bin;%OSG_DIR%\bin;%QTDIR%\msvc2013_64\bin
;%PCL_ROOT%\bin;%PCL_ROOT%\3rdParty\VTK\bin;%ARTOOLKIT_DIR%\bin;%OPENN
I2%\Redist;
```

Kinect SDK installs the necessary system environment variables during the installation process, so a separate set of user environment variables will not be necessary. It is recommended that all AREDA dependencies are installed/copied at the root level on the Windows computer (e.g., `C:\AREDADeps`) so anyone using the PC can have access to the dependencies by setting environment variables listed above and will not need individual copies of all dependencies.

D. AREDA Software Dependencies

AREDA has been compiled with specific versions of software libraries and they are listed below. It is possible newer versions may work, but they have not been tested.

a. **ARTool Kit version 2.7.2:** As of writing this documentation, the newest ARTool kit version available is 5.3.2. Between major releases, ARTool Kit's function calls have changed substantially. Hence, the latest version will most likely not work without making changes within AREDA source. The use of AR libraries is fairly minimal and the code mainly uses ARToolkit functionality to identify AR markers. So, v2.7.2 of ARTool Kit is recommended and the AREDA dependency zip file (described in Section 2 A. Obtaining Source) contains the required source code to build the libraries and binaries. Please note that this version of ARToolkit does not have a CMakelists file to generate Visual Studio projects. Therefore, Visual Studio project creation will have to be a manual effort. It is a fairly straightforward process. Use batch build mode to build 64-bit version of the library files: a) `libAR.lib` (release), and b) `libARd.lib` (debug). Then, copy these files manually to `%ARTOOLKIT_DIR%\lib`. Additionally, copy the entire '`include`' folder within the ARToolkit source directory and paste it in: `%ARTOOLKIT_DIR%\include`.

b. **Autodesk FBX SDK:** This SDK will be needed by OpenSceneGraph (see next section) to load/export .FBX geometry. This can be obtained from Autodesk's website: http://usa.autodesk.com/adsk/servlet/pc/item?siteID=123112&id=26012646. The VS2013 version is the one tested and used by AREDA, although if AREDA is migrated to a newer Visual Studios, the corresponding SDK must be used.

c. **Kinect SDK 2.0:** This can be downloaded from the Microsoft's website: https://www.microsoft.com/en-us/download/details.aspx?id=44561. Install it to `C:\Program Files` directory. AREDA code was initially developed with Kinect sensor (the one that used to come bundled with Xbox One). So, there's some AREDA code that uses Kinect SDK's data structure definitions for handling depth information regardless of the depth camera used. Therefore, PointGrey SLS camera implementation also uses Kinect's data structures. Hence, this SDK is currently required even if a Kinect camera is not used. It can be fixed by creating a separate data structure instead of '`CameraSpacePoint`' defined in `Kinect.h`. This seems to be the only dependency to using Kinect SDK for a non-Kinect depth camera. The data structure is very simple and it includes three float members x, y, and z. If this data structure is defined somewhere else, non-Kinect depth cameras will then not require the Kinect SDK to be installed on the PC. While this is a solvable issue, it did not go up the priority level during the course of AREDA development. Fixing this issue is recommended for future development.

d. **OpenCV 3.1.0:** Download OpenCV from http://opencv.org/downloads.html. This version of OpenCV comes with a CMakeLists file and can be used for generating visual studio 2013 x64 project file using CMake. Within CMake, use default settings. However, point `CMAKE_INSTALL_PREFIX` to `%OpenCV_DIR%`.

e. **OpenSceneGraph, OSG 3.2.1:** OpenSceneGraph (OSG) is the primary graphics rendering engine used within AREDA. The API manages loading, rendering, and exporting 3D geometry objects within AREDA

Qt UI. The default geometry file formats OSG can support are `.osga`, `.osgb`, `.ive`, `.obj`, and `.3ds`. Support for other file formats can be achieved by compiling OSG with appropriate third-party plugins. AREDA requires `.FBX` plugin support within OSG for project export into a Unity game engine based viewer. Our codebase was tested with Autodesk FBX SDK (2017.1) and worked with Microsoft Visual Studio 2013. At the time of writing this documentation, the latest release of OSG is 3.4.1, but was not tested with Visual Studio 2013 and the latest version of Autodesk FBX SDK.

f. **Point Grey FlyCapture SDK:** This SDK is required for the SLS point grey camera system to work. It can be downloaded from here: https://www.ptgrey.com/flycapture-sdk. This SDK is not required if the depth camera used with AREDA is a Kinect (or any other depth camera). The SLS camera system was custom designed and built by Dr. Song Zhang, Purdue University. Below are the camera specs used in the SLS system:

```
Grasshopper3 GS3-U3-23S6C
Serial # 1637412
```

These specifications are important because the drivers are specific to the camera. Use the above numbers to identify the driver that should be downloaded and installed. The SDK comes with a sample program '`Point Grey FlyCap2`'. This program can connect to the Point Grey camera and show the capture in a window. If this works properly, then it means that the camera installation went smooth and can be used within AREDA.

g. **Point Cloud Library, PCL 1.7.2**: Point Cloud Library (PCL) is a very important requirement for AREDA. Point clouds for CAD models are compared against point clouds generated by the depth cameras to perform part matching. As of writing this documentation, the latest PCL release is v1.8.1. PCL 1.7.2 was tested with AREDA without issues. PCL has a number of dependencies for compilation. Compiling PCL from source and all their dependencies is too tedious. A third-party user packages PCL and all their dependency binaries together within a simple installer executable. Download the version of pre-built binaries that corresponds to Visual Studio 2013 from: http://unanancyowen.com/?p=1255&lang=en.

h. **Qhull, Flann, Boost, OpenNI2:** Qhull, Flann and Boost are PCL dependencies and the third-party URL in the Point Cloud Library section above takes care of them. Hence, a separate download and compilation is not necessary. During the third-party library installation, `PCL_ROOT` directory must be specified, so they are installed relative to the PCL paths. OpenNI2 is another PCL dependency for AREDA and a 64-bit version should be downloaded and installed from https://structure.io/openni

i. **Qt 5.7:** All UI events are managed by Qt. During AREDA development phase, the Open Source version of Qt was used and can be downloaded from here: https://www.qt.io/download. Please make sure you download Qt version specific to Visual Studio 2013. Qt source code download is not needed, but it provides a wizard allowing the user to select which version of Qt is required. The wizard can be executed again for installing additional components/updating existing ones. **Figure 8** shows the options that were selected when installing Qt for use with AREDA. Qt does not like spaces in folder names. It is highly recommended that Qt and all dependencies do not have spaces in folder/directory names.

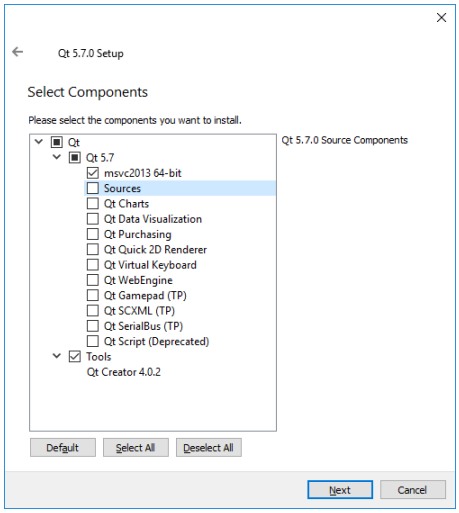


Figure 8. Qt 5.7.0 set up for AREDA

Of course, all these dependencies compiled/downloaded/built for Visual Studio 2013 are available in the zip file in the cyBox folder indicated in section 2A – Obtaining Source Code. The only dependency that was not included in the zip file is the Microsoft Kinect SDK v 2.0, which must be downloaded and installed separately.

E. AREDA Code Base Directory Structure

Here is the overall directory structure of AREDA code base when pulled from the git repository.

```
AREDA/
   .git/
   CMakeModules/
   data_art/
   fonts/
   include/
   resources/
   shader/
   src/
   ui/
   CMakeLists.txt
   phaseMin_FP380_zmin600.raw
   README.docx
```

- `data_art` directory contains basic image and AR pattern utility files that AREDA uses.
- `fonts` directory contains .ttf files used for font rendering within AREDA.
- include and src contains header files and source files respectively.
- `resources` contain images that Qt UI uses.
- `shader` folder contains glsl vertex and fragment shader files for rendering geometric objects
- `ui` contains files created/generated with Qt and relates to the graphical user interface

- `CMakeLists.txt` is a file required by CMake for visual studio project generation
- `phaseMin_FP380_zmin600.raw` is a raw image file needed by the point grey camera for fringe image calculations
- `README.docx` is a preliminary manual for AREDA. It was created a while ago and is not up to date.

F. Compiling AREDA

Visual Studio project files are generated using CMake. Ensure the latest version of CMake is installed on the computer. To generate visual studio project files, CMake will need a `CMakeLists.txt` file, which is provided with AREDA source code. In the same directory where CMakeLists file is located, create a build directory where Visual Studio project files are to be created as well as where binaries will be built (e.g., myBuild). Within CMake, specify the following options:

`Where is the Source code`: Path-to-directory-where-CMakeLists.txt-file-is-located
`Where to build the binaries`: Path-to-myBuild-folder

Next, configure '`Visual Studio 12 2013 x64`' as the build environment with '`default native compilers`' build option. CMake checks against dependencies and will generate Visual Studio project files if all goes well. Address any errors shown in the CMake window before opening the AREDA visual studio project.

Within the AREDA Visual Studio project, look for 'Solution Explorer' and set 'AREDA' as startup project. Select the build option as 'Release' and not 'Debug'. 'Debug' mode is recommended while making code changes and testing. Debug builds are generally (and sometimes excruciatingly) slow in execution. Sometimes, building a project in Release mode and then executing AREDA in Debug mode helps with speed.

Multi-threaded compilation did not work during AREDA development and is NOT recommended for building AREDA, unless the underlying issues are resolved.

G. AREDA Installer

A standalone AREDA executable was never built during AREDA's development process. The project has always been executed via the Visual Studio IDE. Therefore, a formal installer was never built. Nullsoft Scriptable Install System (NSIS) (nsis.sourceforge.net) is a simple to use scriptable installer and is recommended for building an AREDA installer.

H. AREDA Projects

A folder titled 'default' (if it doesn't already exist) is created within the directory of Visual Studio solution file when AREDA is executed via Visual Studio IDE. Below is the folder structure of a typical AREDA project, and their descriptions are indicated in green color:

```
default/                    //AREDA's default project folder
      Processed/            //Contains animations, Skin and mask files
            Animations/
                  *.ive     //OpenSceneGraph readable animation files
            Intermediate/
                  Mask*.png //White areas in the image indicate parts
```

```
            Skin*.png   //White areas indicate detected skin
      Layers/           //Unused
      Models/           //Contains uploaded part files
            Model*.ive  //Files converted from CAD format to .ive
            PAT*.ive    //Position attribute transform for models
      AREDA1.db         //SQL database with paths to parts

Recorded/                     //Data captured by the camera
      Area/                   //Work area
            Data*.raw         //K-frame depth & color data
            PartModel*.ive    //Uploaded CAD files
            PartModel*.pcd    //Uploaded point cloud files
      Background/
            Data*.raw         //K-frame depth & color data
      Skin/
            *.png             //Skin files
            *-m.png           //Skin mask files
      Data*.raw               //K-frame depth & color data

Project_settings.ini          //Project path settings
RecordedBGPlane.png           //Background work area plane
```

The 'default' project folder is always loaded upon executing AREDA from within Visual Studio IDE. Any change made within AREDA (calibration, part library, assembly recording, processing, or refinement) overwrites existing 'default' project contents without additional prompts in the UI. To save/archive an existing project, the 'default' folder should be saved under a different name elsewhere on the computer.

I. Multi-threaded Code Execution

AREDA was designed to run on a single thread, not because of a requirement, but out of convenience. Over the course of code development, various sections of the code are made to execute multi-threaded (e.g., point grey fringe image calculations), but AREDA can benefit from more multi-threading.

For example, each step identified within AREDA has an associated point cloud generated by the camera. This point cloud is matched against every CAD part (and its point cloud) uploaded to the library in a serial manner. As such, part matching is time consuming and a computationally intense process. This can easily be done as a multi-threaded implementation, because part matching of camera point cloud against a CAD cloud file is independent of another CAD cloud file. Multi-threading can speed up the overall time taken for point cloud matching.

J. Hardware Requirements

AREDA currently supports two depth camera sensors:

- Kinect sensor (the one that used to be bundled along with Xbox One game console)
- A custom built structured light system using Point Grey depth camera

A high-resolution depth camera sensor capturing/providing depth information in real-time is an ideal requirement for AREDA. Due to the unavailability of such a depth camera in the market, AREDA was developed using what is available with acceptable tradeoffs. Kinect camera is cheap and works very quick. However, it is low resolution and sensitive to light conditions. This was the first camera AREDA was developed around.

Point Grey depth camera encased in a structured light projection system was developed by Prof. Song Zhang's team at Purdue University, a partner in this project. The camera captures depth and color with great precision and is many times better in resolution compared to the Kinect. The camera is capable of capturing images at 1280 x 960 resolution. However, it is slow. The camera system requires six frames (called fringes) to create an acceptable depth frame. Capturing six consecutive frames and processing them for depth at a high resolution makes it computationally intensive. GPU processing for depth information was considered during AREDA development, but data traversal between GPU and CPU every video frame dramatically reduced any time gains from using a GPU. For example, below is a theoretical data bandwidth required for data traversal between GPU and CPU for a 3-channel image at 1280 x 960 resolution, captured at 15 frames per second:

((4-byte x 3 (channels xyz) + 4-byte * 3 (channels rgb) + 1-byte mask) x 1280 x 960 x 15 FPS)/1024)/1024) = ~440 MB/sec.

This amount of data transfer from GPU to the CPU via the motherboard bus causes bandwidth bottlenecks leading to data loss for frames. For this reason, depth processing was implemented on the CPU and not the GPU.

3. AREDA Code Structure and Components

A brief outline of AREDA steps is given below, followed by more detailed dev information

1. Depth camera selection from a drop-down to capture color and depth images
2. Calibrate the background, work area, and skin tone
    a. Background calibration establishes the perimeter within which an assembly is performed
    b. Work area calibration is to establish a base coordinate system on the work area
    c. Skin tone calibration trains AREDA to filter out the user's skin tone from depth images
3. Create a part library of CAD and their corresponding point clouds to perform a match with the point clouds generated by the depth camera. This is used for identification of parts in each assembly step from within the part library
4. Record an assembly by the depth camera
5. Process the captured images for generating steps, identify parts and animations
6. Refinement phase to make edits to parts that were identified incorrectly or to alter their positions and animation directions

Figure 9 below shows the startup screen of AREDA. A user can either use the wizard mode 'Beginner view (step-by-step Guide)' or advanced mode to proceed to the next steps. A wizard mode shows a sequence of button presses that a user should perform during AREDA usage and is unidirectional in nature. Therefore, some buttons show up as inactive and will not allow the user the flexibility to go back and forth. The 'Advanced View' mode allows the user more interactivity and flexibility.

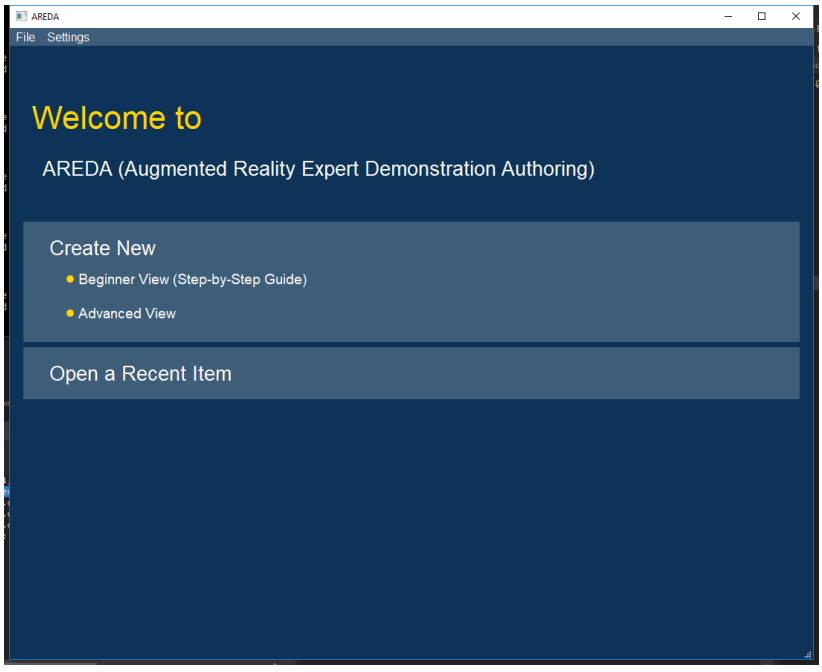


Figure 9 AREDA Start up view

A. Connect Camera

A camera selection drop down menu lets a user select which camera to use with AREDA. The `init` function in `inputmanager.cpp` allows adding cameras to the list of dropdowns. Files `dev_num` in `baseinput.h` let the developer specify what the text in the dropdown should read as.

A Kinect depth camera and an SLS Point Grey camera system were implemented within AREDA. This section describes an overview on how they were implemented, and the general procedure can be used for implementing any other depth camera. The Kinect camera was implemented within AREDA as kinectinput.cpp/.h and SLS camera was implemented as `slscamerainput.cpp/.h`. Depending on the drivers for the depth cameras, there may be other source codes and headers that need to be used along with the `*input.cpp/.h`. For example, the Kinect SDK is self-contained, and any depth/color information can be obtained by invoking calls to Kinect SDK functions directly. The Point Grey camera is attached to a projector for capturing and processing surface deformations via fringe images. While the driver for the Point Grey camera was built by flir systems (ptgrey.com), the SDK to capture and process fringe images via the projector was custom built by Purdue University. These fringe images were processed separately and come with a number of other source/header files, as described in SLS camera section below.

a. Kinect camera

AREDA supports the Kinect for Windows v2 sensor. This sensor has significantly more noise than the SLS camera, but is significantly faster. With that said, the distance threshold in the background calibration step should be higher than 0.015 (1.5 cm). In addition, the assembly should be located in the working range of the sensor, which is 0.4 – 4.5 meters.

Checking camera connectivity

The Kinect for Windows v2 sensor requires a USB 3 port. On many computers, USB 3 may be disabled in the BIOS, even if it is supported by a particular port, so you may have to check that if you encounter problems.

Plug in the Kinect camera, and launch the Kinect Studio v2.0 application that comes with the Kinect SDK. If the sensor is installed correctly, you can verify its connectivity by clicking the *Connect to service* button in the top-left corner, and viewing live camera data in the main window. See **Figure 10** below.

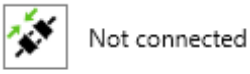


Figure 10. Kinect camera connectivity

Connecting to the camera in AREDA

Connecting to the sensor in AREDA is performed by selecting *KINECT* in the drop-down menu, and clicking the *Connect* button. If the connection is successful, you will see live, streaming video in the window.

Management of the Kinect camera is handled in: `kinectinput.cpp/.h`

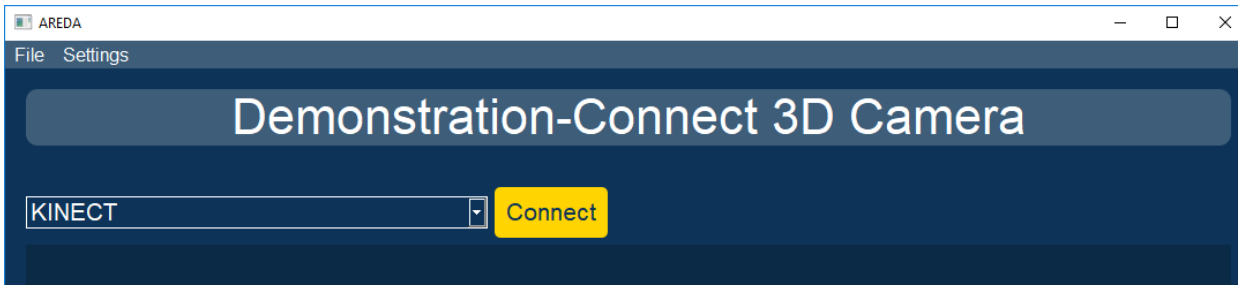


Figure 11. Camera connection dropdown

b. Structured Light System (SLS) PointGrey Camera

General working of the SLS camera

The SLS camera system was custom built and it encases a PointGrey camera from FLIR Inc., and a projector that continuously produces a fringe pattern. The SLS camera captures six consecutive image frames and calculates depth information based on distortions in the reflections from the fringe patterns. These six images are called fringe images, and are processed/composited to create an RGB texture frame as well as corresponding x, y, z point cloud depth information.

RGB texture creation is relatively quick compared to depth calculations. Depth calculations in an SLS camera are expensive and time consuming, so enable them only when needed. RGB and depth calculations are self-contained and can be called separately. For example, `FringeToRGB()`, defined in `SLS_FringeToCoord.cpp/.h` returns an unsigned `char*` RGB image for a camera capture. `FringeToXyzm`(), also defined in `SLS_FringeToCoord.cpp/.h` returns x, y, z depth information.

Assuming the FlyCapture2 SDK corresponding to the PointGrey camera is installed and paths set correctly, the following files included in AREDA source distribution are required for depth and color calculations:

```
SLS_FringeToCoord.cpp/.h
SLS_ImageFilters.cpp/.h
SLS_InitFrame.cpp/.h
SLS_Phase2Coord.cpp/.h
SLS_PhaseMinUnwrap.cpp/.h
SLS_phasewrapunwrapAux.h
SLS_PointGreyCamera.cpp/.h
SLS_stdafx.cpp/.h
SLS_XyzmFileIO.cpp/.h
```

Some parts of the above files were altered from their original Purdue University versions because multi-threading was added to improve the efficiency.

SLS Camera Coordinate System

The SLS camera system is calibrated to measure distances in meters, whereas the Kinect camera measures distances in mm. Therefore, every SLS camera capture is converted to mm to ensure a consistent coordinate system and measurement. This is done in the function `frame_from_device()` within `slscamerainput.cpp`.

```
for (int i = 0; i < _kframe->width * _kframe->height; i++)
{
      _kframe->camsp[i].X /= 1000;
      _kframe->camsp[i].Y /= 1000;
      _kframe->camsp[i].Z /= -1000;
}
```

SLS Camera implementation

`slscamerainput.cpp/.h` is the starting point for implementing SLS PointGrey camera within AREDA. The header file contains function definitions and variable declarations that instantiate various PointGrey SDK functions and fringe image processing variables built by Purdue University. `slscamerainput.h` have definitions such as `device_ID()`, `connected()`, `getFrame()`, `start()`, `stop()`, `getHeight()`, `getWidth()`, etc. Generally, any new depth camera implementation will have similar definitions.

Certain variables such as `_projectorWidth`, `_projectorHeight`, `_imgWidth`, `_imgHeight` are preset and should not be changed. These variables are defined in `slscamerainput.h` and their values are set in `slscamerainput.cpp` constructor.

When AREDA requires a frame from the camera, `frame_from_device()` will check whether depth information is required. If required, it will call `FringeToXyzm()` and `FringeToRGB()` to calculate depth and color. Otherwise, it will only call `FringeToRGB()` that returns RGB texture.

KFrame Usage with SLS Camera

The class instantiation for KFrame occurs in many places across the AREDA codebase. This variable encapsulates color, depth, image height/width information.

Color information is stored as a character array (`BYTE *`) defined in Windows.

Depth information is stored as `CameraSpacePoint`, a datastructure containing X, Y, and Z values. As described in Kinect SDK 2.0 section in AREDA Software Dependencies, this is the only class definition that ties to Kinect SDK. If needed, `CameraSpacePoint` can be replaced by another data structure so Kinect SDK requirement can be avoided when using a non-Kinect depth camera.

`height` and `width` refer to the pixel height and width of either image capture or video frame capture from the depth camera.

Three Channel Color from SLS Camera

SLS camera SDK is designed to provide only a three-channel RGB image, as opposed to the Kinect which can provide RGBA image. As such, any frames captured should be of the form CV_8UC3 and not CV_8UC4. Also, the camera captures BGR image so an OpenCV function can be used to convert it into an RGB image. This is implemented in `cv::Mat slscamerainput::getFrame()` within slscamerainput.cpp

AREDA codebase uses RGBA color images. To maintain consistency when recording frames using the SLS camera, BGR frames are converted into RGBA with null values for the alpha channel. This can be seen in `recordFrame()` function like below:

```
cv::Mat temp(pFrame->height, pFrame->width, CV_8UC3, pFrame-
>colorFrame);
cv::Mat temp2;
cv::cvtColor(temp, temp2, cv::COLOR_BGR2RGBA);
memcpy(pFrame->colorFrame, temp2.ptr<uchar>(0), sizeof(unsigned char)*
pFrame->width * pFrame->height * 4);
```

Depth information is typically written to a file rather than processed via memory. `fileio` singletons are generally called to instantly write to files.

Calibration

The depth camera must be fixed and calibrated once to a specific work area. When calibrated, re-calibration need not be performed unless: a) Camera position/orientation changed, b) Work area borders changed, c) A different person performs the assemblies. Developer information for the following camera calibration steps is described below:

Background

The background calibration process is how you specify the 3d surface the expert will be working on. It has three steps, described below. See "`background.h/.cpp`" and "`calibrationcontroller.h/.cpp`" for implementation.

Capture: This simply grabs a still frame from the active camera and displays it, to be marked up in the next two steps.

Click 4 corners: Four user-selected points of an image are used to create a flat surface, denoting the plane of the work area.

Process: This uses the four points specified from the previous step to attempt to extrapolate a 2D plane representing the work surface. Using PCL's plane model (SACMODEL_PLANE) and the random sample consensus method (RANSAC), the depth information from the camera is used to calculate a planar model. The threshold value in the top-right corner is used as a tolerance for the plane height, in meters. For example, if the value is 0.01, then a variance of 1cm is allowed when calculating the plane from the camera's depth information. Any data outside of these threshold is discarded by AREDA. Background calibration threshold of 0.015 works for most cases.

Area

The area calibration process establishes a coordinate system for AREDA to use. There are two steps to this process, described below. This step requires the "Hiro" image target to be in view of the camera (**Figure 12**). See "`area_calibration.h/.cpp`" and "`calibrationcontroller.h/.cpp`" for implementation.

Figure 12. Hiro pattern image target

Record Area: This grabs a still frame from the active camera. Make sure the 'Hiro' image target is in unobstructed view of the camera, without a large angle between the camera and the image target

Analyze: Using ARToolkit, the captured frame is analyzed to find the location of the image target. The slider in the upper-right is used to set the threshold for binary image generation. This slider can be moved to adjust to the contrast of the camera. Note that the image target needs to be located within the work space calculated in the background calibration stage.

Skin

The skin calibration process enables AREDA to filter the user's hands during the recording process, and is done in the four steps below. See "`skincalibration.h/.cpp`", "`psogmm.h/.cpp`" and "`calibrationcontroller.h/.cpp`" for implementation.

Start Recording: This records a video from the active camera. While recording, the assembly expert should put their hand inside the work space for a few seconds, moving the hand at multiple angles.

View: This plays back the recorded video frame sequence from the previous step, allowing the operator to ensure the hand is within the work area.

Process: Using the captured video from the recording step, each frame is analyzed to detect skin pixels, and uses particle-swarm optimization for a Gaussian mixture model (PSOGMM) to train AREDA to detect and subtract the assembly expert's skin tones from each subsequent frame. The value in the top-right is used as the threshold for skin detection. High values above 0.900 often work well, but each camera/skin tone combination will have different values.

Check: To see if AREDA has been properly calibrated to filter out skin tones, this is used to play back the processed video to the operator. An ideal setup will remove all hand pixels (making them black) and leave the rest of the image unaffected. If more than just the hand pixels were removed, the threshold should be lowered, and the video should be reprocessed.

If many of the hand pixels were not removed, the threshold should be increased, and the video should be reprocessed. If no improvement is seen, the original recording may need to be redone. Skin detection threshold value of 0.995 works in most cases. A value higher than 1.0 for threshold can result in unpredictable behavior. The UI can be benefited by capping the upper limit for the threshold at 1.0.

Add parts to the library

This interface allows the operator to add all models relevant to the assembly. These models will be used for all future calculations in the processing phase and the refinement stage. See "partslibrary.h/.cpp" for implementation. Every CAD part uploaded to AREDA must have a corresponding point cloud file with the same file name in the directory where the CAD part is located. Acceptable file formats for CAD parts are .3ds and .obj. Acceptable file formats for point clouds is .pcd. These pcd files can be generated multiple ways:

a. PCL libraries: PCL is a pre-requisite for use with AREDA. When compiled/installed, a number of sample utility binary executables are made available. One of them is '`pcl_mesh_sampling_release.exe`'. When this executable is run at the command prompt as below, a corresponding point cloud will be created. `pcl_mesh_sampling_release.exe Base.obj Base.pcd`.
b. Meshlab: This is an open source software that allows creating and manipulating point clouds. This program can be used to import CAD geometry (e.g., .obj) and convert them into corresponding point clouds.

Record assembly

This interface records a sequence of depth images that makes up the assembly process. The spacebar must be pressed after each assembly step is completed, except on the final step. Be aware that any depth information outside of the defined work area will be discarded by AREDA's parts matching algorithm.

While putting together the assembly, make sure the base of the object does not move. The algorithm works by detecting new depth information in the scene. If the assembly moves, the algorithm will attempt to match a larger point cloud than was added.

While assembling, try to keep your hands opposite of the depth camera. While AREDA does ignore the parts of the image with skin tones, this means that there will be less information available to AREDA during the parts matching process, leading to a less accurate matching process.

Process assembly

The assembly is processed in three sections: 'Steps', 'Parts', and 'Animations'. Assembly processing is computationally challenging, so there is opportunity for it to fail if some of the parameters in the calibration stage were not set properly. Processing begins in the integrated process widget in `integrate_process.cpp/.h`.

Detect number of steps

In the steps section of assembly processing, the recorded video is processed for the number of steps. If the number of steps was manually specified during the assembly (by pressing the space bar at the end of each step), it will be quickly processed. Otherwise, the video frames are analyzed for the presence of a hand entering and exiting the work surface area. If the skin threshold was not set and processed properly, this step will fail. This section is implemented in the following files:

```
process_parts.cpp/.h
background.cpp/.h
skincalibration.cpp/.h
psogmm.cpp/.h
```

Identify parts

The parts section of assembly processing attempts to determine the part captured in each assembly step. To do so, the background, work surface, and skin are segmented out from each frame. The largest cluster in the resultant point cloud is determined to represent points on the surface of the assembly part. This cluster of points is then matched to each point in the parts library using the Iterative Closest Point (ICP) algorithm. The ICP match is performed ten times on each part in the library for robustness, and the part in the library with the lowest ICP error is chosen as the matching part.

This section is implemented in the following files:

```
process_parts.cpp/.h
partslibrary.cpp/.h
```

Animations

At the time of this writing, animations are disabled because they are incompatible with manual specification of steps (e.g. by pressing the spacebar while recording). When enabled, the animation processing seeks to determine which steps are *move steps* and which steps are *static steps*. Move steps indicates steps when the part is moving in the scene. Similar to part matching, each frame of the move steps are segmented, and the largest cluster is chosen as the part in the assembly step. The animation for each step is recorded as the trajectory of the largest cluster during its associated move steps.

This section is implemented in the following files:

```
process_parts.cpp/.h
animation.cpp/.h
```

Troubleshooting

This section documents a couple useful tips if the processing step does not complete as intended.

Processing

To view the current status of the assembly processing, click the *Settings* -> *Verbose Output* menu option. During processing, you will see the following window (**Figure 13**).

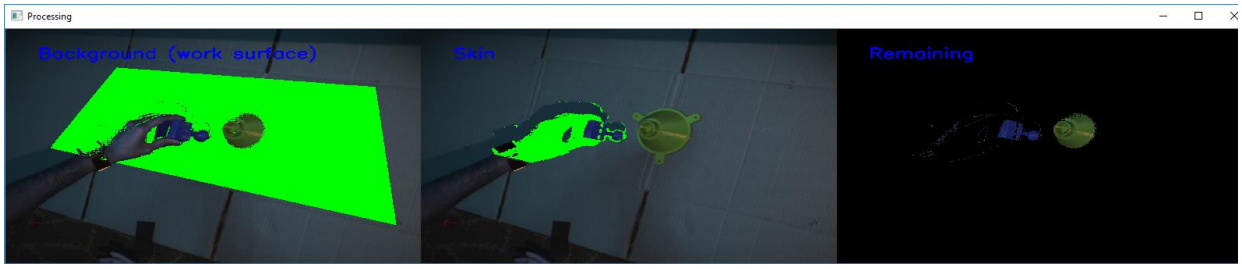


Figure 13. Processing stage verbose output

On the left side of the window shows the background segmentation, which removes all 3D points except for those in the work surface and above the work surface (e.g. a hand or assembly part). Each step in the assembly processing segments the background. If the background threshold was set correctly, you will

see a green quadrilateral specifying points on the work surface. Points not in the work area are removed. Similarly, points considered part of the work surface are removed.

In the middle of the window shows the results of the skin segmentation. If the skin threshold has been appropriately set, points in the work area detected as skin will be green.

In the far right of the window shows the resultant points after segmenting out points not in the work space, points in the work surface, and points detected as skin. The result should be the assembly parts.

Parts Matching

Parts matching process is time consuming and may appear that the code might have halted/stuck. There are a few utility functions included in `aredautilities.h` to help visualize what actually is happening under the hood. The displayPointClouds function in particular helps visualize the alignment of each part in the part library with the points left after segmentation in each assembly step. The visualizer is only displayed when '`verbose output`' mode is activated from the menu. The location to insert this call to display is in `partslibrary.cpp`, in the function:

`Nodeplusplus* parts_library::get_match(pcl::PointCloud<pcl::PointXYZRGBA>::Ptr _cloudIn)`, after ICP is run 10 times.

Refinement

The refinement window is shown below in **Figure 14**. This window shows the results of processing, and allows you to update automatically detected steps and parts. Refinement requires the Hiro fiducial marker from the AR Toolkit to display the model. Binary thresholding for the marker can be adjusted at the top right, and if the marker is actively detected, you will see text updates in the console window. At this time, refinement is only partially functional. Scale, rotation, translation, and geometry swapping is implemented. This section is implemented in the following files:

```
refinement.cpp/.h
animationcontroller.cpp/.h
artoolkitosg.cpp/.h
contentdata.h
contentmanager.cpp/.h
Modelmanager.cpp/.h
icontent.h
layermanager.cpp/.h
scenecontroller.cpp/.h
scene.cpp/.h
```

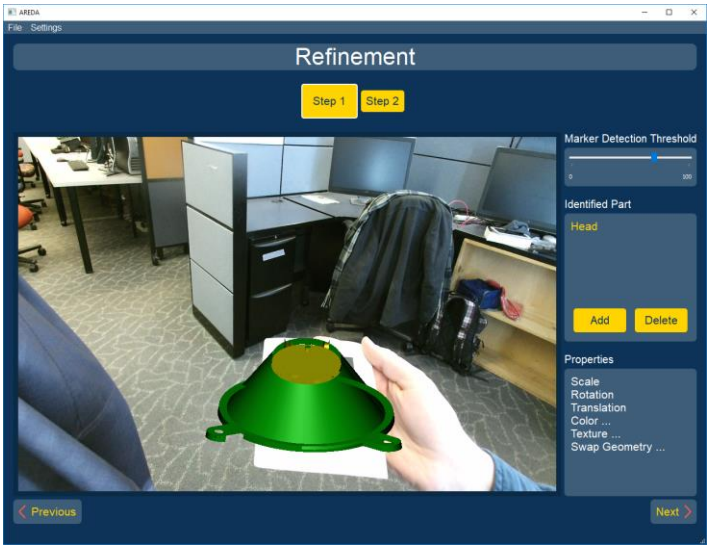


Figure 14. Refinement Section

Export

To export the models and animations as they appear in the refinement stage, go to the file menu, select "Export…", and choose which directory to export to. This will produce three files for each step of the assembly:

The model#.fbx file: This file is the binary FBX model of the equivalent part added previously to the parts library. If OSG is not configured with FBX (2017.1) build support, no model files will be written.

The animation#.osg file: This is an ASCII file representing a sequence of animation frames detected by AREDA.

The animation#_offset.osg file: This is an ASCII file representing the modifications made in the refinement stage to the displayed model.

To create your own visualization program, simply load the model#.fbx files, apply the position, attitude (rotation), and scale from each corresponding animation#_offset, and then continually loop through the list of positions, attitudes, and transforms located under the 'Controlpoints' subsection of the corresponding Animation#.osg file. Each control point has 11 values per captured animation frame, representing:

[keyframe] [X, Y, Z position] [X, Y, Z, W quaternion] [X, Y, Z scale]