INCREASING SAFETY AND ACCURACY IN AUTOMATION SYSTEMS WITH REAL-TIME ROBOT MONITORING

Rohan Dixit, Poolesville High School

Abstract:

With advances in the world of automation and the increased presence of robots in industrial settings, it is essential that robot systems have protective measures to prevent accidents from occurring. Faults within the robot control system or the environment may lead to erratic behavior resulting in injuries to employees, defects to products, and damage to the environment and other machines. Robots are at risk of malfunctioning due to worn out parts, failures in the control system, or cyber-attack. To mitigate this risk, a robot monitoring system was designed to track a robot's movements in real-time and check the performance of the robot. A Kinect sensor was placed above the robotic enclave to act as a camera that records all of the robot's movements; this live video stream is processed frame by frame with the original video stream to determine if a significant difference between the two exists. If one is detected, indicating a system error, the program will terminate by stopping the robot to prevent any potential harm. Of the four tested comparison methods specified by OpenCV, Chi-square was determined through testing to be the most effective one for implementation in the final program. The system designed is a viable solution to detecting errors in manufacturing robots and is a concept that eliminates weaknesses in existing technologies, like cybersecurity. A similar monitoring system could be applied in an industrial setting by positioning a camera to oversee the robots and performing image comparison methods to ensure production is occurring as expected.

Personal Section

Computers. Who doesn't use one these days? I have always been interested in computers. At the age of one, I would sit in front of the boxy desktop screen, moving the mouse while in awe by the moving cursor. As I grew up, my interest in computers expanded and I started to learn about the function of basic components like the CPU, motherboard, memory, and so on.

However, my real interest in computer science began in 9th grade when I first walked through the doors of Poolesville High School. As a member of Poolesville's Science, Mathematics, and Computer Science magnet program, I was exposed to a variety of unique introductory computer science classes. I learned programming and the fundamentals of computer science from my two teachers Mr. Estep and Mr. Stansbury. Outside of class, both of them supported my computer science learning through the after-school club Computer Team. Through this club, I learned about deeper aspects of computer science, like cybersecurity, which encouraged me to enter several local and national competitions that fostered learning and applying my newly acquired skills in the field. Through school, I recognized that I was passionate about computer science and that I wanted to continue pursuing the field in my future.

I was fortunate enough to have the support of a number of teachers as I began my senior research project. The head of the magnet program, Mr. Curran, and my research teacher, Dr. Miller, provided recommendations and the initial guidance to help me identify suitable internships. I knew of past students working at various labs, but ever since I went on a field trip in 9th grade to the National Institutes of Standards and Technology (NIST), I had a burning desire to work there. I looked up several ongoing projects at NIST and emailed several researchers. My first few applications were unsuccessful, but Mr. Curran and Dr. Miller gave me the support and

encouragement to keep searching. In the end, I was fortunate to receive the opportunity to work at NIST as an intern under Mr. Keith Stouffer.

For my internship, I received the freedom to do the work that I wanted, and I was able to explore topics that I was interested in: robotics and computer science. In addition to Mr. Stouffer, I got support from other researchers in the lab, which included Mr. Timothy Zimmerman, Mr. Frederick Proctor, and Mr. Richard Candell. They helped me learn in several ways, such as various Linux commands which I was unaccustomed to, and provided suggestions on improving my project. Although the focus of my project was computer science, I still had to learn about some advanced mathematics topics since histogram comparison, the basis for my project, uses metrics with different mathematical formulas.

A lot of students think that doing an internship over the summer is a waste of time and can be boring because you are only learning like school with no benefit. Even I initially thought the same way, but by doing this project, I learned of practical applications of computer science. It differed from school where I just had to learn theory. I was able to gain more of a real-life experience of what computer scientists do and it solidified my desire to pursue a career in computer science this coming fall at the University of Maryland.

Doing research sounds like it could potentially be a lot of work but it is an extremely rewarding experience, especially because you get the chance to explore your passions and do something that you enjoy. Not many high school students get the opportunity to perform research so seizing an opportunity helps set you apart from everyone else. I was able to perform my work in a professional work environment. By having a first-hand experience, I became more self-confident to do similar work in the future and having this experience under my belt gives me a bit of an edge for future opportunities. Whenever I talk to other professionals at conferences or

competitions, they are very impressed that I performed such high-quality work. After completing my research, I got the chance to write up a research paper to present my work. I submitted this paper to several competitions and won awards ranging from local science fair community awards to Intel Science Talent Search semifinalist.

The hardest part is actually acquiring an internship, but if you remain ambitious in your pursuits, you will find something that you enjoy working on. My teachers and friends were especially helpful and provided guidance to help me find out what I wanted to do and how I should approach getting my internship. If you are proactive and willing to prepare for your future, go for it and you will undoubtedly have a valuable experience.

Below you will find the research paper I submitted to Intel. If you have any questions, feel free to email me at rohan.s.dixit@gmail.com.

Research Section

Rationale:

Automated devices, commonly referred to as robots, have had an increased presence in the field of manufacturing. Over the last five years, there has been about a 10% increase per year in industrial robots sold, adding up to over 1.6 million industrial robots[3]. Robots are commonly used in assembly lines to carry out repetitive functions to help produce a certain product. An overlooked issue is that these robots are at risk of malfunctioning due to manufacturing defects, worn out parts, or failures in other areas of the inter-connected system. Also, a cyber-attack on the control system is possible. In 2014, the number of reported Supervisory Control and Data Acquisition (SCADA) control system attacks doubled from the previous year[4]. Moreover, many similar incidents are not reported since they do not involve acquisition of personal information.

A minor error in the robot's process could result in product defects and liabilities for the company and complaints by the general public. Erratic robot behavior may also result in injury to humans and severe damage to the robot and its environment (including other machines). Studies have shown that robot accidents occur when a robot unintentionally strikes a worker because of unusual activity or when an operator is not aware that the system is running[7]. Since 2000, the Occupational Safety and Health Administration (OSHA) has reported 37 severe robot/human incidents[7]. It is important that these robot control systems have a method of ensuring that the robot is functioning properly to prevent any harm. Developing a system to recognize irregular behavior can prevent many accidents by making sure that the process is being carried out as it supposed to.

Many industrial companies are developing more effective cybersecurity technology that can be used to combat hackers. For example, firewalls and secure protocols, like SSH, are being implemented along with segmented network structures to reduce the impact of a potential attack on the control system. However, there are still ways that a hacker can break into a system and spoof their identity so that a detection system is unaware of the unauthorized entry. Additionally, the implementation of cybersecurity technology slows down system processes, making companies hesitant to adopt such counter-measures. Traditional sensor-based safeguards within the control system are unreliable when then system is compromised in a cyber-attack. Therefore, it is necessary that other systems exist to verify that the control system is functioning properly.

Objectives:

The objective of the research project was to create a vision-based system to monitor a robot's movements in real-time. The system provides a mechanism to monitor the control system's function and check that there are no errors in the system. Video feed captured by a camera (Microsoft Kinect sensor) is compared to how the robot is supposed to function at a particular instant of time to determine any discrepancies using image comparison. These discrepancies are detected using threshold values. For example, if a measured relationship value is below the threshold, one can determine that the robot is not in the position it is supposed to be in. In addition to this, a message can be sent to the system manager or the robot system can be shutdown to prevent any damage to its surroundings. This system was also designed to be secure and not disrupt system processes.

A program was created using Open Source Computer Vision (OpenCV) histogram comparison methods that compares video footage in real-time. Histogram comparison, or

histogram matching, is a method of image processing that involves comparing the color content of two images based on their color histograms. Color histograms display the distribution of red, green, and blue values. This method was chosen over other image processing techniques because it is a fast solution, which is necessary to keep up with real-time visual monitoring. Histogram comparison involves splitting the color histogram into a number of bins in which the color pixels are counted[1]. The chi-square OpenCV metric was chosen for implementation in the final program since it proved to be the most accurate of the OpenCV metrics in detecting changes in the video feed.

Procedures:

A robotic enclave was modified to be used as the testbed for the robot monitoring system. The robotic enclave consists of two KUKA Youbots with customized and 3D printed grippers that can pick up the golf balls. The stations in which the golf balls rest were also manufactured using 3D printed parts. A Microsoft Kinect sensor was mounted above the robotic enclave which allowed the Kinect to capture the images of the robot moving the golf ball around the three stations with a full perspective of the enclave.

Two KUKA Youbots were programmed to perform the collaboration task of moving golf balls around five different stations, like a manufacturing line. Once the simulation was completed, it was modified to include a monitoring system using the Kinect sensor. A series of programs were created for the monitoring system. Methods from the OpenCV library for image processing and mathematical calculations were incorporated and modified to determine a particular relationship between two images. Four different histogram comparison metrics

specified by OpenCV were used: Correlation, Chi-square, Intersection, and Hellinger. The table below summarizes these four metrics.

Table 1: Summary of OpenCV Metric Ranges

OpenCV Metric	**Strongest Relationship**	**Weakest Relationship**
Correlation	1	0
Chi-Square	0	∞
Intersection	∞	0
Hellinger	0	∞

These metrics are calculated by the following four mathematical equations:[9]

Correlation: $$d(H_1,H_2)=\frac{\sum_I (H_1(I)-\bar{H_1})(H_2(I)-\bar{H_2})}{\sqrt{\sum_I (H_1(I)-\bar{H_1})^2 \sum_I (H_2(I)-\bar{H_2})^2}}$$, where $$\bar{H_k}=\frac{1}{N}\sum_J H_k(J)$$

Chi-Square: $$d(H_1,H_2)=\sum_I \frac{(H_1(I)-H_2(I))^2}{H_1(I)}$$

Intersection: $$d(H_1,H_2)=\sum_I \min(H_1(I),H_2(I))$$

Hellinger: $$d(H_1,H_2)=\sqrt{1-\frac{1}{\sqrt{\bar{H_1}\bar{H_2}N^2}}\sum_I \sqrt{H_1(I)\cdot H_2(I)}}$$

H_1 and H_2 are the two color histograms to be compared

N = number of histogram bins

The Correlation metric is based on the normalized cross-correlation operator in signal processing used to measure similarity. Chi-square sums the normalized square difference between the histogram bins, while Intersection returns the minimum comparison value of the bins. Lastly, Hellinger is commonly used to estimate similarities in probability distributions [5]. The most effective metric was to be determined for implementation in the program.

Code Function

In the start script for the two-arm collaboration process, a launch file was written that would record a video stream and save it in a file. Simultaneously, another launch file would process the stream frame by frame converting and storing the frames in image files every 0.1 seconds. The third program is written in Python and uses the histogram comparison functions from the OpenCV library. The program compares the two images and determines a relationship between the two. This relationship value is returned and can be compared to a threshold to determine if the robot is functioning properly. The threshold values are to be specified after defining a base set of values when the process is running correctly. If the robot is malfunctioning and the robot position is incorrect, a message is sent to the master computer and the system is shutdown to prevent any potential damage.

The first run through the program creates a baseline video stream with its accompanying image frames. Therefore, the third program would not run since there is no image comparison necessary. For the following runs, the third program runs to compare the image frames as they come in to the respective frame from the first run. The image files taken from each completed run are deleted to save computer memory.

To ensure that the file to be checked is present and to prevent any errors in the program, a statement was added to check whether or not the file exists. If not, the program continually checks for the file every five milliseconds while avoiding a FileNotFoundError. This allows the program to have a maximum of five millisecond lag, keeping the process running in real-time and making sure the program is efficient to keep up with the images as they come in.

Testing:

The process was run four times, once to establish a standard for comparison and then perform three test cases:

Base Configuration - When the process was running correctly: Running the process correctly allowed the researcher to determine an acceptable range of values for when the process is functioning correctly, which were then used to develop a threshold value for comparison.

Test Case 1 - When it was running excessively slow: Running the process slowly simulates when communications are slowed down. This may occur when the system is undergoing cyber-attack because of excessive network traffic.

Test Case 2 - When the commands had been manipulated: Manipulated commands simulate faulty operation, which is another potential effect of cyber-attacks.

Test Case 3 - When an object entered the robotic enclave: The last test scenario, with a foreign object, represents a case when a human may be present within a robotic enclave.

It is important that the robotic system is able to determine all of these discrepancies to minimize any damage. For each of the test cases, the process loop was run five times, so that the image comparisons were made four times.

Results:

Base Configuration: Process Running Correctly

To determine the threshold values, the process was run in a normal scenario. The table below lists the minimum, maximum, average, and standard deviation values calculated for each of the OpenCV methods.

Table 2: Process Running Correctly Run Statistics

OpenCV Method	Weakest Relationship	Strongest Relationship	Average Value
Correlation	.986	.999	.994
Chi-Square	.375	.004	.075
Hellinger	.147	.013	.093

The Intersection values did not provide much differentiation and there seemed to be no base value that could be used as a threshold for comparison. Therefore, the Intersection method for image comparison was not chosen.

Based on the values collected, the threshold values for each of the methods were calculated to be:

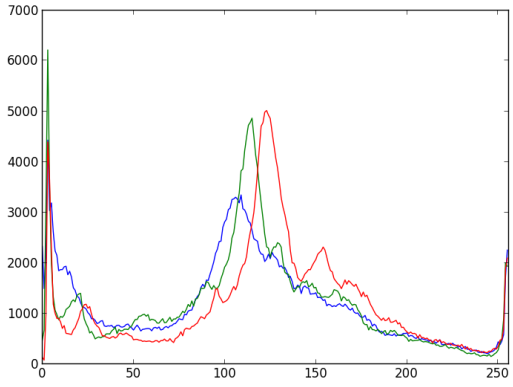
Table 3: Threshold Values

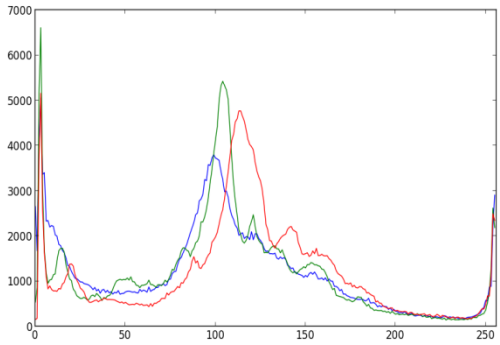
OpenCV Method	Threshold Value
Correlation	.95

Chi-Square	.5
Hellinger	.15

Figures 2 and 3: 11 seconds into process loop (process running correctly)

As pictured above, one can see that the robot is approximately at the same location, as expected at a particular instant of time. Therefore, there is a strong relationship between the two images. The following two graphs display the color histograms obtained for each image.





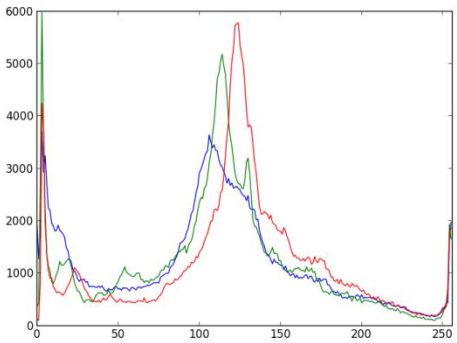
Figures 4 and 5: Color histograms for Figures 2 and 3. The two graphs have approximately the same values, resulting in a strong relationship between the two images

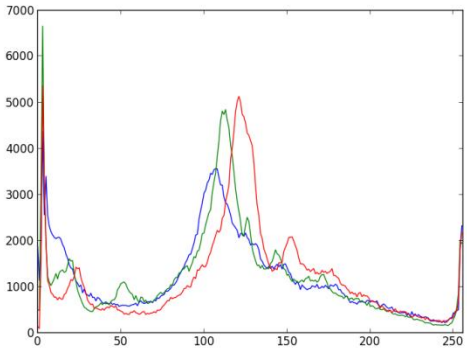
Test Case 1: Process Running Slowly

For this test case scenario, the robot simulation was programmed with an excessive number of delays, slowing down the entire process and communications to the robot. In the case of a cyber-attack, such as a denial-of-service attack, the communications between the robots may slow down due to excessive traffic on the network.

Figure 6: Position of robots when operating correctly

Figure 7: Position of robots when communications delayed





Figures 8 and 9: Color histograms for Figures 6 and 7. Although Figure 9 has a different y-scale, the values for the red pixel are still significantly different, resulting in error detection

Listed below are the average values for the OpenCV methods through all trials.

Table 4: Process Running with Delays Run Statistics

OpenCV Method	**Average Value**	**Did System Detect Discrepancy?**
Correlation	.937	Yes
Chi-Square	.525	Yes
Hellinger	.183	Yes

The only OpenCV measurement method able to detect an irregularity in the system function was Chi-Square. Although the process still functions correctly, it is important that a system manager is notified because a slow process would slow down a company's production. Loss in product yield can severely impact the company's profits. Also, if the robot is working with other robots, a slowdown may cause the overall manufacturing system to get de-synchronized.

Test Case 2: Changed Commands

To simulate a cyber-attack, the robot's instructions were modified within the source code. An actual attack, such as a denial-of-service, was not conducted because the robotic enclave's network system had not been configured to handle foreign traffic. For this test, the code was changed to move joint position 1, the base, 10 degrees to the left, moving the location of the end effector about an inch.

The table below displays the data for all trials when this test case was run.

Table 5: Process with Modified Commands Statistics

OpenCV Method	**Average Value**	**Did System Detect Discrepancy?**
Correlation	.967	No
Chi-Square	.525	Yes
Hellinger	.132	No

The images and color histograms for how the robot should have looked and how it actually looked are depicted below.

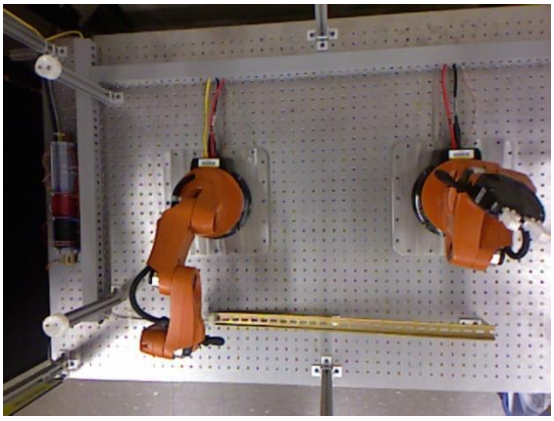


Figure 10: Position of robot when acting properly

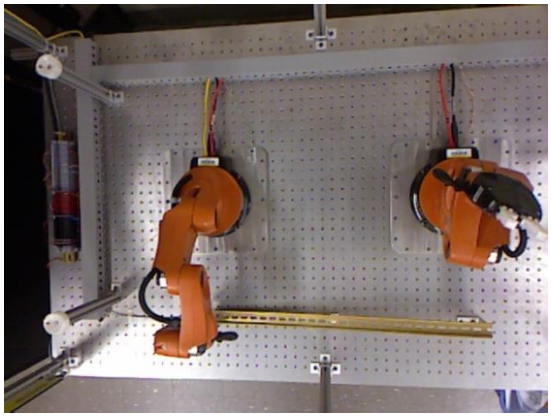
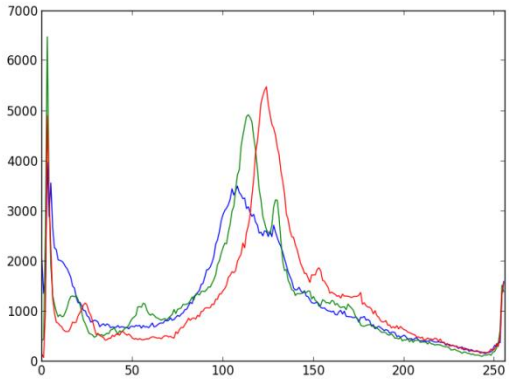
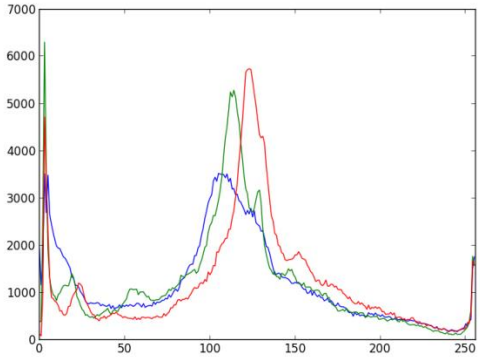


Figure 11: Position of robot with commands changed





Figures 12 and 13: Color histograms for Figures 10 and 11. The red values for Figure 13 are slightly greater than those of Figure 12.

The robot system was only able to detect the changes for the chi-square method within 0.3 seconds, and not very responsively for correlation and Hellinger. It is necessary that the robot monitoring system is capable of detecting the minor change to ensure manufacturing accuracy. The movement of a part by a small distance, even an inch or smaller, can cause the manufactured product to fail to meet dimensional tolerances or cause other defects within the products.

Test Case 3: Foreign Object

In many robot/human accidents, the human worker is not aware that the robot is still turned on and is at risk of being struck by the robotic arm. The final aspect of the program was checked by the insertion of a foreign object in the robotic enclave. During testing, the foreign object was a piece of paper, just to ensure safety in the case that the system did not turn off.

When the piece of paper entered the robotic enclave, the robot system was shut down immediately because the chi-square value exceeded .8, the threshold value. As the object became more visible in the image, the value continued to rise showing that the program is able to

recognize changes in the robot environment. Similarly for Correlation and Hellinger, the program returned values below the threshold value and the system shut down.

Figure 14: System shut down at this point for Chi-square test (Piece of paper indicated within black box)

Table 6: Process with Foreign Object Statistics

OpenCV Method	**Value When Paper Entered Enclave**	**Did System Detect Discrepancy?**
Correlation	.942	Yes
Chi-Square	.987	Yes
Hellinger	.190	Yes

Conclusions:

As can be seen from the tables for the test cases above, it was determined that Chi-square was the best OpenCV metric to be used for histogram comparison. Of the four metrics, Chi-square was able to effectively detect changes in the robots' environment for all of the tested scenarios and report these discrepancies back to the human operator. One potential reason for

this result is that Chi-square considers all of the histogram bins and their similarities. This metric could be modified to further increase accuracy in the system, further minimizing errors.

The system designed using the Kinect sensor is able to accurately detect discrepancies in robot's behavior and is an effective solution to monitor robots' movements throughout a manufacturing process being performed. The monitoring capability can be used to improve safety and eliminate manufacturing errors in automation systems.

A monitoring system like this one can be easily modified to fit an industrial setting since it consists of a few basic parts. A Kinect sensor could be replaced by any other type of camera as long as it is able to send a video stream or images. Also, the set of programs can be modified as needed to perform different image comparison techniques. The research conducted provides a basis that can be adapted and implemented to work in an actual manufacturing plant.

Bibliography:

[1]Histogram calculation. (2015, February 25). Retrieved July 25, 2015, from http://docs.opencv.org/doc/tutorials/imgproc/histograms/histogram_calculation/histogram_calculation.html

[2]Histograms - 1 : find, plot, analyze. (n.d.). Retrieved August 22, 2015, from http://docs.opencv.org/master/d1/db7/tutorial_py_histogram_begins.html#gsc.tab=0

[3]International Federation of Robotics (IFR). (n.d.). Industrial robot statistics. Retrieved August 30, 2015, from http://www.ifr.org/industrial-robots/statistics/

[4]Korolov, M. (2015, April 17). Attacks against industrial control systems doubled last year. Retrieved September 4, 2015, from http://www.csoonline.com/article/2911160/cyber-attacks-espionage/attacks-against-industrial-control-systems-double.html

[5]Laganière, R. (2011). Retrieving similar images using histogram comparison. In *OpenCV 2 computer vision application programming cookbook* (pp. 114-116). Birmingham, UK: Packt.

[6]National Institute of Standards and Technology (NIST). (2014, December). *Measuring impact of cybersecurity on the performance of industrial control systems* (K. Stouffer & R. Candell, Authors). Gaithersburg, MD.

[7]Occupational Safety & Health Administration (OSHA). (n.d.). Industrial robots and robot system safety. Retrieved August 7, 2015, from https://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html

[8]OpenCV. (n.d.). Retrieved July 20, 2015, from http://opencv.org/

[9]OpenCV. (2015, February 25). Histogram comparison. Retrieved July 30, 2015, from http://docs.opencv.org/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html

[10]Rosebrock, A. (2014, July 14). 3 ways to compare histograms using OpenCV and Python. Retrieved July 19, 2015, from http://www.pyimagesearch.com/2014/07/14/3-ways-compare-histograms-using-opencv-python/