A Fast and Robust Algorithm for the Detection of Circular Pieces in a Cyber Physical System

S.E. de Vegt

ES Reports
ISSN 1574-9517

ESR-2015-02
10 August 2015

Eindhoven University of Technology
Department of Electrical Engineering
Electronic Systems

http://www.es.ele.tue.nl/esreports
esreports@es.ele.tue.nl

Eindhoven University of Technology
Department of Electrical Engineering
Electronic Systems
PO Box 513
NL-5600 MB Eindhoven
The Netherlands

A Fast and Robust Algorithm for the Detection of Circular Pieces in a Cyber Physical System

S.E. de Vegt

***Abstract*— This paper presents an algorithm for the detection of circular pieces in a Cyber Physical System xCPS (eXplore Cyber Physical Systems). The algorithm is specific to this system, but the presented steps could easily be modified to fit other systems as well. Graphical input is split into multiple parts, based on which type of transport mechanism is shown in that particular part of the frame. The algorithm uses different strategies to detect the pieces for different parts of the image. These strategies involve grayscaling, edge detection and circle detection or circle localization. The algorithm is implemented in Matlab and is capable of processing 1 frame per second for a frame size of 720×960, benchmarked on a Intel Core i7-2670QM @2.20GHz without external GPU acceleration. The detection rate is 93.9% true positive.**

***Index Terms*— Image edge detection, Image processing, Object recognition, Image decomposition.**

I. INTRODUCTION

THIS paper presents an algorithm which will be used to process data from a camera sensor. The presented algorithm is developed because there was a need for an algorithm that could detect circular pieces on a Cyber Physical System (CPS). This setup consists of several actuators, each with their own sensors. This algorithm would allow the replacement of the current sensors on the machine by one camera sensor. The image from the sensor will be processed by the algorithm and the location and color of the detected pieces will be used by the system. The direct implementations of existing circle detection algorithms to find the circular pieces are not effective because these algorithms tend to detect other curved edges in the image incorrectly as pieces. The target system contains 4 transport bands and a rotating buffer. A camera is located above the system, centered right above the circular buffer. One frame of the captured video is shown in Fig. 1, the pieces are of a circular shape and they are available in three different colors: red, black and grey. Low contrast is a challenging part of the problem, the black pieces have a similar color to the transport bands and the grey pieces show very little contrast to the aluminum parts of the system such as the rotating buffer. Another challenging factor is the complexity of each frame, many objects could be detect as a circle. Furthermore, lighting is also not homogeneous, and reflections are present as well. The goal of this research project was to create an algorithm with an as high as possible correct-detection rate while the false detections stay as low as

S.E. de Vegt, was Electrical Engineering student at Eindhoven University of Technology, Eindhoven, The Netherlands.

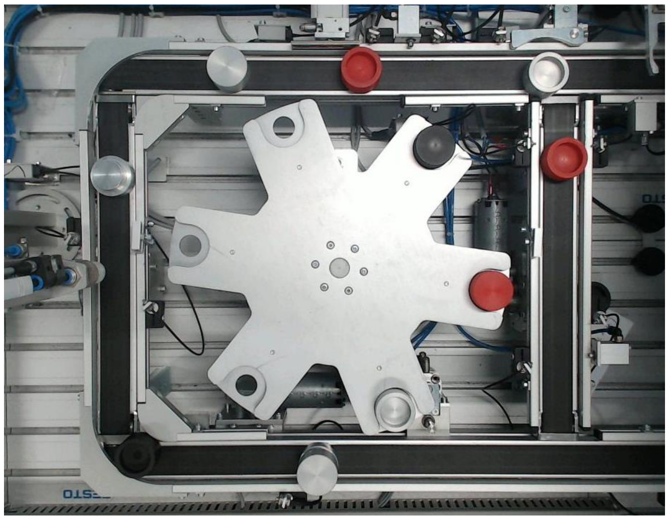
Fig. 1. An overview of the setup for which this algorithm is designed, note the four transport bands and the rotating buffer in the center.

possible. Matlab is chosen for the implementation of this algorithm. The algorithm divides each frame into multiple smaller frames in such a way that each of the created frames can be processed differently. The implementation uses earlier developed algorithms such as Canny edge detection [1] and a circular Hough transform [2]. This paper is organized as follows: the next section elaborates on related work. Section III explains the developed algorithm in detail. Section IV elaborates on the results of the algorithm. Section V proposes future improvements and the last section concludes this paper.

II. Related Work

A great deal of algorithms already exist on the topic of circle detection, edge detection and object tracking. There are sufficient implementations for circle detection algorithms, for example the widely used Hough transform [2], of which a lot of improvements are presented [3]. These two algorithms are implemented in the Matlab function `imfindcircles` [4]. Other circle detection algorithms such as a randomized circle detection algorithm [5] can detect partially covered pieces as well. The build-in Matlab function `imfindcircles` is used in this paper. Edge detection algorithms of sufficient quality do already exist. This paper's implementation uses both a Canny edge detection algorithm [1] and a Sobel edge detection algorithm [6]. Canny provides more continuous edges which is ideal for the detection of circular pieces, while Sobel on the other hand is a faster algorithm but less accurate [6]. Both algorithms are already implemented in Matlab.

III. The Algorithm

The basic idea behind the algorithm is that the frame captured by one camera consists of several sections, for example a transport band and a rotating buffer. For each section there exists a different solution for detecting the circular objects, that leads to a high detection quality for that particular section. As shown in Fig. 2, the first step of this algorithm is to split the image into different parts, based on spatial constraints, this division is shown graphically in Fig. 3. This process is a hard-coded combination of bitmasking and cropping the frame, this step has a few benefits. First of all, the image processing algorithms can be specialized for each section. There is no need to settle for a less efficient way of solving one particular part in favor of another (computationally bigger) part. Because the algorithm is split into different methods which are each related to physical areas in the system, it also becomes easier to update the algorithm when a part of the system changes since only the method corresponding to that particular area has to be changed. Another big upside to this modular way of solving is that it becomes feasible to parallelize this problem. In this case, one of the two methods could be delegated to another thread or even another CPU. With an efficient implementation and sufficient task size, a parallel implementation can already outperform a single-threaded implementation with two or more tasks and a CPU with two or more cores. The specific implementation for this setup splits each input image into two sub-problems, as can be seen in Fig. 2. The first sub-problem is to detect for each possible location of a piece in the rotating buffer and if that location is empty or if there is a piece. When a piece is detected, also the color should be determined. The second sub-problem is to detect the location and color of every piece on the transport bands. These two sub-problems are solved in different ways. Subsection A describes Method 1, a solution to sub-problem 1 in area 1 and subsection B describes Method 2 which is the implemented solution for sub-problem 2 in area 2.

A. Method 1

Method 1 is used to solve the problem in Area 1, the area with the rotating buffer. Using a circular Hough transform in this region led to many erroneous detections, mainly because of the other circular shapes present in that particular area. Also the empty buffer locations will not always be detected in this way. However, circle detection is not needed to determine the possible locations of the pieces if the rotation of the buffer is known. Determining the rotation of the device is easiest done by looking at the screws in the center. Because the buffer is point symmetric in the center with 3 symmetry axes, only angles between 0° and 60° are interesting. Unlike the buffer, the screws can be detected very easily by using a circle detection algorithm and a mask. The area around the screws is empty and the edges of the screws show sharp contrast with respect to the background. The buffer-locations have to be entered manually for one 'setup-frame'. From the locations of the screws in the 'setup-frame' and the locations of the screws in a random other frame an affine matrix can be created which transforms the known set into the newly found set [7].

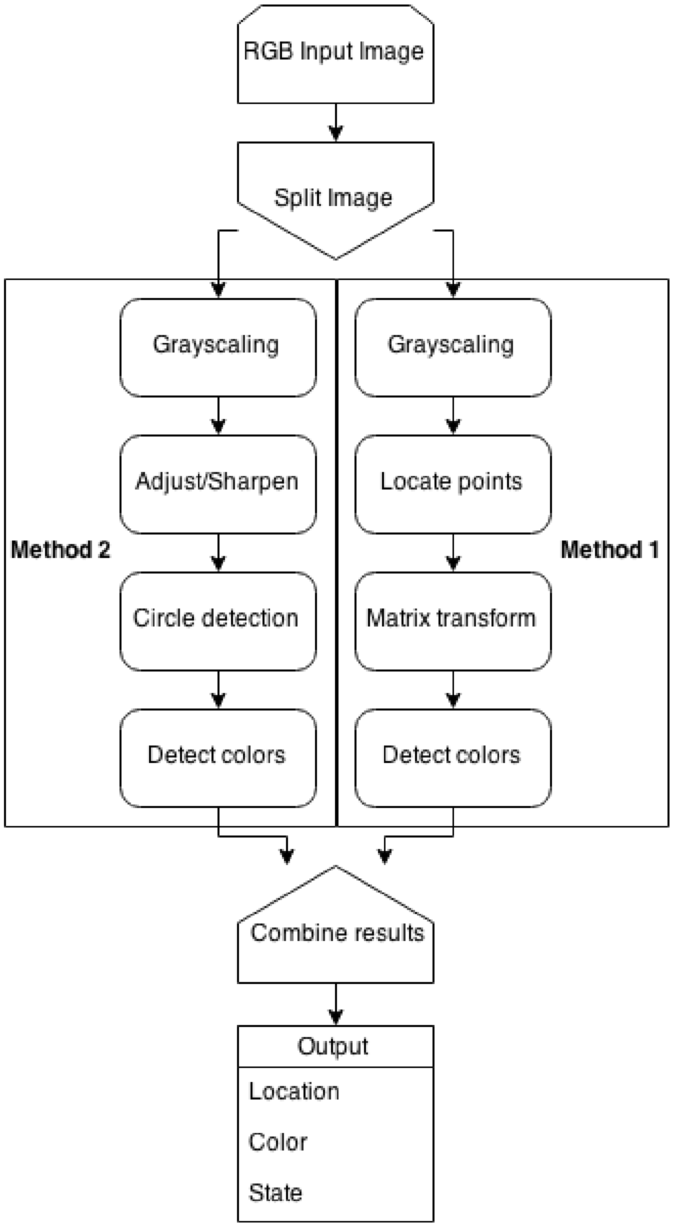

Fig. 2. A systematic overview of the implementation of the presented algorithm

Because there is no translation and scaling involved but only rotation, the same matrix can be used to transform our known buffer locations to the buffer locations in the new frame. These steps are visualized in Fig. 4. Once these new locations are known, it is necessary to determine if a location is empty and determine the color of the detected pieces. For the algorithm, the only difference between an empty location or a location with a piece is the visible hole on the buffer locations which is only visible at empty spots (see Fig 1.). Detecting this hole can be done by using a circle detection algorithm. For each of the pieces on the buffer the color has to be detected. This color is determined by averaging the color over the area in the center of the buffer-location. A piece is considered red if the difference between the red color component and the other two color components is sufficiently high, a piece is

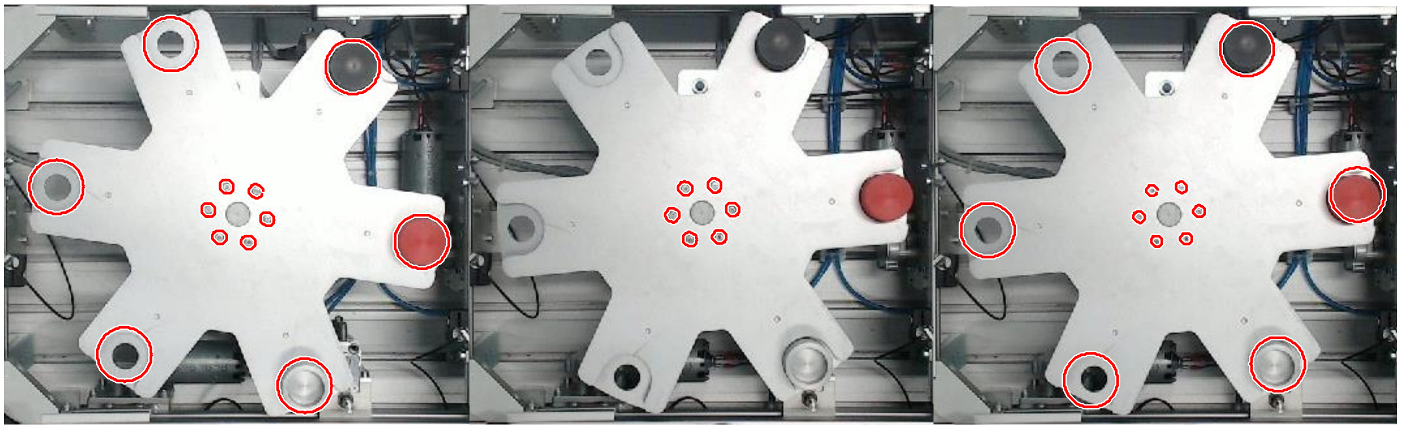

Fig. 4. Several steps of Method 1, from left to right respectively: Manual calibration, detect the angle of rotation by finding screws in the center, locate outer points by using angle and matrix multiplication.

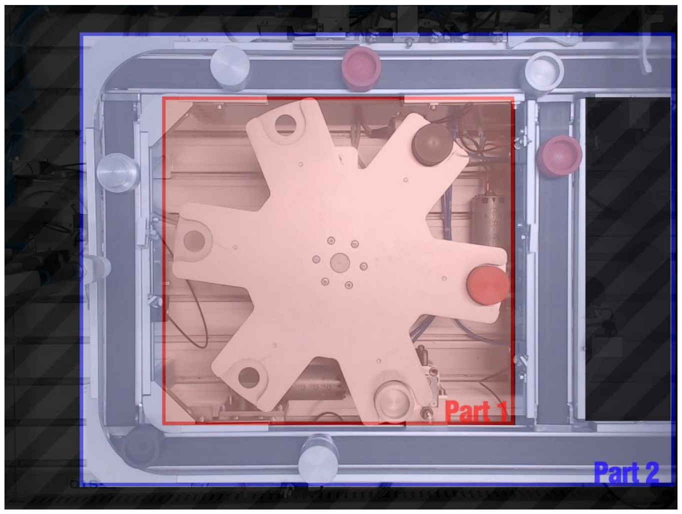

Fig. 3. One frame split up into 2 parts. The blacked out area is bitmasked away by the algorithm, to avoid confusion by the circle detection algorithm. Part 1 corresponds to Area 1 and is solved by Method 1, Part 2 corresponds to Area 2 and is solved by Method 2.

considered black if the average of the color components is less than a certain value. All the other pieces are considered grey.

B. *Method 2*

Method 2 is used to detect the pieces on the transport bands. In this case the positions of the individual pieces are not related to each other, nor to the location of other physical parts of the system. Some sort of object detection or in this case, circle detection, has to be used to find the pieces in the image. However, because of the poor contrast between the black pieces and the black transport bands, some preprocessing has to be done before `imfindcircles` is able to detect the pieces. The first option was to increase the sensitivity of `imfindcircles`. This parameter sets a threshold for the minimum quality of the detected circles. Increasing the sensitivity of this function will lead to a lot false positive results so this is not an option. The second option was to look at static background subtraction [8]. However, this did not increase the detectability of the black pieces. Therefore, the final solution was to use contrast stretching, an image adjustment technique to increase the contrast in the darker regions [9]. This is done by clipping brightness values up to a certain threshold and linearly rescaling all the values afterwards. This means that the values in the very bright regions are clipped while increasing the contrast in the darker areas of the image. This improved the contrast enough to use further process the image. The next step consists of an canny edge detector [1] followed by a circular Hough transform algorithm [2]. This results in the locations of the pieces. Color is again determined by thresholding the obtained color values of the found locations.

IV. Results

Table 1 shows the results of the current implementation, area 1 and area 2 combined, on a prerecorded video stream. The correct-detection rate of the algorithm is at 93.9%, even though the lighting was not homogeneous during the test, which resulted in an increased amount of erroneous detections. This accuracy is reached by carrying out similar tests on different video's and tweaking the sensitivity and making other small changes in the algorithms implementation. For example, the way of labeling a red piece was a result from tests on video data, which showed that thresholding only was not sufficient. Investigating on the errors in a wider variety of test samples will expose other weak spots which can then be fixed to improve the detection rate and decrease the error rate. It is remarkable that the errors almost exclusively occur in method 2, which solves the problem with the transport bands. The only errors occurred in method 1 were in the three frames marked as ‘bad frames’. ‘Bad frames’ are frames in which the detection of the screws went wrong inside the `imfindcircles` function, due to the sensitivity being too low. In such a case less than 6 screws are detected and this results in a corrupted affine transformation matrix and thus, erroneous locations for the buffer. Using a higher sensitivity and selecting the 6 best circles can resolve this problem. Other than this error in these three frames, method 2 showed no errors in our test sequence. A large amount of the remaining errors were reoccurring problems for a couple of consecutive frames. The first 40 frames do not show these persistent erroneous detections, the detection rate in those frames is 97%, and the correct-framerate, the percentage of frames

Algorithm results				
Total Frames	**100**	**Total Detectables**	**1233**	**100 %**
Correct	51	Correct	1158	93.9 %
1 Error	29	Wrong state	24	1.9 %
2 Errors	12	Wrong location	18	1.5 %
3 Errors	5	Undetected	33	2.6 %
Bad frames	3	Double detections	13	1.1 %

Table 1. Results of the implemented algorithm, performed on a prerecorded video stream with a resolution of 720×960. Detectables consist of all the visible pieces and all six buffer locations (both empty and non-empty).

Method 1		vs Method 2	
Total Detectables	**600**	**Total Detectables**	**633**
Correct	582	Correct	576
Wrong state	0	Wrong state	24
Wrong location	18	Wrong location	0
Undetected	0	Undetected	33
Double detections	0	Double detections	13

Table 2. Results of the specific parts of the implemented algorithm, ran on a prerecorded video with a resolution of 720×960. Detectables consist of all the visible pieces and all six buffer locations (both empty and non-empty).

without a single error, increased to 58%. This is because when the frame number increases, the pieces collectively move to the bottom right of the setup, where the brightness is less then on the top side of the setup. This is the reason for the increasing color-detection errors. Creating homogeneous lighting will lead to better results. Also 28 of the 33 cases of a piece not being detected happened in the last 60 frames. Thus fixing the brightness will also lead to an improved detection rate. Table 2 shows the results of the two separated methods. From this can be concluded that both methods struggle with different issues. Method 1 only has errors when the bad frames happen, so when the screws are not detected correctly. Because Method 1 handles the center area of the image, where the illumination is quite homogenous, there is no problem detecting the colors. This is also due to the fact that Method 1 has a more robust color-detection method, it averages the color over an area while Method 2 uses only the center point. This is to counteract that pieces in area 2 can move outside the camera's view which would result in averaging over an area which is partly outside of the frame.

Some concessions had to be done on performance. The current speed of the algorithm is in average 1 frame per second. This results in problems with 'real-time' tasks because the real locations of the pieces may differ already from the location determined by the algorithm. The processed video was recorded with 10 fps. Fig. 5 shows the first 12 frames. When the image is used in real-time environments, this means the first frame of this video would be processed and the next nine frames will be 'skipped' before another frame will be processed. So the starting point and endpoint of the movement of a single piece in Fig 5 almost show how much the real and processed location could differ in a worst case scenario. Estimating the possible movement of a piece in the processing time of one frame results in a distance of around 15cm. However, a controller will know this and can anticipate on the future the movements of its own actuators.

V. Future Improvements

Further improvements to the algorithm need to be performed. Besides the already mentioned improvements, there are a few more advanced ideas to improve the current state of this algorithm which will be presented in this section.

A. Overhead Protocol

The current implementation has a low false positive detection rate. However the algorithm is unlikely to ever become 100% accurate. Therefore, a future improvement may be to add an extra algorithmic layer on top of the current implementation. This layer could implement error correction to filter out the incidental erroneous detections. For example averaging the color of a piece over a couple of frame will result in neglecting one incidental erroneous detection. This will work since the algorithm does not show a lot of persistent errors when there are no problems with illumination, most of the errors occur just for one frame. Also 'bad frames' as introduced in section IV can be detected most of the times by checking if the detected locations are on a circle because the buffer is circular. If the locations are not on a circle, the frame can be dropped. Dropping such a frame might be a better solution then continuing and reporting faulty data.

B. Parallelization

The current implementation of this algorithm is single threaded, even though it can be easily parallelized as proposed in section III. A small attempt has already been made but in Matlab (version R2014a) this did not result in a measurable speedup, however, this is very possibly due to the very poor Matlab out of the box multi-threaded scaling. However, a more advanced Matlab implementation, or an implementation in a more multi-thread friendly language such as C++, C# and many others, will result in a significant speedup, if the parallel tasks are large enough to overcome the overhead introduced by the parallelizing.

C. Preprocessing

Only simple preprocessing is done at the moment, however this could increase the detection rate. For example the already mentioned background subtraction could be implemented again. Even though it did not help for the detection of the black pieces, it can increase the correct detection rate of the other colors. Also other preprocessing steps such as nonlinear contrast adjustments with the focus on the colors of the pieces or better sharpening of the frames could lead to better results with `imfindcircles`. Furthermore, improved preprocessing could be used to counter problems with the luminosity as stated in Section IV.

D. Tracking

In order to reduce the workload, tracking algorithms can be implemented. In this situation the possible directions of movement of the pieces are known, good tracking algorithms can reduce the area in which circles have to be detected, and thus reduce the computational workload. A good tracking algorithm for this scenario which also is able to keep tracking when occlusion occurs would be the algorithm presented by Wang, Ma Wang and Liu [10]. The current implementation of the developed algorithm is not capable of handling occluding pieces, while the system is built to also stack pieces on top of each other, which makes it interesting to think about implementing such a tracking algorithm which is able to track the pieces on locations where circle detection fails.

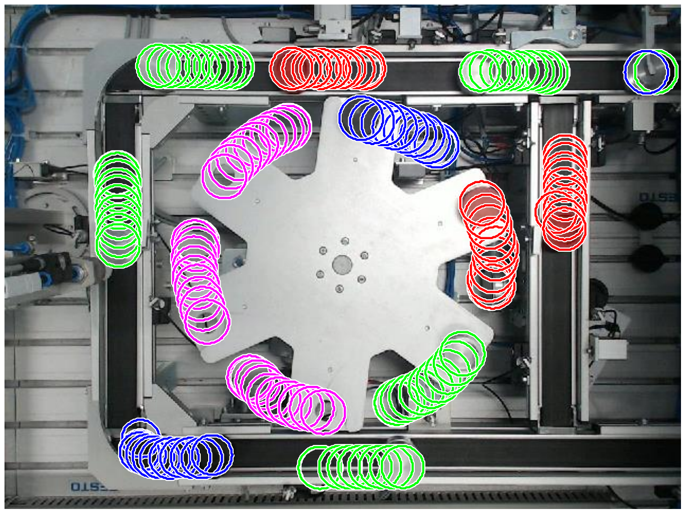

Fig. 5. The algorithm working on the prerecorded video stream, currently tracking 14 pieces. The tracks shown are from the first 12 frames of the video. The circles have the color corresponding to the piece (not visible).

VI. Conclusion

In this paper an algorithm for detecting circular pieces in a Cyber Physical System has been presented. This algorithm splits one task in to multiple tasks (area 1 and area 2) which can be solved individually and parallel to each other. The overall correct-detection rate of this implementation is 93.9% (See Table 1). Area 1 performs error-free for 97% of the frames, however, the remaining 3% of the frames show completely wrong detections in area 1. The detection rate in area 2 is 90.9%.The error rate of area 2 can be halved by creating homogeneous lighting across the system. Also future work is proposed, which shows that the algorithm is not fully-developed and performance can thus be improved.

VII. References

[1] J. Canny, "A computational approach to edge detection.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.

[2] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures.," *Mag. Commun. ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[3] T. J. Atherton and D. J. Kerbyson, "Size invariant circle detection," vol. 17, no. February 1997, pp. 795–803, 1999.

[4] "Find circles using circular Hough transform - MATLAB imfindcircles - MathWorks Benelux." [Online]. Available: http://nl.mathworks.com/help/images/ref/imfindcircles.html. [Accessed: 28-May-2015].

[5] L. Q. Jia, C. Z. Peng, H. M. Liu, and Z. H. Wang, "A fast randomized circle detection algorithm," *Proc. - 4th Int. Congr. Image Signal Process. CISP 2011*, vol. 2, pp. 820–823, 2011.

[6] R. Maini and H. Aggarwal, "Study and comparison of various image edge detection techniques," *Int. J. Image Process. …*, vol. 147002, no. 3, pp. 1–12, 2009.

[7] D. C. Lay, *Linear Algebra and Its Applications*, 4th ed. 2012.

[8] S. Zdonik, P. Ning, S. Shekhar, J. Katz, and X. Wu, *Moving Object Detection Using Background Subtraction*. 2014.

[9] "Linear Contrast Enhancement." [Online]. Available: http://www.r-s-c-c.org/node/240. [Accessed: 06-Jun-2015].

[10] J. Wang, Y. Ma, C. Li, H. Wang, and J. Liu, "Multi-object tracking with explicit reasoning about occlusion," *Proc. 2009 Int. Jt. Conf. Comput. Sci. Optim. CSO 2009*, vol. 2, pp. 325–327, 2009.