AIMS™ Computational Challenges - Computational Challenges to using Gaming technology for Accurate Measurements of Clinical Procedures

Benjamin Cawrse, Eric Wilkinson

SimIS, Inc.

Portsmouth, VA.

Benjamin.Cawrse@simisinc.com, Eric.Wilkinson@simisinc.com

ABSTRACT

The medical profession often uses sophisticated and costly training equipment specifically designed for the medical industry. This works well for organizations with large training budgets but others are not able to afford those capabilities.

This paper identifies opportunities to use commercial-off-the-shelf gaming technology in AIMS™ software to provide hands-on training, capture movements and provide feedback to users. This technology provides cost savings and is commonly available; however, there are computational challenges that must be overcome to provide validation for medical procedures. These challenges include developing validation tools for performance assessment and extending existing interfaces for better object and skeletal tracking.

Using three common medical procedures (CPR, Intubation, and Lateral Transfer) this paper will discuss the challenges defined above and how the commercially available Microsoft Kinect™ is utilized to effectively train users. The results show the feasibility of high-tech low-cost solutions applicable for use in the medical community.

ABOUT THE AUTHORS

Benjamin Cawrse is a senior software engineer for SimIS who obtained his MS in computer science and minor's in modeling and simulation and computer engineering from Old Dominion University.

Eric Wilkinson is a software engineer for SimIS, Inc. with an Applied Science degree in Modeling and Simulation who also brings his hobbyist experience as a MMORPG developer to the table. Eric has been working on AIMS since June 2012.

AIMS™ Computational Challenges - Computational Challenges to using Gaming technology for Accurate Measurements of Clinical Procedures

Benjamin Cawrse, Eric Wilkinson

SimIS, Inc.

Portsmouth, VA.

Benjamin.Cawrse@simisinc.com, Eric.Wilkinson@simisinc.com

INTRODUCTION

Often, medical training is done with expensive equipment designed to simulate real situations. While this equipment has progressed in its sophistication and effectiveness, it is not easily available to trainees practicing outside of simulation labs. Besides limited access, if these devices are intelligent enough to report feedback, they generally focus on changes to the device instead of what the user is doing, e.g. a CPR manikin which can sense a change in depth, reports the quality of that depth irrespective of whether or not it is done with a trainee's foot or locked hands.

Using commercial-off-the-shelf gaming technology, we have developed AIMS - the Automated Intelligent Mentoring System. This is an inexpensive alternative which provides hands free training as well as real-time feedback to the user based on what they are doing. Using a depth camera, the Microsoft Kinect, we are able to capture the user's movements and body mechanics. For instance, AIMS knows which hand was used to pick up a laryngoscope while performing intubation, or whether or not the user's shoulders are correctly over the manikin while performing compressions during CPR.

Creating an effective training tool with gaming technology has its challenges. Although gaming technologies are designed for user interaction and virtual immersion at an unrestrictive cost, they have their boundaries; which we have undoubtedly pushed. To achieve a satisfactory level of accuracy and the ability to identify objects within the depth camera's scene, we had to extend the available features of the Microsoft Kinect, integrate computer vision techniques, and develop validation and assessment tools. In order to illustrate our efforts, this paper describes three medical procedures which we have included in AIMS: CPR, Intubation, and Lateral Transfer.

In order to use the Microsoft Kinect, you must have a way to communicate with it through software. The two main solutions are to either use Microsoft's software development kit (SDK) specifically designed for the Kinect, or the open source software OpenNI®. Although OpenNI has been around longer and supports depth cameras other than the Kinect, we chose to go with the SDK provided by Microsoft. Our reasons for choosing the Microsoft's SDK over OpenNI were that the Kinect SDK was guaranteed to have consistent support from Microsoft, it was Kinect specific, and it supported the C# programming language natively. Since OpenNI is open source, the development is community driven. In many cases, community driven tools can be quite successful, but we did not want to rely on the community for improvements to our key system in AIMS. Microsoft already invested a considerable effort to create the Kinect for their Xbox 360 gaming system. Once they released a Kinect for Windows version, we were confident it would exhibit rapid growth. Since we already decided to use the Kinect for our depth camera, choosing a Kinect specific source with native C# support was also a perk. A Kinect specific solution removes any requirements to specify camera parameters, everything is already designed for our device, and our developers are most familiar with

C# as a programming language, so there was less of a learning curve.

TRACKING SPECIFIC USERS

One issue which arises when using the Kinect is identifying who to watch. By default the Kinect tracks the closest user, but it also supports more tracking schemes; none of which are robust enough for our requirements. Default tracking of users is not adequate because we need to function well in training or learning environments which are more serious and less forgiving than a recreational environment. When using depth cameras for gaming, the closest, most active user is likely the one who should be watched. This is not always the case in a training environment. The first Kinect for Windows camera can identify six users by position, but only two of those users may be fully tracked. Fully tracked refers to the Kinect returning bone and joint information of the users as well as positions shown in Figure 1. Multiple people may exist within the camera's view while training with AIMS, and we need to switch watched users as well as monitor lost users. If people other than the trainee pass by or walk near the camera, AIMS should still function as designed. A second issue is analyzing watched users. Receiving user movement data is not helpful if the users actions are still obscure, and the Kinect SDK does not provide much to understand what a user is doing from their raw data.

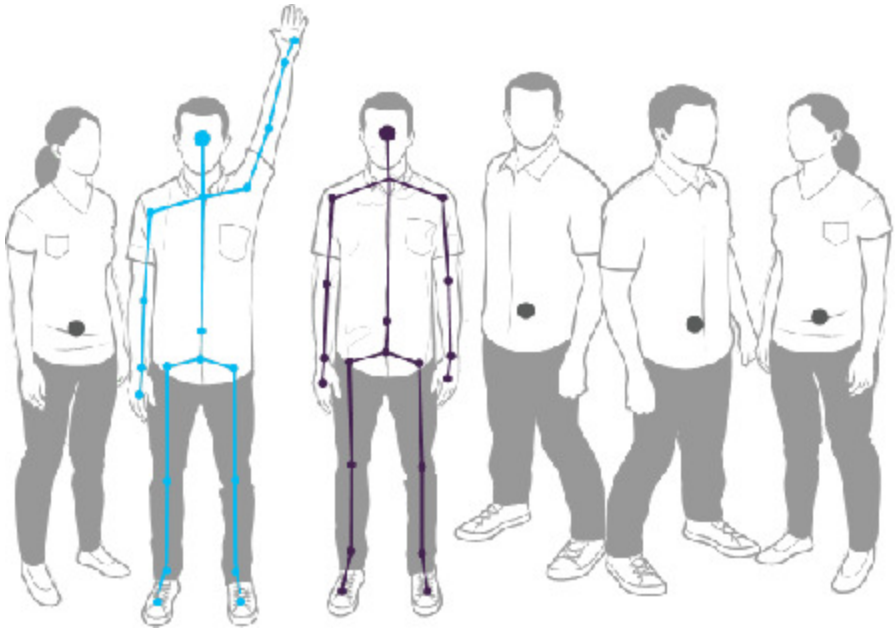

Figure 1. Kinect Can Recognize Six People and Track Two. Reprinted from Microsoft Developer Network, Retrieved from http://msdn.microsoft.com/en-us/library/hh973074.aspx. Copyright 2014 by Microsoft.

Identifying Who to Watch

We solved the problem of tracking specific users with stances that are essentially defined by an algorithm which compares joint positions. Before the training begins, the user is asked to mimic a given stance, one hand raised above their head for example, and with that we can identify the user exhibiting this stance as one the system would like to watch. Identifying users using stances with the Kinect for Windows can be difficult because the Kinect is only able to track joints and bones for two people at a time and only positions of six people at a time. Bone and joint

information is necessary to identify the stance that a person in the scene is exhibiting, or if they are in a stance at all. In order to check stances of more than two users within the camera's scene, we developed a technique to choose the fully tracked users on each camera frame. This means on each frame we can check the stance of at most two users, and on the next frame we can check the stance of up to two other users. On each frame, all users that have not had their stance checked are put on a queue, and each user which has had their stance checked, but does not exhibit the correct stance, is put at the end of the queue. Only users who do not already exist in the queue are placed in it, which is used to decide which users should be fully tracked next. At the end of every frame, when searching for a stance, the next users to be fully tracked are taken from the queue. This system ensures that each user in the scene, up to six, can have their stance checked until a user is found with the correct stance. After a user is found performing the correct stance, unless more than one fully tracked user is required, the system only tracks that specific user.

Switching Watched Users

Identifying specific users to track based on a stance worked well, unless another user exhibited the correct stance unintentionally. We needed to build a mechanism to switch which user would be watched in case the wrong one was being watched. When a single user is being watched, the Kinect is able to fully track one more user within the scene. Just as we identified the current user being watched, we compare the other users in the scene for the correct stance while we already have, what we think, is the correct user being watched. Since the Kinect can only fully track two users at once, we can only check the stance of one user per frame. If a user, other than the currently watched user, is performing the stance, we quit iterating through users and keep watching this user each frame. If the user holds that stance for a specified amount of frames, then that user becomes the watched user. To ensure this is desired functionality and it does not inhibit AIMS during training, we only use this technique in the initial setup portion of training when we are identifying the variables in the scene.

Watching Lost Users

Besides identifying specific users to watch, we needed to develop a means to keep watching users which were lost by the camera. A user may become lost if they are occluded by another object, if the camera fails to recognize them as a human, or if they leave the camera's field of view. We created a solution for the first two cases, the third case can be resolved by re-finding the user with a stance. In order to re-find users that are lost, AIMS keeps track of the last position for every fully tracked user. Also, each fully tracked user has a time to live. Once a user is initially lost, all of the users in the scene, including position only, are compared against the last known position of the lost user. This comparison occurs for every frame returned by the camera. If the user is not found within the collection of seen users, then the lost user's time to live is decremented by one. Once the time to live reaches zero, AIMS begins searching for stances to replace the lost user.

Analyzing Watched Users

After identifying which user to watch, we developed techniques to better understand what the user was doing. The Kinect SDK gives joint positions of users, among other things, for every frame delivered by the camera. Single instances of user positions were not enough, so we built a tool to collect that information into a history of joint positions over time. The length of the history is limited for performance reasons, but having a history allows us to transform single instances of joint positions into identifiable actions. This is further extended by a trending functionality that we built into the history mechanism.

The trending functionality can identify a positive, neutral, or negative trend in each of the three dimensions of a joint's position across any range of that joint's collected history. This gives us easy ways of identifying how joints move over time. Firstly, if a joint changes direction, the trend in one dimension of that joint changes from positive to

negative, or negative to positive. Secondly, if a user's joint stops moving, the trend changes from positive or negative to neutral. Thirdly, to identify more complex changes in joint movement, a sequence of trends may be used. This is very useful, for instance, if we want to track when the user has remained relatively still, we wait for a neutral trend over a large range of history. This does not force the user to remain absolutely still, but notices that they have not moved significantly. Another common use for trending is to identify change in movement. Identifying when a change occurs gives us an understanding of what the user is doing and allows us to further analyze data around that action.

While AIMS is well equipped with assessing one user, determining how to compare the movement of two users is another challenge. As we collect the positions of joints and objects, we are able to map these points out over time, creating its path. Comparing two paths, we are able to determine whether or not the two are moving in a similar, synchronous manor. In order to do this successfully, we created a development tool which records and renders paths, and then compares them and produces the percent of synchronization. Using this tool, we developed a Lateral Transfer module which assesses two users transferring a medical patient from one bed to another.

IDENTIFYING OBJECTS WITHIN THE SCENE

For AIMS to identify objects within the scene during training, we use computer vision. Specifically, we use color tracking capabilities to locate colors within the video stream given by the depth camera. This is not necessarily the easiest approach, and comes with its own issues and triumphs; problems which arose because we used color tracking and how we overcame them.

Computer Vision

AIMS identifies objects, such as manikins and medical tools, by utilizing computer vision techniques. This is very important, so that we can know what the user is interacting with, and where they are relative to the items within the scene. Computer vision, roughly, deals with analyzing digital images for identifiable traits; recognizing actions or items within an image. We identify colors within the scene which are simply placed on the objects we are tracking. We use the open source computer vision library, OpenCV©, for this functionality because of its reputation for being stable, it covers most areas of computer vision, and it's free with a flexible license.

When we identify objects in the scene by color we get more than two-dimensional pixel locations. The Kinect can map color locations to depth locations allowing us to get three-dimensional real-world locations of objects, relative to the camera system, using a two-dimensional pixel location within the frame. Essentially, when we locate one of our colors on an object, we know where that object is in the same coordinate space as the user.

Color Tracking Issues and Triumphs

Our ability to identify colors lets us relate the user to the rest of the objects within the scene. We have put a substantial effort developing supporting tools for enhancing this. The added support is necessary because of the issues that arise when relying on color tracking across different environments. We had to develop a means to collect and store color values of objects. In its infancy, this was a grueling process which was required with major environment changes. Now, this has been simplified and can be performed within minutes. The largest reason different environments do not work well with color is lighting. Different lighting conditions can substantially alter the color values returned by the camera. As this became more apparent, we improved our collection and storage tools to compensate, as well as create a tool to analyze known colors.

To make working with colors easier, we developed a color calibration tool. This tool identifies a color range from user mouse clicks by grabbing the Hue, Saturation, and Value (HSV) values of the color in the camera's video where the user clicked. As more values are collected with clicks, the range expands and changes to include these values. To help with the lighting issue, a brightness slider exists on the tool to change the camera's brightness value of the video stream. Giving access to the brightness allows the user to get HSV values which include different lighting conditions; making the generated color range more robust and compatible with different environments.

AIMS relies on multiple color definitions, per color, in order to manage the issues of identifying colors across varying environments. Because of this, the color calibration tool allows storing colors. Every stored color is given a text descriptor, which should be the color it is meant to represent: red, blue, green, etc. It is important to be able to set colors to objects as well, otherwise identifying colors is useless. Colors are set to objects easily by selecting the object in a dropdown, and then clicking a 'Set Object Color' button. Once the button is pressed, the current color definition within the tool is set to that object. This association allows AIMS to look for the object during training with a specified color definition.

To ensure a better user experience, and require less interaction with the color calibration tool, we developed an auto color calibration tool which works behind the scenes during training. The auto color calibration tool attempts to find a suitable color within the environment if the currently set color, for the object it is looking for, is not being found. Because each color has a descriptor, all of the stored color ranges with the same descriptor as the one set on the object that is being looked for may be checked on each frame from the video stream. In short, all of the color ranges which represent the color being looked for are checked. In order to check a set of color ranges for compatibility in an environment, the auto color calibration tool runs them across multiple frames and retains information on how they performed. After sufficient information is collected, it is analyzed to identify color ranges with sporadic results vs those expected from a good color range. If a color range exists, after the analysis phase, which exhibits good results, it is then set to the object being searched for. If no color exists with good results after the analysis phase, then manual color calibration must be performed. Although manual color calibration is necessary, as it is performed the collection of compatible color ranges expands and the system becomes better able to deal with varying environments. This means, whenever colors are manually calibrated and added, conditions causing AIMS difficulty identifying objects are mitigated; variances in lighting or other varying issues. Over time, color recognition becomes more robust.

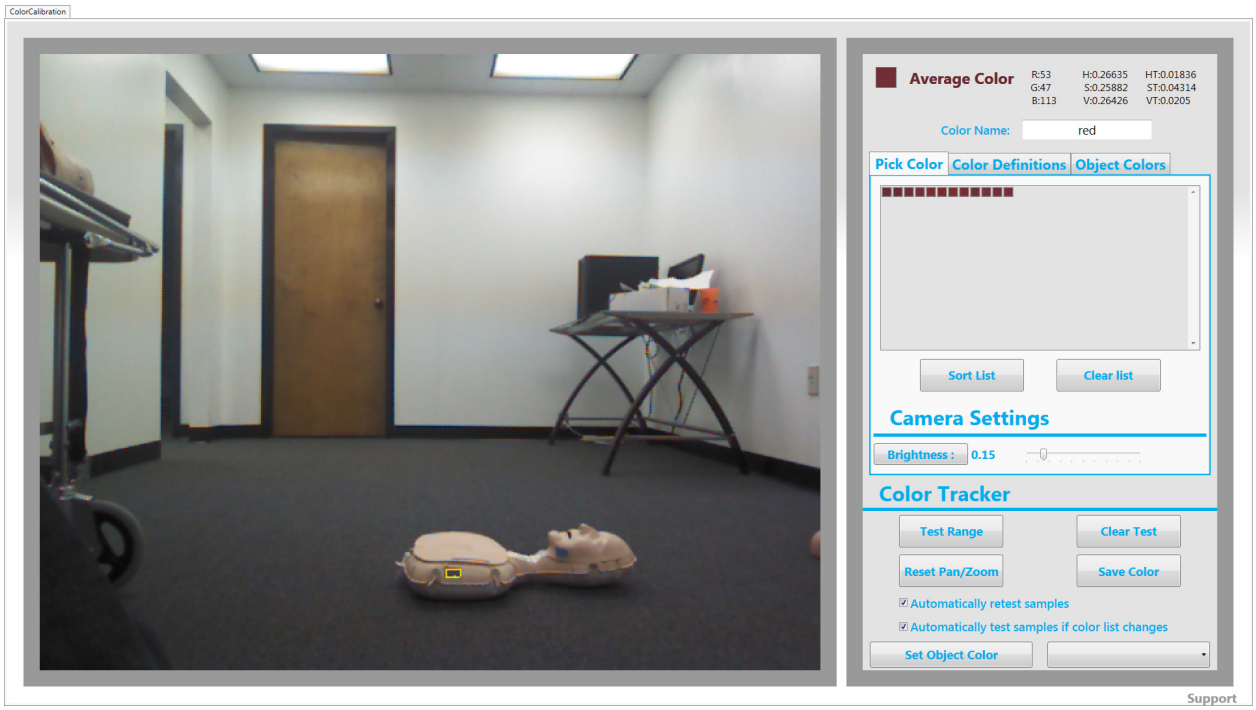

Figure 2. AIMS Color Calibration Tool

BRINGING IT TOGETHER

In order to model a procedure in AIMS, we must break it down into it's basics, and reconstruct it into our system's architecture. At the height of AIMS we have the lesson which consists of multiple stages which analyze various actions made by the user as they progress through the procedure. At the end, AIMS assesses their completion and, based upon the user's performance, calculates their result - providing relevant feedback tailored to their particular actions.

Lessons

Any procedure which can be seen from one point of view, within a range of 1.2-3.5 meters, and to which a checklist can be applied, can be mapped out into AIMS and molded into a lesson. The lesson, in essence, is a simulated procedure which is evaluated for performance accuracy. Through our development process, we equip an empty lesson with the procedural flow through which a user must progress to perform the procedure. This procedural flow is derived out of field research and standards in the community. In the example of our Compression-Only Adult CPR lesson, we follow the latest American Heart Association® (AHA) guidelines and received input from local medical schools, emergency response organizations, and CPR trainers. From here we take a look at the steps of the procedure, break down the task into organized patterns of user actions, explore the nuances and variances in technique of accomplishing each step, identify the measurable and unmeasurable aspects of the procedure, and finally apply our developed arsenal of measuring techniques.

Creating he model of mastery or a esson s a highly coordinated ask of bringing ogether subject matter experts, and ecording hem using a Kinect) unning hrough he procedure as t should be done. The advantage of his s that AIMS doesn't ust rain o one master's model, t also rains o he models of he procedure's extbook definition combined with the master techniques of multiple subject matter experts. Each procedure requires a new algorithm.

During lesson development, the procedure is broken down into a collection of basic actions. If the procedure has multiple accepted techniques, each must also be considered. Once the steps of the procedure are defined, we assess each one and determine its complexity, trackability, influence, and importance. Occasionally, we run into a step of a lesson which our current measuring techniques are simply insufficient to track its performance. AIMS grows whenever we run into this issue, as this is how our arsenal expands. Whether it be measuring the path of a tool, the synchronization of movement between two users, or assessing compressions during CPR, discovering a new tracking technique unlocks new possibilities for additional procedures.

As the lesson runs, the user's actions are compared against the expected actions of the mastery model. The user is assessed regarding whether the actions are performed properly, improperly, or at all. For example, if a tool was meant to be picked up with the left hand, and it was picked up with the right, then AIMS will consider the implication of this and react accordingly, e.g., by deducting points and providing critical feedback.

Stages

Attempting to tackle the procedure as a whole is a difficult task. It is only once we break down the lesson into the various actions which combined make up the process that we are able to approach. Every step in the procedure is reflected as a separate stage in the lesson. For example, if the user must pick up an item and then set it back down, then these two actions would each be a separate stage. This not only effectively breaks down the process, but it also allows us (in our User Interface) to inform the user of the next step in the procedure with a quick glance.

A stage is responsible for assessing the performance of the user as its actions are completed. Once the actions of the stage are completed, the lesson is informed and moves along into the next stage. Occasionally multiple stages are run concurrently, depending upon the logical flow of the steps throughout the procedure. Additionally, stages are setup to measure not only the proper actions, but also the improper actions. For instance, in CPR, it is proper to perform compressions with your arms locked at your elbows (you should use the weight of your body), and improper to perform compressions bending at your elbows. In order to build a robust training solution, we must detect and assess each measure.

Watchers

Just as stages reflect an action, each action is broken down into the requirements for fulfilling that action. The creation of watchers arose out of a requirement to search not only for the existence of proper actions, but also that of improper actions. For example, if the action is to pick up a tool, then the requirements for fulfilling that action are that the tool is found, the tool is moving, and that the user is holding the tool. We can be even more specific with our action by ensuring that the user is using the correct hand versus the improper hand to use the tool. In either case, the stage is comprised of watchers, each of which is observing the requirements which fulfill an action.

Watchers have two states: on and off. As the requirement is observed, the watcher trigger states turns to on. It is up to the stage to manage and interpret the states of its collection of watchers in order to progress its own state. Watchers will often be coupled together and rely on other watchers to make its observations. Additionally, for every action that can be performed improperly, additional watchers are added to detect them. This chain allows us to have a complex cluster of watchers flow together in order to accurately assess an action being performed.

Zones

The goal of a zone is to identify whether or not an object (or skeletal joint) is within a key sector of our scene, and to

indicate for how long said object has occupied the space. As the Kinect skeletons move their joints throughout the scene, and as the Color Tracker locates and tracks pertinent objects within the scene, their three dimensional positions are calculated and compared against the calibration point, on a per frame basis. A zone can be thought of as an area of importance in the scene. The location of a zone is often positioned relative to the calibration point of the lesson in which they are being used (static zones), but there also exists zones whose location is constantly being re-derived in relation to other significant points in three dimensional space.

Calibration

The most essential step in any procedure in AIMS is its calibration. As it turns out, this can also be the most challenging. During the calibration phase, and as a prerequisite to beginning a procedure, the lesson becomes aware of the pertinent people and objects in the scene. This special process sets up the essential links between the real world and AIMS which enables the system to accurately track the actions of the user. It is during calibration that the system identifies which colors and objects to track, the position of the lesson's "zero point", and the role of users. Each calibration is uniquely specific to a lesson. For example, in CPR we identify the user as well as the location of the manikin's side, while in Lateral Transfer we not only identify the manikin's position and orientation but also the primary and secondary user. Successfully calibrating the accurate "zero point" is imperative to the remainder of the lesson.

Result Calculation

The reason for all of this is to produce a result which accurately depicts the performance of the user. Translating the users performance into a result is a challenge. Calculating the result means to go back through and assign a score and weight value to each aspect of the lesson. Determining this balance effectively takes us back into our research on the nuances of the procedure, as well as the valid input of subject matter experts. The system analyzes the detected actions and compares them to the expected path of mastery. The further the user strays from the path of mastery, the worse off the users score. AIMS will tell the user whether they are either successful or unsuccessful in their attempts. In addition to this, the system must also inform the user of why they've been given the grade they are given. AIMS provides the user their result, gives critical feedback, and provides tips for improving their performance - all based upon the difference between their performance and the mastery model.

Comparisons and Overall Capabilities

AIMS strives to build high-tech low-cost solutions for medical training and assessment. The system provides the users a hands-on training experience with real-time critical objective feedback, using low-cost gaming technology. Given that AIMS requires no physical sensors within the training devices, and is entirely based off of the software linked to the Kinect camera, sensorless manikins have once again been made ideal for use in the trending medical training market. Furthermore, AIMS provides a feature that's lacking in even the most expensive medical training manikins (which cost tens of thousands of dollars): the ability to measure the human.

CONCLUSION

AIMS was created with commercial off the shelf gaming technology to increase the availability of objective health care training. Although gaming technology is designed for user interaction and virtual immersion, it presents challenges when adapting it to other purposes. For us designing AIMS, the main challenges presented were identifying and analyzing users training with the system, tracking objects in the camera's scene that the user interacts with, and bringing all of the tools together into an interactive training environment with objective real-time

feedback. Microsoft relieves some of the work associated with identifying users in the next generation Kinect by allowing all users within the scene to be fully tracked instead of only two out of the six, but it does not mitigate the issue entirely. We have successfully overcome the challenges placed in front of us designing AIMS which has allowed us to create a high-tech low-cost solution that replaces subjective matter evaluation with objective metrics based on experts consensus.

ACKNOWLEDGEMENTS

The authors would like to specially thank the simulation department of Eastern Virginia Medical School for serving as our main source of medical guidance and expertise - and for allowing us to use them as a testbed throughout development. We also would like to thank the AIMS team including: Justin Maestri and Beth Johnson for helping to provide vision, guidance, and stability to the project; and Thomas Langhorne and Tien Nhan for their development efforts. Finally, a thank you to Dr. Johnny Garcia, President and CEO of SimIS, Inc. and owner of HealthCare Simulations, LLC., and our Chief Scientist, Dr. Tolk, for making it all possible.

REFERENCES

American Heart Association (2011, June). CPR & Sudden Cardiac Arrest (SCA) Fact Sheet. Retrieved from http://www.heart.org/HEARTORG/CPRAndECC/WhatisCPR/CPRFactsandStats/CPR-Statistics_UCM_307542_Article.jsp

American Heart Association (n.d.). *AHA Hands Only CPR*. Retrieved from http://www.heart.org/HEARTORG/CPRAndECC/HandsOnlyCPR/LearnMore/Learn-More_UCM_440810_FAQ.jsp

Microsoft Developer Network (n.d.). *Tracking Users with Kinect Skeletal Tracking*. Retrieved from http://msdn.microsoft.com/en-us/library/jj131025.aspx

Microsoft Developer Network (n.d.). *Skeletal Joint Smoothing White Paper*. Retrieved from http://msdn.microsoft.com/en-us/library/jj131429.aspx

Microsoft (n.d.). Latest Kinect for Windows SDK(v1.8) [Computer Software] Retrieved from http://www.microsoft.com/en-us/kinectforwindowsdev/Downloads.aspx

OpenCV (n.d.). *OpenCV*. Retrieved from http://opencv.org

OpenCV (2013). OpenCV For Windows (2.4.8) [Computer Software]. Retrieved from http://opencv.org/downloads.html

WorkWithColor (n.d.). *Color Theory - Introduction*. Retrieved from http://www.workwithcolor.com/color-theory-introduction-4619.htm

Tolk, A., Miller, G. T., Cross, A. E., Maestri, J., & Cawrse, B. P. (2013). AIMS: Applying Game Technology To Advance Medical Education. *Computing in Science and Engineering*, 15(6), 82-91.