

A Picture-in-Picture Interface for a Multiple Robot System

Andrew M. Pitman, Curtis M. Humphrey, Julie A. Adams
Department of Electrical Engineering and Computer Science
Vanderbilt University
Nashville, Tennessee USA 37235

An exciting possibility is that robot teams may be deployed in the future to assist in Chemical, Biological, Nuclear, Radiological, and Explosive (CBNRE) situations. As such, the human-robotic interface that people use to control these robot teams must be near-ideal. The interface must provide for what Drury et al. call situational awareness (Drury et al., 2003). Previous studies have shown the importance of situational awareness when one person controls multiple robots (Humphrey et al., 2007). This paper describes the development of an interface that used picture-in-picture functionality that may improve the situational awareness when controlling multiple robots. The new interface utilized the University of Pittsburgh and Carnegie Mellon University's Urban Search And Rescue Simulation (USARSim) and its MultiView object (Wang et al., 2005) to add a bird's-eye view of the simulation in the grey halo area from the previous interface (Humphrey et al., 2007).

INTRODUCTION

Controlling multiple robots is a difficult task when assigned to one person. Often, a ratio of two or more operator(s) to one robot is used to overcome the shortcomings that a higher ratio presents. One of the limiting factors of any multiple remote robot system is the lack of situational awareness (Yanco and Drury, 2004). For the robots to operate as an effective team, the operator must have at least a general idea about where they are in relation to each other (Humphrey et al., 2006), in addition to knowing information about the terrain surrounding the robots in all directions.

A basic form of gathering information from a robot is from a camera mounted somewhere on the robot. Previous projects have suggested overlaying information onto the video feed coming from the robot's camera (Calhoun et al., 2005). For example, if an unmanned aerial vehicle (UAV) is planning to bomb a specific target in a crowded, urban environment, then the operator might use an overlay on the video feed from the UAV's camera to find the target's location. Another way of getting information from cameras mounted on robots is to use a picture-in-picture interface (Calhoun et al., 2006). The picture-in-picture interface may use a robot's unaltered camera feed as the interior picture, and the exterior picture can provide a different view from the robot's camera

view. This interface could provide more visual information to the operator.

Humphrey et al.'s (Humphrey et al., 2007) interface for controlling a multiple robot system provided some situational awareness, but because each robot had only one camera placed just a few feet above the ground, there is room for improvement. This system provides a grey halo area surrounding the video feed from the robot's camera that was somewhat empty. Situational awareness is difficult to obtain when teleoperating a robot with limited cameras.

When robots are used in CBNRE situations, situational awareness can play a key factor in their effectiveness in identifying victims in need of help and also in identifying potential obstacles and threats. When only using the single camera mounted on the ground robots, a single operator may have great difficulty while operating teams of multiple robots when the objective is to eliminate a threat in a CBNRE situation.

One of this project's objectives is to create a picture-in-picture interface where the interior image is the video feed from a camera mounted on a ground robot and the exterior image is the video feed from a camera mounted on a flying robot pointed at the selected ground robot. The flying robot is intended to keep the selected ground robot in the center of its camera's view.

An overhead, top-down view can provide situational awareness by giving the operator visual information about the objects in the robot's

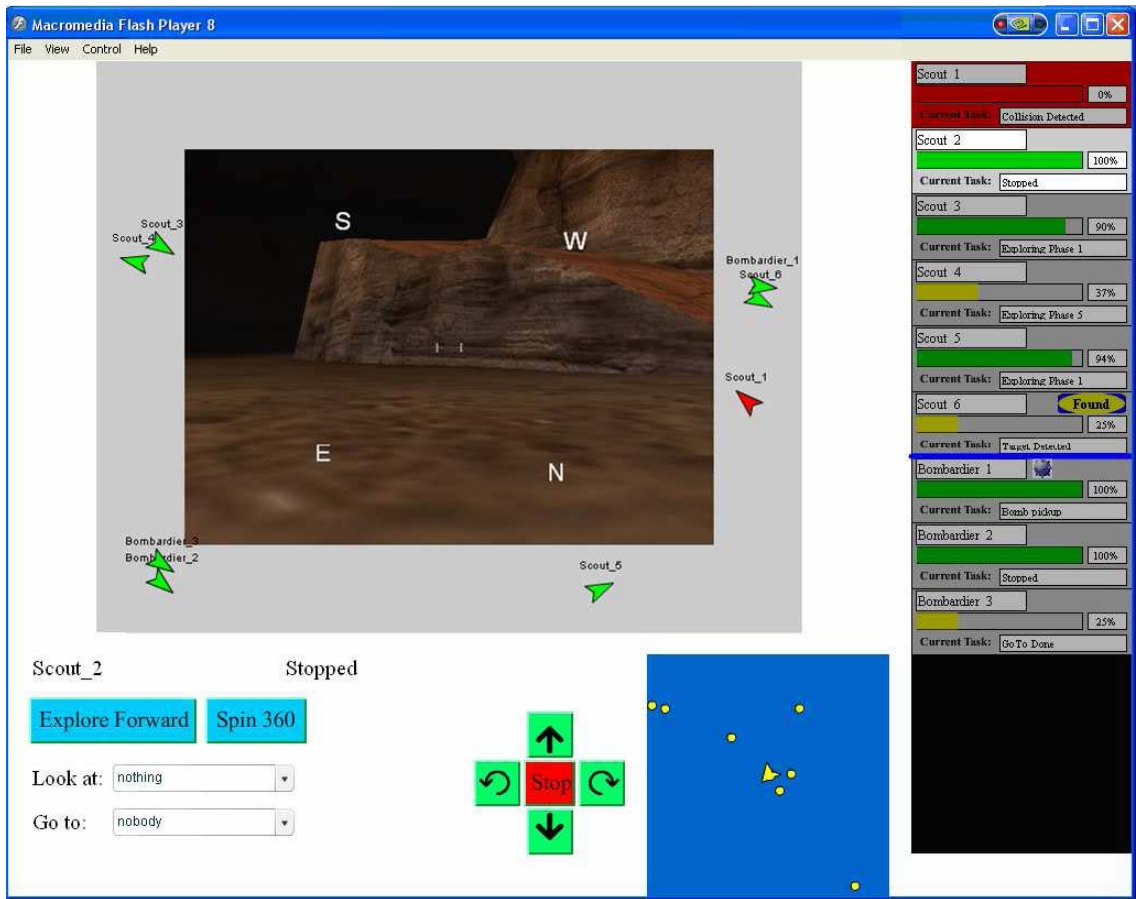


Figure 1: The original Flash® interface

proximity that would otherwise take several seconds for the robot's camera to view, as it rotates 360 degrees. Merely giving an operator the ability to switch between a bird's-eye view – from a camera mounted on an unmanned aerial vehicle – and a camera mounted on a ground robot may help, and it serves as a good starting point, but it also leaves room for improvement. One improvement comes in the form of a picture-in-picture format, allowing the operator to see images from both the ground robot's camera and from the overhead camera simultaneously. The objective of such a display is to provide a broader perspective along with a focused view.

Clearly, with the possibility of using these robots in disaster situations, situational awareness will be an important factor in their effectiveness. Using a combined overhead view and a ground view may significantly improve situational awareness. A picture-in-picture format may be more natural and intuitive than a split-screen format when one view is from a forward-facing

camera on the ground robot and the other view is a bird's-eye view. A picture-in-picture interface that combines a top-down view with a ground-level view may permit the operator to gain a higher level of situational awareness.

SYSTEM DESCRIPTION

The Human-Robotic Interaction system used in this project was composed of three parts: the Adobe® Flash® interface, Unreal® Tournament 2004, and the USARSim Image Server. The USARSim Image Server grabbed frames from the Unreal® Tournament 2004 engine and converted them into a format that Flash® could read using FreeImage (FreeImage, 2007). The USARSim v3.12 extension for Unreal® Tournament was also used, so that working models of real robots in the Karma physics engine could be employed. The USARSim manual provided all the information needed for configuring the interface for USARSim (Wang and Balakirsky, 2007). This project required that the UnrealScript code for USARSim



Figure 2: The AirRobot® provided in USARSim

be altered in order to allow Flash® to send commands to Unreal® Tournament that would enable and change the MultiView class (Wang et al., 2005) according to which ground robot was currently selected. The UnrealWiki was a useful reference when editing UnrealScript (UnrealWiki, 2007). The Image Server's C++ code was also modified in order to reform the two MultiView subviews into a picture-in-picture image. Flash®'s ActionScript code required some of the most important changes, because it needed to interface to both ground robots and the AirRobot®. An image of the USARSim AirRobot® is provided in Figure 2. This robot has different driving controls than the ground robots.

Humphrey et al. had already modified the Image Server and developed the Flash® ActionScript to develop their human-robot interface (Humphrey et al., 2007). This work started with a version of their Image Server and Flash® code. The existing Image Server already grabbed frames from Unreal® Tournament and converted them into a JPEG format so that Flash® could read them. This project made changes by copying the MultiView subviews and reformatting them into a single picture-in-picture image. The FreeImage v3.9.3 (FreeImage, 2007) package allowed the images to be altered exactly as was required. The Flash® code was also updated, because it had been designed for an older version of USARSim and required the use of some functions that USARSim no longer supported. Beyond the changes required due to the new version of USARSim, the Flash® code also needed to be made compatible with both ground and air robots. This meant that behavioral code, as well as manual movement controls, had to be

altered so that the code also works with the AirRobot®.

The first change to the system, made before updating to USARSim v3.12, was a change to the ground robot's camera. This camera was coded to hover several meters above the robot and point directly at it, instead of being attached to the top of the robot and facing the same direction that the robot faced. This was accomplished by altering the robots' configuration files. It is anticipated that the operator will have situational awareness due to the broader image perspective, but that it was more difficult to maneuver in close quarters due to the lack of a forward-facing camera mounted on the robot. The overhead view provided additional help in getting the ground robot near a bomb for a simulated bomb detection task, but once near the bomb, the ground robot requires more precision. The camera was unable to zoom in or move closer to the robot, thus hindering the operator's task of defusing the bomb.

The second step in the system modification added a second camera to the ground robots. The first camera was reset so that it would rest on top of the robot and face forward, and the second camera was suspended high above the robot, facing straight down. This time, the interface allowed the operator to switch between the two different cameras so that the operator had the advantage of being able to see the objects around the robot in all directions, but could also drive more easily with the forward-facing camera. This modification appears to provide a broader environmental perspective and the ability to get to the bomb quicker because of the hovering camera image. Precision movements around the bomb are also possible because the operator can switch to the camera view mounted on the robot. See Figure 3 to view the camera portion of this interface.

Version 3.12 of USARSim included a model for a flying robot and a tool called MultiView that is able to display multiple cameras' viewpoints on the screen at the same time as split-screen views. The first step was to learn how to operate the flying robot, since it operated using new and unfamiliar parameters in its drive command. Next, the Flash® interface was altered to support generating the drive commands for either a ground robot or the flying robot. After that, the MultiView class in UnrealScript code was modified so that it could accept commands from the Flash® interface

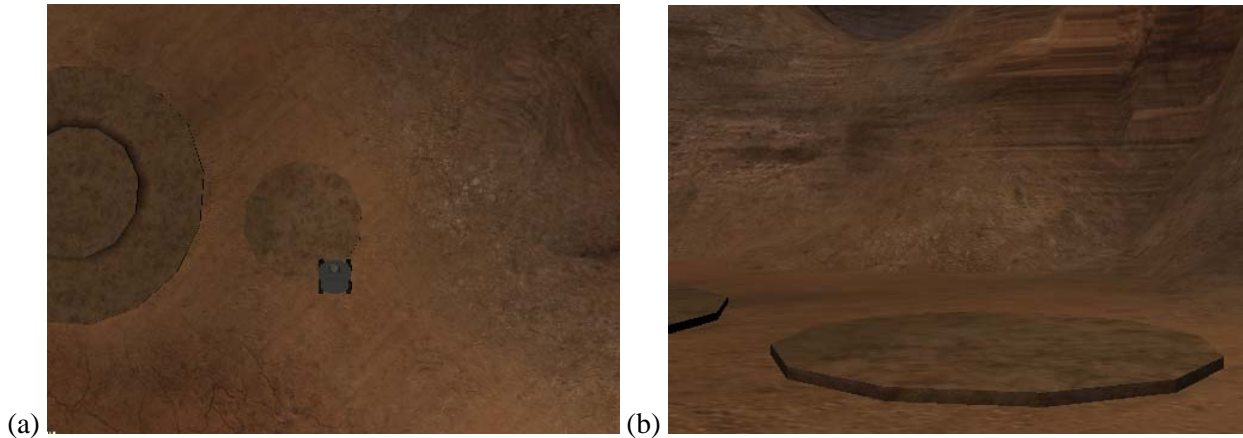


Figure 3 (a): The overhead camera view, (b): The ground-level view for the robot

and adjust which cameras it was displaying in each subview. Several additions had to be made to a few of the UnrealScript files before it worked, because the file containing the code that accepts and parses commands does not have direct access to the MultiView class.

The AirRobot® is a significant help because it uses the Karma physics engine to simulate the object that has the camera attached while hovering in the air looking down at the ground robots, rather than having an unexplained, floating camera defying gravity. Incorporating the AirRobot® into the simulation allows us to create a more feasible physical system. For example, when the AirRobot® moves, it must tilt a little, causing its camera to aim in a slightly different direction until it stops moving and the camera faces straight down again. The AirRobot® must fly from one ground robot's position to another's, introducing a time delay when the operator selects a different robot to work with. The two camera per robot system described earlier, allowed the views to instantaneously switch from one robot's overhead view to another's, since no traveling was necessary.

The MultiView object is also very helpful, because it allows the operator to view both the ground robot's mounted camera's viewpoint and the bird's-eye view of the surrounding area at the same time. Thus, the operator does not have to switch back and forth between the two views. The AirRobot® and the MultiView object combine the information of the overhead view with the precision of the robot's forward view, without having the inefficiency of switching back and forth between the two views while teleoperating a

robot. Figure 4 contains a screenshot of the camera portion of the interface using MultiView.

The MultiView object makes it much easier for the ImageServer to rearrange the two camera images into a single picture-in-picture image. With MultiView, the system does not need to take samples from two different sources, or have the cameras quickly switch back and forth each frame and integrate the images from those two frames. Instead, the ImageServer is able to grab single images from Unreal® Tournament 2004 and use FreeImage to enlarge the AirRobot®'s subview and integrate the ground robot's subview into the middle of it. Although there seems to be a significant drop in frame rate for the images that Unreal® Tournament is displaying based upon the images that the ImageServer is creating and placing in Flash®; however, the frame rate is still high enough to be able to navigate the robots in the simulation through Flash®. This low frame rate issue is addressed again in the Future Work section.

A true picture-in-picture display, as shown in Figure 5, was created as a result of the described steps. The existing software helped the process significantly; Humphrey et al.'s human-robot interface and USARSim were very helpful starting points for this project. FreeImage was another helpful tool that made the coding process much more efficient, since it takes care of image formatting and resizing algorithms automatically. FreeImage is an open-source tool that is useful for processing and reformatting image bitmap data on the bit level. It allows a programmer to easily change images from one image format to another,

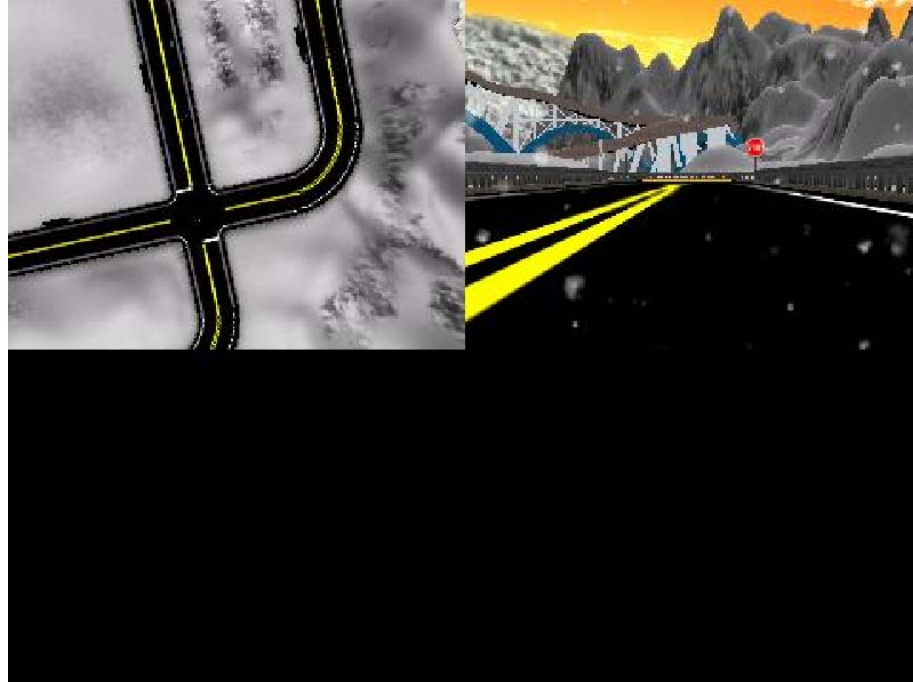


Figure 4: The image with MultiView

such as from BMP to JPEG (FreeImage, 2007). Among other things, it also contains functions that a programmer may use in order to resize or transform any particular image. It was used in this project to rescale the AirRobot®'s subview, then to paste the currently selected ground robot's subview into the middle of the newly rescaled image. It was also used to convert raw image bits into the JPEG format. Trace statements and output statements were the most common forms of debugging techniques used in this coding process, and there were frequent minor tests to determine if small pieces of code were functioning properly.

RESULTS

There were three major evaluations during the software development process. First, an evaluation was performed with the existing system, which was modified so that the robots had two cameras apiece, one on the robot and one floating in space, and the operator was able to switch between the two (see Figure 3). Second, an evaluation was performed after AirRobot® and MultiView had been integrated into the system (see Figure 4). Third, the last evaluation was performed with the ImageServer grabbing the frames from Unreal® Tournament 2004 and editing them to make a picture-in-picture image (see Figure 5).

When testing the existing system with the added ability of being able to switch between the ground robots' camera and the overhead, floating camera, it was determined that both the bird's-eye view and the ground view should both be visible at the same time when teleoperating the robots. It was perceived that the operator had some difficulty navigating the robot when switching between the two different views while the robot was moving.

The second evaluation determined that the two subviews for the AirRobot® and the ground robot were perceived to be helpful, but the AirRobot®'s subview seemed small, and the operator still had to focus on one or the other subview in order to navigate. See Figure 3 for the appearance of the camera portion of the interface during the second evaluation.

The third evaluation showed a significant decrease in the number of frames per second being displayed in the Flash® program. The ImageServer was at fault due to a memory leak, which was quickly found and fixed. Since the frame rate was no longer so much different from the earlier versions of the system, the final version, with the picture-in-picture functionality, was perceived to be the most useful in providing the



Figure 5: The final picture-in-picture version of the image

operator a broader environmental perspective without sacrificing navigational efficiency. The frame rate is still slightly slower than previous versions of the interface, and this issue is addressed again in the Future Work section. Overall, the picture-in-picture interface is perceived as more natural than the side-by-side interface that was implemented in the previous version. See Figure 4 for the appearance of the camera portion of the interface during the third evaluation.

DISCUSSION

The existing system was a great code base to build from, and the open-source projects, such as USARSim and FreeImage were a great help, but they were not completely bug free. The USARSim download was missing an important file, which caused some stress for a few days. Thankfully, this issue was fixed very quickly after the developers of USARSim were contacted. The existing Flash® program had a few minimal bugs, and all of the ones that showed up during this project's process were fixed.

Many others have experimented with a picture-in-picture interface, and it has had varying degrees of success, depending on the specific

application, but it does appear that this type of display may help for a multiple robot system. For unmanned aerial vehicles (UAVs), “the picture-in-picture concept itself did not improve performance”; however, “comments suggest an alternative instantiation of the concept will improve utility” (Calhoun et al., 2006). Calhoun et al. show that it is not the picture-in-picture concept itself, but its specific implementation that can help achieve a better interface design. My project's particular implementation of the picture-in-picture interface may provide the operator with better situational awareness, which is shown to be a key factor in multiple robot systems (Humphrey et al., 2007). With the increased situational awareness, the operator of a multiple robot system may have an easier time keeping the big picture in mind as he/she navigates the robots through the field looking for bomb-like objects that need to be defused.

Since the operator using the picture-in-picture interface may have an easier time keeping the big picture in mind, the use of multiple robots may also become more effective, when the ratio of operators to robots is one to many. The big picture and situational awareness are two concepts that may help single operators in multiple robot systems, and with this picture-in-picture interface,

those two things may be enhanced significantly. In effect, the picture-in-picture interface may make single operators of multiple robot teams more effective.

FUTURE WORK

Although many things have already been achieved, there is much future work to be done on this project. As is the case in any project, optimizing and simplifying the code is an important step in the software engineering process, but, unfortunately, there has not been enough time to completely optimize and simplify the code. Surely, a few bugs have been created while writing the code for the picture-in-picture interface. For example, an apparent memory leak problem in the version of Unreal® Tournament 2004 that I have altered, but there has not been enough time to identify the problem in the UnrealScript.

Another important future step is the complete automation of the AirRobot®. We would like the AirRobot® to fly to the selected ground vehicle and follow it, independent of any operator's control. Once the AirRobot®'s follow and movement behaviors are completed, it will be able to function on its own, without any input from the operator. This will allow the operator to focus more of his/her concentration on the ground robots and the task at hand, since the operator will not have to pay any attention to controlling the AirRobot®. Situational awareness, concentration, and the number of robots being controlled are all related (Humphrey et al., 2007), and the AirRobot® can still accomplish its goal without any control from an operator once its behavioral code has been implemented.

Other studies have shown that adding an overlay with important target or status information, such as approximate location, can improve performance (Calhoun et al., 2005). A future version of the picture-in-picture interface can include an overlay on the AirRobot®'s view, indicating the position and heading of any other ground robots in view with a circle or an arrow. This may enhance the picture-in-picture effect even more and may improve situational awareness for the operator. Primarily, the operator may be able to tell where the individual robots are, in relation to each other. The operator may also be

able to identify objects in the field of view of the AirRobot® more easily, since the ground robots would be clearly marked.

Another thing that would improve the picture-in-picture interface is a higher resolution for the images. Higher resolution should be fairly easily accomplished, and better quality images may help the operator. Right now, the resolution is low due to the frame rate issues, but once the code is optimized, a higher resolution should be possible.

User testing is a necessary future step. The picture-in-picture interface seems to provide a broader visual perspective, but only user testing can determine whether it will provide more situational awareness. User evaluations may also determine other interface improvements.

CONCLUSIONS

The primary goal of this project has been accomplished: another viewpoint and picture-in-picture functionality has been added to the existing simulation interface. In the process, I have updated the version of USARSim that the simulation is running, which will allow for possible expansion to the simulation in the future. The picture-in-picture interface provides a broader visual perspective to an operator controlling a team of multiple robots, which may increase his/her effectiveness significantly. Even though much has been accomplished, there is still a lot of future work that can be done to improve the existing picture-in-picture interface. Indeed, this has been an important milestone in the development of the picture-in-picture interface for a multiple robot system.

REFERENCES

This project has been partially supported by the 2007 Vanderbilt University School of Engineering Summer Research Program for Engineering Undergraduate Students and partially supported by an NSF Research Experiences for Undergraduates supplement to the NSF IIS-0643100 grant.

REFERENCES

Calhoun, G.L., Draper, M.H., Abernathy, M.F., Delgado, F., and Patzek, M. "Synthetic vision system for improving unmanned aerial vehicle operator situation awareness" Proc. SPIE Vol. 5802 p. 219-230, May 2005

Calhoun, G.L., Draper, M.H., Nelson, J.T., and Ruff, H.A.
"Advanced display concepts for UAV sensor operations:
Landmark cues and picture-in-picture" Proceedings of the
Human Factors and Ergonomics Society 50th Annual Meeting,
2006.

Drury, J., Scholtz, J., and Yanco, H.A. "Awareness in human-
robot interactions". Proceedings of the 2003 IEEE
International Conference on Systems, Man and Cybernetics,
pp. 912-918, 2003.

FreeImage, 2007. <<http://freeimage.sourceforge.net/>>.

Humphrey, C.M., Gordon, S.M., and Adams, J.A.
"Visualization of Multiple Robots During Team Activities".
In Proceedings of the 2006 Human Factors and Ergonomics
Society 50th Annual Meeting, San Francisco, CA, 2006.

Humphrey, C.M., Henk, C.P., Sewell, G., Williams, B.W.,
and Adams, J.A. "Assessing the scalability of a multiple robot
interface". Proceedings of the 2nd ACM/IEEE International
Conference on Human-Robotic Interaction, 2007.

UnrealWiki, 2007.
<http://wiki.beyondunreal.com/wiki/Home_Page>.

Wang, J., Balakirsky, S. "USARSIM V3.0 A Game Based
Simulation of the NIST Reference Arenas" USARSim
Manual, 2007.
<http://www.mirrorservice.org/sites/download.sourceforge.net/pub/sourceforge/u/us/usarsim/USARsim-manual_3.1.pdf>.

Wang, J., Lewis, M., Hughes, S., Koes, M., and Carpin, S.
"Validating USARSim for use in HRI Research". Proceedings
of the Human Factors and Ergonomics Society Annual
Meeting, pp. 457-461, 2005.

Yanco, H.A., and Drury, J. "Where am I? Acquiring situation
awareness using a remote robot platform". Proceedings of the
2004 IEEE International Conference on Systems, Man and
Cybernetics, pp. 2835-2840, 2004.

Zaratti, M. "MultiView.uc". USARSim.
<<http://usarsim.sourceforge.net/>>