Proceedings of the ASME 2015 International Mechanical Engineering Congress & Exposition
IMECE2015
November 13-19, 2015, Houston, Texas

IMECE2015-53617

CONTROL FOR CAMERA OF A TELEROBOTIC HUMAN COMPUTER INTERFACE

Antonio G. Sestito, Tyler M. Frasca, Aidan O'Rourke
Program of Electromechanical Engineering
Wentworth Institute of Technology
Boston, MA 02115, USA

Lili Ma
Dept. of Electrical Engineering and Technology
Wentworth Institute of Technology
Boston, MA 02115, USA
mal@wit.edu

Douglas E. Dow
Dept. of Electrical Engineering and Technology
Dept. of Biomedical Engineering
Wentworth Institute of Technology
Boston, MA 02115, USA
dowd@wit.edu

ABSTRACT

Controlling remote robots is a difficult task for human computer interface (HCI). Control of remote robots enables accomplishment of tasks without the human controller physically present due to safety concerns or the expert cannot be physically present. This paper presents a method for using an Oculus Rift to improve HCI for telerobotic control. Using the Oculus, an operator could become immersed in the robot's environment and could more naturally control the desired position of a remotely positioned vision system via head movements. To provide the appropriate visual feedback, a three-axis gimbal was implemented as a test platform. Through software implemented motion tracking, the response of the Oculus was compared to that of a mouse which demonstrates the efficiency of the proposed system over comparable HCI.

Keywords: HCI, Oculus Rift, head mounted display, gimbal, PlayStation Eye, Arbotix-M Robocontroller, head tracking, inertial measurement unit, IMU, accelerometer, gyroscope, magnetometer

INTRODUCTION

Human Computer Interface (HCI) is difficult for dynamic, multiple degree of freedom (DoF) systems such as controlling remote robots [2, 4]. Many tasks needed by society are dangerous for the person doing the task. Two examples of such tasks are toxic waste management and bomb defusal [1]. These tasks help keep communities safe, but require the personnel to put their lives in danger. Telemedicine is another example. Expert medical personnel would be best able to perform difficult emergency and surgical procedures, but the expert clinicians may be located far from the patient in need [3]. An effective HCI that would allow expert operators to control remote physical actuators using visual and other sensor inputs would enable many tasks, reduce safety risks, and save lives [4]. The HCI would allow an operator to assess a dynamic situation and properly control robotic actuators at a distance from the critical site [1]. Currently, typical HCI systems control robotic systems with either a computer or a handheld device with joysticks and buttons.

Simultaneously controlling multiple functions of a teleoperated robot proves difficult using tactile input methods, such as buttons and joysticks, especially if a camera view also requires tactile input for control. Although a human operator can control multiple functions simultaneously, relying primarily on tactile interfaces restricts the operator to a finite set of inputs. By requiring the operator to control the camera view with tactile input, a delay and level of misdirection is introduced. This increases the time between obtaining scene information and deciding on a desired action due to the translation from natural thought and behavior to equivalent input commands. By utilizing input methods that are more natural to the mind and body of the operator, less of an artificial conversion would be necessary. Therefore, responsiveness would be increased and human resources relieved for other controls. One way to do this would be to use a head-mounted vision system in which the operator physically turns their head to look, and the camera automatically turns to match the direction. As the camera is turned, the video would be passed back to the head mounted system so the operator sees that scene.

The aim of this project is to use natural head movements as the control input for the camera. Intuitively moving the head to control the camera direction would allow the operator to utilize both hands for alternative functions and alleviate some of the

delay in control. Such a system would require accurate motion sensing and low delay in order to be useful.

HCI for telerobotic control may be improved by using the human body as the controller. The control system should be intuitive with faster dynamic response, allowing for significantly more inputs for precision control. If the operator can move their head to position the camera, then response time may diminish due to the human natural reaction to visual stimuli. A 9-axis inertial measurement unit (IMU) containing an accelerometer, a gyroscope, and a magnetometer was used to measure the orientation, velocity, and gravitational forces to determine spatial positioning. The IMU supplied information for control of the camera actuators and to match the user's movements.

A prototype was developed and tested. Controlling the camera with head position instead of multiple joysticks would improve the control interface and would also increase available human resources that could be utilized for other applications.

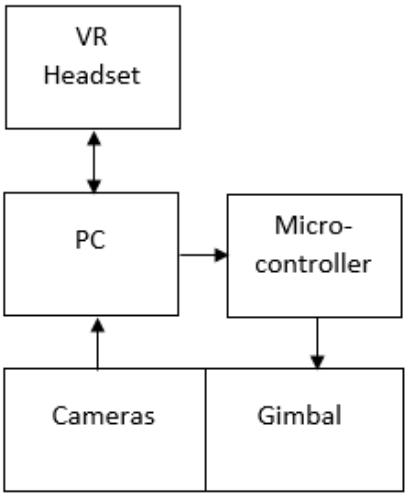

Figure 1: Flowchart describing the vision control system

METHODS

A system to implement control of camera view by head motion could utilize sampling of acceleration of the head to more quickly determine the angular position. The head position would then be mapped to corresponding robotic actuator control values, moving the camera appropriately. The recently developed Oculus Rift Head Mounted Display (Oculus, Menlo Park, CA) was selected for the prototype [5]. The Oculus Rift has a 9-axis IMU combining angular speed, acceleration, and polar direction, using a gyroscope, accelerometer and magnetometer. The IMU values are used to calculate angular position and compensate for drift error over time. The Oculus Rift also has software functions to perform position prediction for latency compensation [6]. This algorithm predicts future values which represent the position required at the time of the inputs arrival to the actuators. A gimbal platform to mount the camera has been assembled for the prototype, comprised of three high-resolution, high-speed Dynamixel AX-18A actuators with feedback capabilities [7].

The 3-Dimensional orientation of the Oculus Rift is projected onto this gimbal system to dynamically mimic the angle of the head.

System Overview

The Oculus Rift is used to give the user control and feedback of the robotic system. Figure 1 shows the overall block diagram of the prototype. The system is made up of an electrical system, an electromechanical system, and a control system. The Oculus Rift sends head position to the personal computer (PC) and receives a video feed from the PC. The PC gets video from the cameras, processes and sends to the Oculus Rift for display. The PC also processes the sensor data from the Oculus Rift then maps to servo positions and sends these to the microcontroller. The microcontroller then sends the specified values to the gimbal for motor control which will cause the camera mount to move and point the desired direction.

Oculus Rift

The Oculus Rift is promoted as a next-generation, virtual reality system which is meant to visually immerse the user. The immersion is accomplished through two low-persistence organic light-emitting diode (OLED) displays, one for each eye. Each OLED screen has a resolution of 960x1080 pixel and has a maximum refresh rate of 75 Hz. The Oculus rift also has internal tracking sensors in the form of a gyroscope, accelerometer, and a magnetometer. These sensors are used to track the heads position, direction, and speed [5, 6].

Figure 2: Diagram of head mounted Oculus Rift and camera on the gimbal. Position and angular rotation of the head mounted Oculus Rift is mapped to generate corresponding position and angular rotation in the camera on the gimbal. Video from the camera is mapped back to display in the Oculus Rift.

Cameras

Two cameras, (PlayStation Eye, Sony Corporation, Tokyo, Japan) were selected for this project because their specifications seemed to fit the project objectives. The cameras operate at a resolution of 640x480 pixel with a maximum frame rate of 120 Hz. They have a 75° field of view and transmit the frames over USB [8].

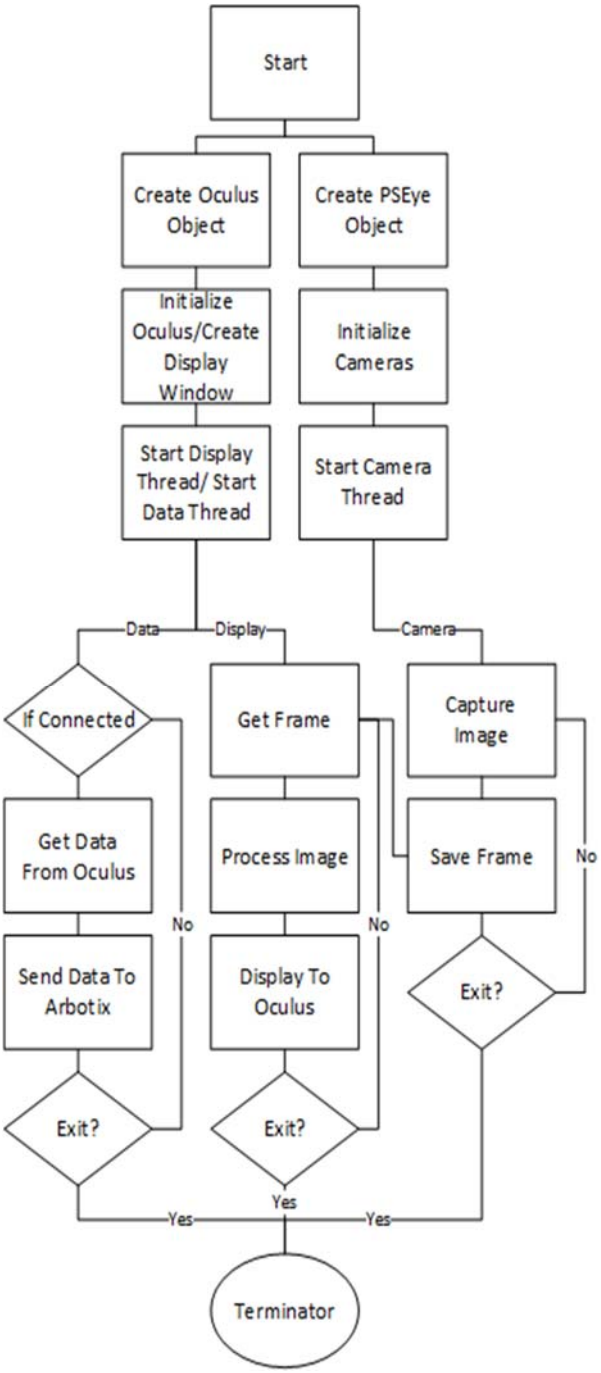

Figure 3: Oculus and Camera Program Flow Chart for Data Acquisition and Control

Arbotix-M Microcontroller

The controller for the prototype was the Arbotix-M Robocontroller. Based off the Arduino Microcontroller, the Arbotix-M Microcontroller serves as the intermediate step between the computer and the three-axis gimbal. The board has a 16 MHz ATMEGA644p (Atmel, San Jose, California) chip as the processor. Unlike the Arduino microcontrollers, the Arbotix has three dedicated 3-pin Dynamixel TTL half-duplex serial ports, making it easier to connect the microcontroller to the gimbal. An important aspect of the Arbotix microcontroller is compatibility with Arduino, and thus having a large open source community [9].

Three Axis Gimbal

For testing the system, a pre-designed, two-axis gimbal was combined with another actuator to assemble the required three axes. The gimbal is comprised of three Dynamixel AX-18A Robotic Actuators. Each AX-18A actuator is equipped with a dedicated microcontroller, allowing the actuator to set the compliance margin and slope, the torque, and the goal position among other attributes. The actuators also have the capability to store the current position and the speed, which can be used for feedback control. The servos have a stall torque of 1.8 Newton meter and a resolution of 0.29 degrees [7].

PC

The control system of the prototype was a Windows PC which ran the code that interconnected all of the modules. The PC used to control the system was a Lenovo T440p, having an Intel core i7-4700MQ Processor which ran up to 3.4 GHz. The operating system on the laptop was Windows 8.1 Professional x64 which allowed for full utilization of the 8 GB of RAM installed. The laptop has a 256 GB Solid State Drive. The laptop was used for communication between modules which was facilitated using custom C++ code developed using Visual Studio 2013 that was developed for this project.

Programming

The programming of the system was a primary focus of the project. In order to accomplish the control, three subsystems needed to communicate: The Oculus, the PSEye cameras, and the Arbotix microcontroller (Figure 1). Each of these were handled individually by creating a class for the Oculus functions, the PSEye functions, and the serial communication separately. The program started by initializing a new instance of these classes which handled their own setup.

When the Oculus object was instantiated, all necessary setup procedures and connection status checks were completed regarding the basic function of the Oculus Rift. This class also handled the creation of the window for display as well as setting the required graphics parameters [10, 11]. The serial communication class was also created within the Oculus object in order to establish a connection to the Arbotix controller [12]. After setup was complete, two threads were created within this object to allow for simultaneous operation. These handled the data acquisition and the display function independently. The data thread read the orientation data from the Oculus, converted the native values to a corresponding servo position value, and sent the servo position value to the Arbotix controller through the serial connection. The servo control on the gimbal was handled independently by the Arbotix controller which simply set the servos to the received positions. The display thread

constantly updated the image displayed on the HUD with the image received from the PSEye class through native functions integrated into the Oculus SDK. These functions handled the distortion to compensate for the lenses in the Oculus as well as the image stitching for stereo output.

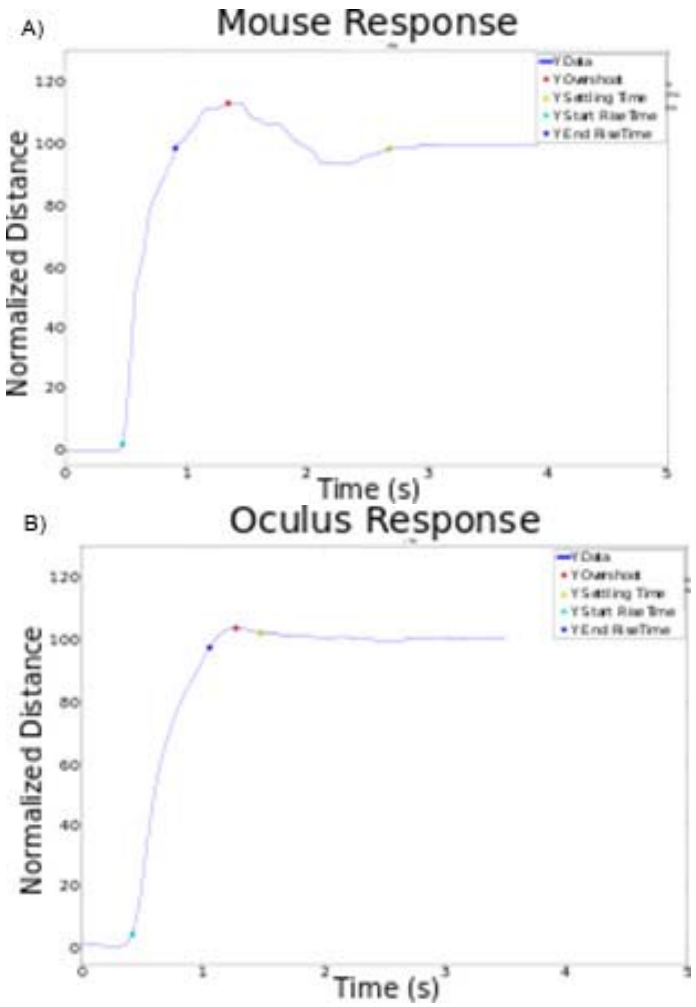

Figure 4: Characteristic response for both the Mouse (6.A) and Oculus (6.B) test. The Mouse system was underdamped with an overshoot of around 18%. Due to the oscillations of the system, the system did not settle until 2.75 seconds. The Oculus system was also underdamped, however, the system had a smaller percent overshoot of 5%. The system did not undergo the same oscillation, and thus settled around 1.5 seconds.

When the PSEye object was created, a connection was established to the two cameras for stereo vision. This process first scanned for the number of cameras found, ensured there were exactly two, and then tested functionality until proven to be usable. Once setup was complete, a third thread was created which constantly captured an image from each of the cameras, converted them to a usable texture, and then updated the container located in the Oculus class with the new textures [13].

Each of the three threads which enabled the control of the system constantly ran as long as a specific control variable was true. The main program continuously waited for an input command alongside the functioning program, and upon the receipt of the escape command sets this control variable to false. This indirectly terminated the threads and began the shutdown procedures for each class which ensured the program closed all aspects correctly. A flow chart of the overall program is shown in Figure 3.

Table 1: Results from testing mouse and Oculus

	Rise Time (s)	
	X	**Y**
Mouse	0.714 ± 1.00	2.611 ± 2.09
Oculus	0.448 ± 0.617	1.68 ± 1.045
	Percent Overshoot	
	X	**Y**
Mouse	8.177 ± 16.90	11.93 ± 27.25
Oculus	3.761 ± 4.744	2.736 ± 4.077
	Settling Time (s)	
	X	**Y**
Mouse	3.433 ± 1.930	3.969 ± 2.700
Oculus	2.148 ± 0.807	2.149 ± 0.830

TESTING

Preliminary testing was completed using the three-axis gimbal system to determine the feasibility of the system in relation to visual feedback and latency. A human wore the Ouculus Rift and moved the camera to follow a LED pointer. Due to communication delays, comparing the response of the gimbal from the mouse and the oculus proves difficult. To evaluate the capability of the devised control system versus current methods, the initial procedure was altered to acquire the response characteristics of the operator instead of the output from the gimbal. A computer mouse was chosen as a current method for comparison due to its wide use in remote applications and the control in question was the control through head orientation using the Oculus Rift. To simplify the testing, the gimbal system itself was not used which eliminated any delay or interference through communication which is present regardless of the control method. A vision system was simultaneously simulated and rendered on both the computer screen and the Oculus Rift display which was controlled by each respective

method. The test required the user to locate and point at a randomly generated object on the screen and move the cursor towards the direction of the object using the current control. The cursor represented the camera, so moving a cursor toward an object would be pointing the camera toward the object. When the object was identified as being in the center of the field of view, a new object would be generated and the process began again. As the user moved the camera view to match the desired position, the X and Y pixel locations of the object relative to the center of the field of view was recorded to a file along with a timestamp until a new object was generated. This data was then able to be processed off line later to determine rise time, settling time, and overshoot based on each control method which together represented the responsiveness and accuracy of each method. Fifteen users were tested overall, each producing thirty samples from each control method. One sample involved the timeframe between the generation and location of the generated object. The individuals used during testing ranged in computer capability, age, gender, and video game experience.

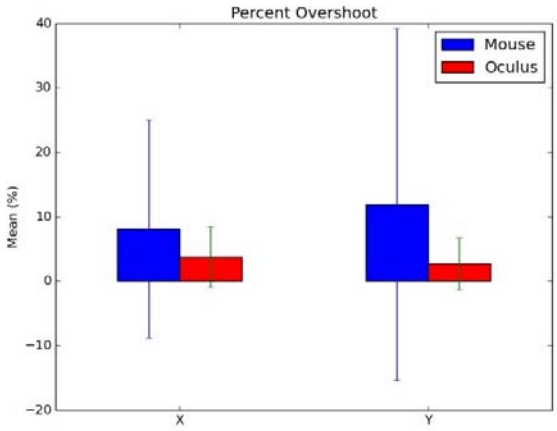

Figure 5: Histogram comparing the Mouse and Oculus percent overshoot in both the X and Y direction

RESULTS

Observations from preliminary testing involving a side by side comparison of movements between a human wearing the oculus and the three-axis gimbal showed that the movements mirrored one another. The human subject naturally moved their head to successfully follow the LED pointer by controlling the three-axis gimbal. This initially confirmed the concept of the system, but more was required to confirm the success of the system.

Three important characteristics when comparing the mouse and Oculus tests are the rise time, the percent overshoot and the settling time. The rise time measured the required time for the response to rise from either 0% to 100% for underdamped systems or 5% to 95% overdamped systems. This determined how quick the operator could respond to an event. The percent overshoot measured the amount which the output response exceeded the final value. The overshoot is an important characteristic, because the more the overshoot the more the output will oscillate and thus take longer for the output to settle to the target position. The last characteristic used to compare the mouse and the Oculus tests was time required for the output to reach and stay within 2% of the target value.

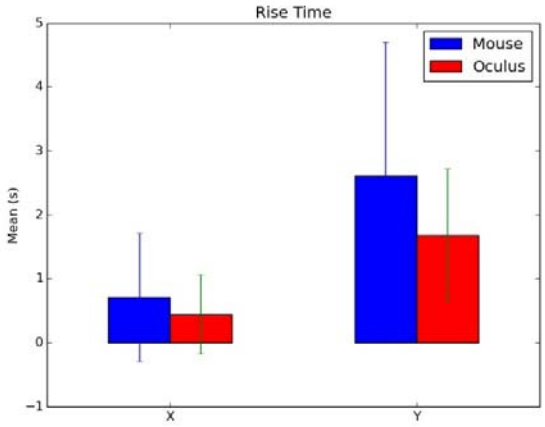

Figure 6: Histogram comparing the Mouse and Oculus percent rise time in both the X and Y direction

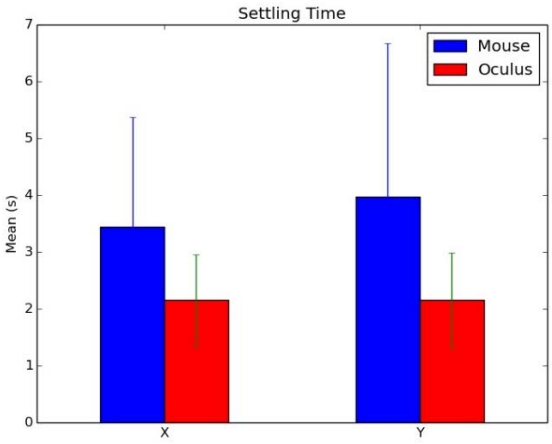

Figure 7: Histogram comparing the Mouse and Oculus percent settling time in both the X and Y direction

Because the tests were performed in a two dimensional space, the three characteristics of the mouse and the Oculus were compared in both the X and Y directions. Figures 4-7 compare all of the responses for each characteristic in both the X and Y direction. Table 1 shows the average of these results. It is evident that the Oculus outperforms the mouse in every category.

CONCLUSION

The operator's response to an event needs to be quick without much oscillation. The more intuitive the movement, the easier it will be for the operator to improve their response to an event seen through a vision system. From the preliminary tests, the Oculus improves the three response characteristics. The

results show that using the Oculus Rift to control cameras with natural head movements improve the HCI. Currently, the system is limited to the precision and response of the Oculus system, as well as the processing power of the host computer. More testing needs to be completed in order to appropriately determine the success of the Oculus for controlling telerobotic vision systems. Future work in feedback control and prediction algorithms could tremendously improve the system and lead into hardware designed specifically for this application.

REFERENCES

[1] Anca D. Dragan, Siddhartha S. Srinivasa. “Teleoperation with Intelligent and Customizable Interfaces.” Journal of Human-Robot Interaction, Vol. 2, No. 2, 2013, Pages 33-57.

[2] "Human Computer Interface Issues." *Integrated Communications Management of Broadband Networks*. Crete, Greece: Crete UP, 1996. N. pag. Web. 08 Feb. 2015. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.9224&rep=rep1&type=pdf>.

[3] Jianhong Cui , Sabri Tosunoglu , Rodney Roberts, Carl Moore, Daniel W. Repperger. “A Review of Teleoperation System Control” Florida Conference, May 8-9, 2003.

[4] Okamura, A.M. "Methods for Haptic Feedback in Teleoperated Robot-assisted Surgery." *The Industrial Robot*. U.S. National Library of Medicine, 31 Dec. 2004. Web. 12 Feb. 2015.

[5] "Create the next Generation of VirtualÂ reality." *Oculus Rift PC SDK, Samsung Gear VR Mobile SDK, Docs, Integrations*. N.p., n.d. Web. 16 Sept. 2014.

[6] *Oculus Developer Guide*. Irvine, CA: Oculus VR, LLC, 3 Dec. 2014. PDF.

[7] "Dynamixel AX-18A Robot Actuator." *Trossen Robotics*. Robotis, n.d. Web. 03 Mar. 2015. <http://www.trossenrobotics.com/dynamixel-ax-18A-robot-actuator.aspx>.

[8] *Playstation Eye Datasheet*. Europe: Sony Computer Entertainment, 2007. PDF.

[9] *Arbotix-M Robot ControllerManual*. N.p.: n.p., n.d. PDF.

[10] "GLFW - An OpenGL Library." *GLFW - An OpenGL Library*. N.p., n.d. Web. 15 Mar. 2015. <http://www.glfw.org/>.

[11] "NeHe Productions - Everything OpenGL." *NeHe Productions - Everything OpenGL*. N.p., n.d. Web. 15 Mar. 2015. <http://nehe.gamedev.net/>.

[12] "Serial Communications." *Serial Communications*. Microsoft, n.d. Web. 10 Mar. 2015. <https://msdn.microsoft.com/en-us/library/ff802693.aspx>.

[13] "PS3 Eye." *CL - Research*. Code Laboratories, n.d. Web. 20 Nov. 2014. <https://codelaboratories.com/research/view/ps3eye-not-your-typical-webcam>.