Real-Time Translational and Rotational Motion Tracking of Large Microrobot Swarms with 250+ Agents*

Zhijian Hao and Ressa Reneth Sarreal

***Abstract*—Because of the limited sensing and communication capabilities of micro-scale robots, tracking by visual inspections is crucial for monitoring, analyzing, and controlling swarm behaviors. In this paper, a custom-built light-weight computer vision program is presented to track both translational and rotational motion of a large microrobot swarm with more than 250 agents. Multiple techniques, including color contrast filtering, template matching, principal component analysis (PCA) and correspondence by proximity, have been integrated to extract spatial locations and orientations of all robots in the swarm, while maintaining a unique identification (ID) of each robot. Using spatial sampling techniques, the program is optimized for real-time application at up to 5 frames per second (fps) without significant decrease in accuracy.**

***Index Terms*—Multiple-object tracking, computer vision, microrobot swarm.**

I. Introduction

MICRO-SCALE robots has been gaining interests recently due to their scalability and usage in confined spaces and in-vivo applications [1], [2], [3]. However, due to the size limit and thus limited sensing and communication capabilities, tracking and analyzing behaviors of both individual robots and swarms of robots are challenging. Visual inspection naturally became a preferred solution to analyze micro-scale robots. Still, in cases of large robot swarms, real-time tracking and analysis generally require large computation power.

In this paper, we present a custom-built light-weight computer vision program using multiple image processing techniques for visually tracking translational and rotational motions of a large microrobot swarm with 250+ agents (Fig. 1) in real time at up to 5 frames per second (FPS).

The robots (green) move freely in the 2-D circular arena (white) and exhibit Brownian motions due to collisions with each other and inherent transnational and rotational motion randomnesses. Each robot also has an red line as the direction indicator, whose detection will provide quantitative information for characterizing individual robot's motion and the collision behavior between robots.

For each frame, the locations of all 250+ robots were determined, and the orientation of each robot were extracted as indicated by the red lines atop as well.

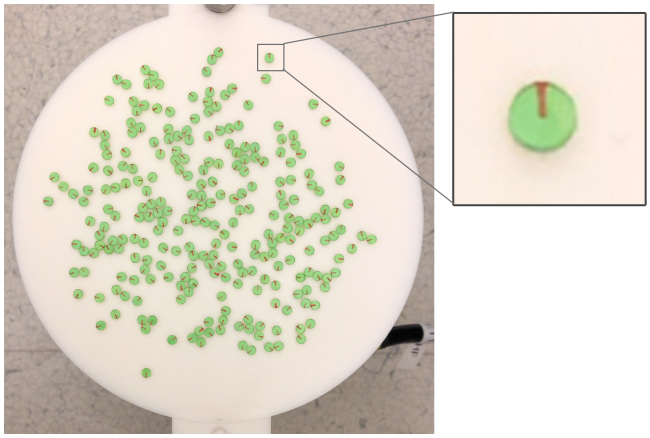

Fig. 1. Microrobot swarm with 250+ agents used for detection.

The correspondence of robots was maintained in real time such that each detected robot was associated with a unique ID based on location and motion constraints between frames.

Methods and techniques include a combination of color contrast filtering, template matching, principal component analysis (PCA), correspondence by proximity, and temporal and spatial video sampling.

Accuracy and computation speed were used as the core performance metrics to evaluate the custom-built program's individual components (position and angle detection) against same functionalities implemented using standard packages (e.g. OpenCV) and manually extracted ground-truth values.

Mean squared errors (MSE) were calculated and used to evaluate the correctness of robot IDs and analyze the effects of video down-sampling.

II. Methods

Object tracking is frequently performed in the fields of computer vision and image processing. Multiple-object tracking requires an additional level of computation as each object must be identified for maintaining individual trajectories [4]. Common approaches and tools used for tracking multiple objects include machine learning classification algorithms, optical flow [5] and Fourier transforms. Other methods include deep learning tools, such as You Only Look Once (YOLO) [6]. These approaches are general-purposed but are more computationally heavy, which is not ideal for real-time tracking on a swarm of identical robots. Ho and Goecke at Adelaide University have developed a method of optical flow

*This paper includes research contents that have not been published. Please do not distribute.

Both authors are with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332 USA [zhao38,rsarreal3]@gatech.edu

using Fourier Mellin transforms that not only capture pure translation, but also capture object rotational motion of image patches [7]; however, the approach will also be difficult to implement in real time with hundreds of robots to analyze per frame.

A common technique for object detection is template matching where a generalized template of some predetermined object is used to find similar objects captured within a larger, more complex image. Template matching has been used for tracking functionality in surveillance systems [8] and aerial vehicle tracking [9] and has been modified to operate on various applications. In the example of the surveillance system, dynamic template matching [10] is used because the target object may change its appearance throughout the time it is being tracked. Additionally, modifications to the algorithm have been made to improve run times in high-speed footage as in the case of tracking aerial vehicles. Modifications include limiting the search area to only blocks that are near the previous location of the identified object.

Template matching proves as a useful tool for identifying objects, and as the target of this project is to identify objects that maintain a uniform appearance, multiple objects will be found with a single template. Dynamic template matching will not be necessary to include; in addition, the described localized search template matching will likely not perform well as multiple identical robots will be surrounding each other.

This project will utilize combination of compression techniques, application-based filters and transforms to efficiently identify the robots and the mentioned features. In addition, some machine learning techniques, such as principal component analysis (PCA) [11] has been used for visual object tracking [12] in complex settings such as dance floors and can be adapted and utilized for robot feature extraction. Combining and tuning the mentioned methods to the application would ideally provide a potential solution for identifying and tracking transnational and rotational motions of micro robots.

III. Implementation

The whole program is implemented in Python. OpenCV and DippyKit packages are used for video/image processing. Math, NumPy and Scikit-learn packages are used for data processing. JSON package is used for data storage. Time package is used for run-time estimation.

The overall flow of the program is:

1) Read template image.
2) Read a new frame from the video and down-sample the image based on user input.
3) Color contrast filtering with user inputs to extract both the green bodies and red orientation indicators of the robots.
4) Run staged template matching algorithm over the entire frame.
5) For each detection, crop out the robot image, and run principal component analysis (PCA) based angle detection algorithm.
6) Assign unique IDs to each detection based on Manhattan distance minimization.
7) Save data.
8) Repeat step 2-7 until all frames of the video are analyzed.

Due to the complexity of the program, implementations of individual components are detailed separately together with the corresponding results and analysis. The overall results, analysis and conclusion regarding the whole program are discussed afterwards.

IV. Frame Segmentation

To minimize the effects of the background and possible shadows, the image is **double thresholded** before robot detection. To determine the best threshold combination and the best color space to perform thresholding in, the spectra of the whole image and the foreground (i.e. the robots) are analyzed.

In Fig. 2, both the RGB and HSV spectra are shown, where the not filled lines correspond to the raw input (Fig. 2c) and the filled areas correspond to ideally filtered images (Fig. 2d) (produced by manual masking). Qualitatively speaking, in the RGB color-space, there is no differentiation between robot pixels and background pixels (as indicated by the gap between the not filled lines and the filled areas in Fig. 2a). However, when the image is converted into **HSV color space**, it is clear that the robots and the background can be differentiated by using the saturation channel. This can be seen as most robot pixels are in the saturation range above 74, while the background pixels are below this range.

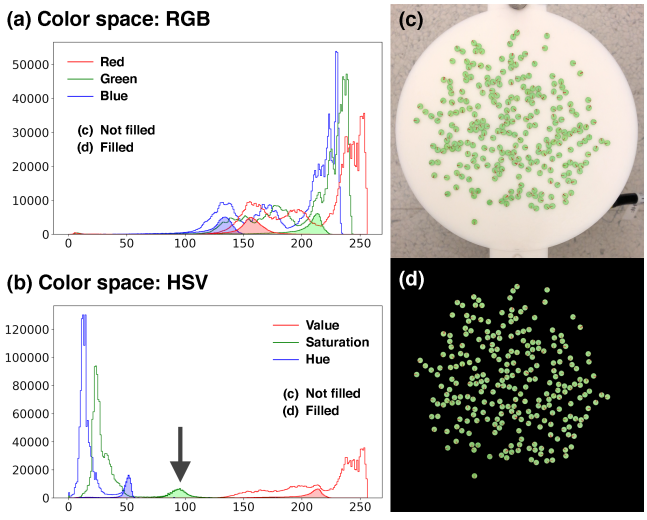

Fig. 2. Image spectrum of raw and ideally filtered images in both RGB and HSV color spaces.

Numerically, the thresholds can be determined by **maximizing information gain (IG)** of a double sided splitting. The equation for IG calculation is:

$$IG = H - (H_1 \times P_1 + H_1 \times P_1) \tag{1}$$

Where H is the entropy of the original image, H_1 and H_2 are the entropy of the two classes after splitting, and P_1 and P_2 are the probabilities of a random input falls into each class.

The resultant information gains given different combinations of the lower and upper thresholds are shown in Fig. 3.

Fig. 3. Information gain with varying double-sided thresholds.

The two rectangular areas with high IGs are associated with the background and the foreground, respectively. Since the foreground is of interest, we then focused on the lower rectangular area. It is shown that IG is maximum when lower threshold = 74 and upper threshold = 255. This matches earlier discussion and [74, 255] is the best threshold for frame segmentation.

V. Robot Localization

Contour detection was originally attempted to localize properly filtered robots; however, segmentation of dense robot clusters was a challenge. As shown in Fig. 4a, when robots are in close contact, they cast similar colored shadow onto the substrate, which cannot be distinguished from the robots.

This issue is especially severe when three or more robots have completely merged together. If part of the background is detected in the shadow area, morphological dilation and erosion operations can be used to separate the merged robots. Or, when only two robots are merged, watershed algorithm [13] can be used to separate them. However, in our case, the robots dynamically form clusters and merge into one foreground (Fig. 4b). Neither morphological operations nor watershed algorithm can be used to separate them (Fig. 4c).

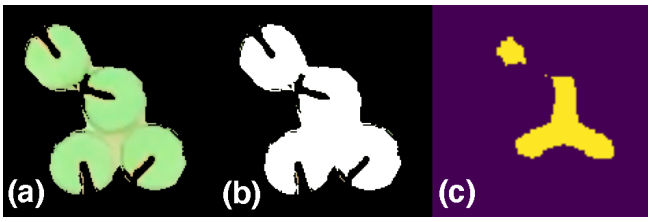

Fig. 4. Segmentation cannot be done due to shadow.

To guarantee accuracy, template matching from OpenCV package were used to detect robots across the whole frame [14]. Yet, the aforementioned segmentation issue still affects the detection in the following two ways: (1) false positive: when the threshold is set low for a fully parallel detection, the shadow area may be falsely perceived as a robot; (2) false negative: the existence of the shadow area lowers the confidence scores of the surrounding robots such that they cannot be detected.

To address these issue, a **staged template matching** method was developed. The flow of this program is illustrated in Fig. 6 and the steps taken are:

1) Initialize a high threshold (α_0).
2) Perform template matching on the original input image.
3) Detect robots from template matching results and based on the threshold. Note that because of the high threshold, not all robots are detected. The detected ones are more likely to be the robots that are not inside clusters.
4) Use the detection as a mask and remove it from the original input. Use the modified image as the new input. Note that by removing these high confidence robots, robots that are originally inside clusters can be exposed. They would have higher confidence scores during the next template matching cycle.
5) Lower the threshold by a scaling factor ($\beta < 1$), such that the new threshold is $\alpha_0\beta$.
6) Repeat step 2-5 and merge the detection results until no more robots can be detected from template matching based on the updated threshold.

Fig. 5 shows the result of robot localization using the staged template matching method where the red circles are the detected robots' boundaries. Based on visual inspection, all 251 robots are accurately detected and located.

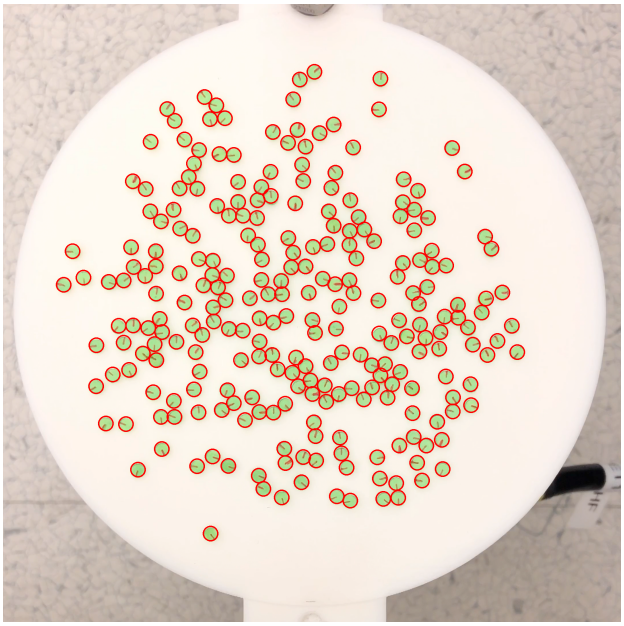

Fig. 5. Localization result: red circles are detected robots.

VI. Angle Detection

The Hough Lines Transform (HLT) [15] was used as the baseline comparison for our target-application methods. Two

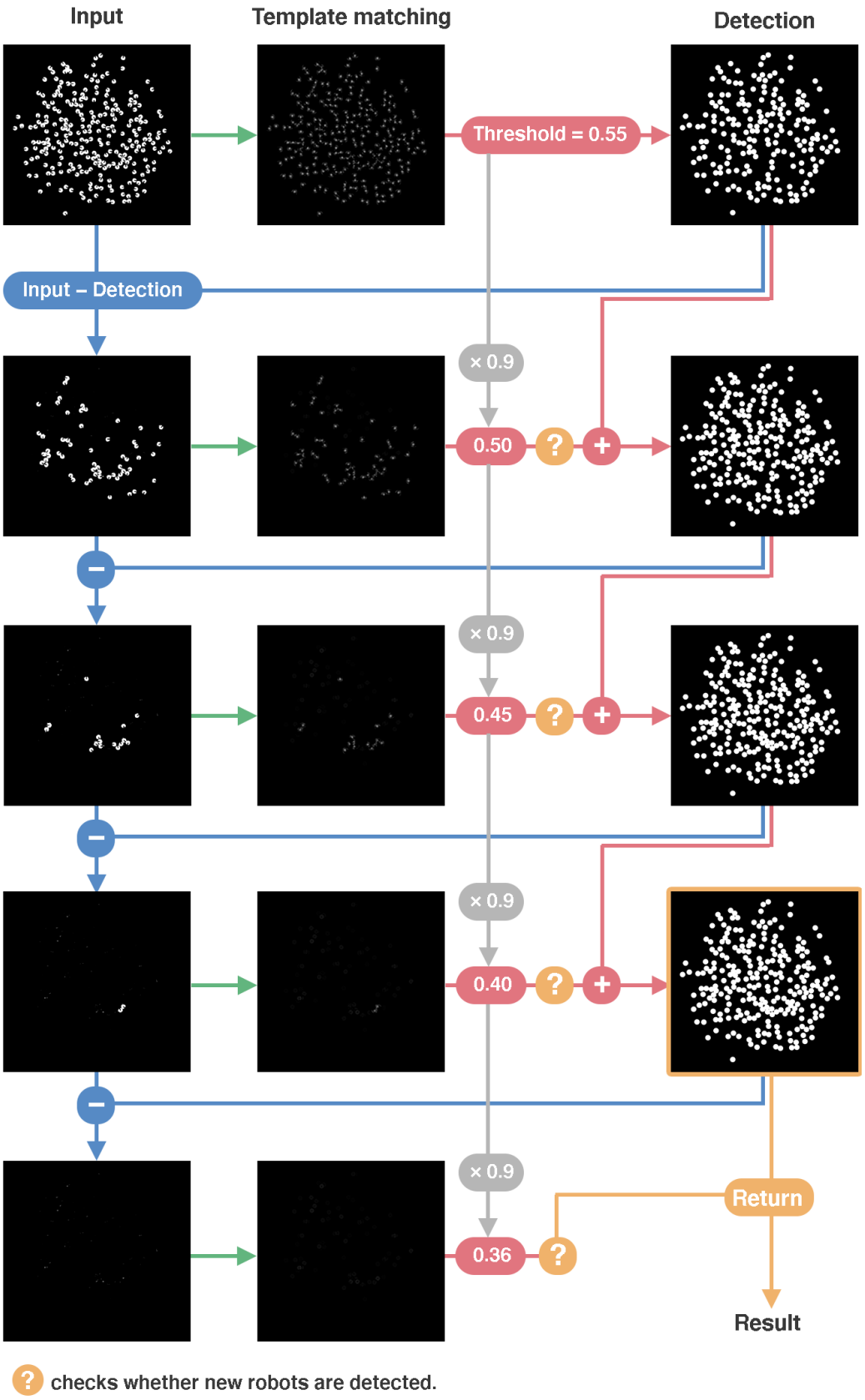

Fig. 6. Staged template matching for each frame.

primary methods were developed. First, the Manhattan Distance (MD) method [16], and second, the principal component analysis (PCA) method. The angle detection process occurs after the isolation of individual robots, shown in Figure 7. The goal in this development stage was to design a more efficient and novel approach to detecting angles than any existing method for the target application.

Fig. 7. General masking process for red line detection. (a) Original cropped image of a single robot. (b) Mask generated after filtering out non-red values within a certain HSV range. (c) Bit-wise intersection of the original image and the mask. (d) Example of orientation line detection using MD method.

The **HLT method** involves transforming the original image into a compilation of edges which, post color filtering, outline the red orientation indicator on the robot. The lines obtained from these edges are then averaged to determine the correct angle orientation of the robot.

In the case of the **MD method**, since a single robot has been isolated prior to orientation detection, the center of the robot is assumed to be at the center of the cropped image of the robot. A mask that retains only red features determines the furthest location of red pixel using the MD formula, the absolute horizontal displacement summed with the absolute vertical displacement, and a line is formed between these two points. The angle of the line is then determined as the robot orientation.

The **PCA method**, like the MD method, requires a mask that maintains only red features of the cropped image of the robot. The first major axis is determined and an angle can be calculated with the original axis. As the first major axis is determined, a point closest to the center and a point furthest from the center, $c = (x_c, y_c)$, are found: points p_1 and p_2.

$$p_1 = c - \bar{v}_1 + \lambda_1 \tag{2}$$

$$p_2 = c - \bar{v}_2 + \lambda_2 \tag{3}$$

where $\bar{v}$ and λ are the eigenvectors and eigenvalues resulting from the PCA major axis calculation, such that they reduce they represent the points that minimize the distance to the center and maximize the distance to the center. The points may also be represented in the form $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$. The bisecting point is then calculated as b:

$$b = \left(\frac{x_1 - x_2}{2}, \frac{y_1 - y_2}{2} \right) \tag{4}$$

A line may be formed between $b = (x_b, y_b)$ and the center of the image and an angle may be calculated to resemble the orientation of the robot:

$$\theta = tan^{-1} \left(\frac{y_b - y_c}{x_b - x_c} \right) \tag{5}$$

To resolve which direction is forwards, rather than backwards, two of the furthest points along the major axis are identified, and the one furthest away from the center of the image determines the forward direction.

The results of the implementation of the three methods are shown in Table II. The first attempt of creating an application-based approach, which used the MD method, did not perform as well as the standard line and shape detection HLT method; however, after approaching the problem with the PCA method, the error was reduced by approximately 3.5%, and the run time was reduced to a third of the HLT method's run time.

TABLE I
ANGLE DETECTION RESULTS

Trial Image	Actual	Hough Lines	Manhattan	PCA
1	103.448°	82.694°	104.349°	100.419°
2	322.058°	333.435°	331.557°	327.063°
3	276.254°	275.856°	277.765°	278.577°
4	229.764°	229.736°	223.831°	227.758°
5	59.036°	61.020°	55.222°	60.957°

TABLE II
ANGLE DETECTION AVERAGE ERROR AND RUN TIMES

	Hough Lines	Manhattan	PCA
Error (%)	5.453	2.682	1.890
Run Time (s)	0.0100	0.0196	0.0032

VII. CORRESPONDENCE (ROBOT ID)

A unique ID was assigned to each robot by finding the **smallest Manhattan distance** in X and Y dimensions among all detected robots between two consecutive frames.

A distance matrix was constructed using the definition:

$$D[i, j] = |X[i] - X'[j]| + |Y[i] - Y'[j]|$$

Where $D[i, j]$ is the Manhattan distance between the i-th detected robot in the previous frame and the j-th detected robot in the current frame, X and X' are the X data of the previous and current frame respectively and Y and Y' are the Y data of the previous and current frame respectively.

For each column of $D[i, j]$, the index of the minimal valve corresponds to the ID for the robot of this column and the data with the corrected ID are stored accordingly.

Note that this rudimentary correspondence algorithm relies heavily on the accuracy of template matching results. It requires a constant number of robot with no occlusion or redundancy in the detection results.

Fig. 8 shows that the algorithm can maintain unique IDs of all 251 robots in most cases when the localization result is correct. For both sparse robots and those in clustered areas where robots are close to each other, correspondence by minimal Manhattan distance is a viable solution, while the use of Manhattan distance reduces the computation when compared to Euclidean distance.

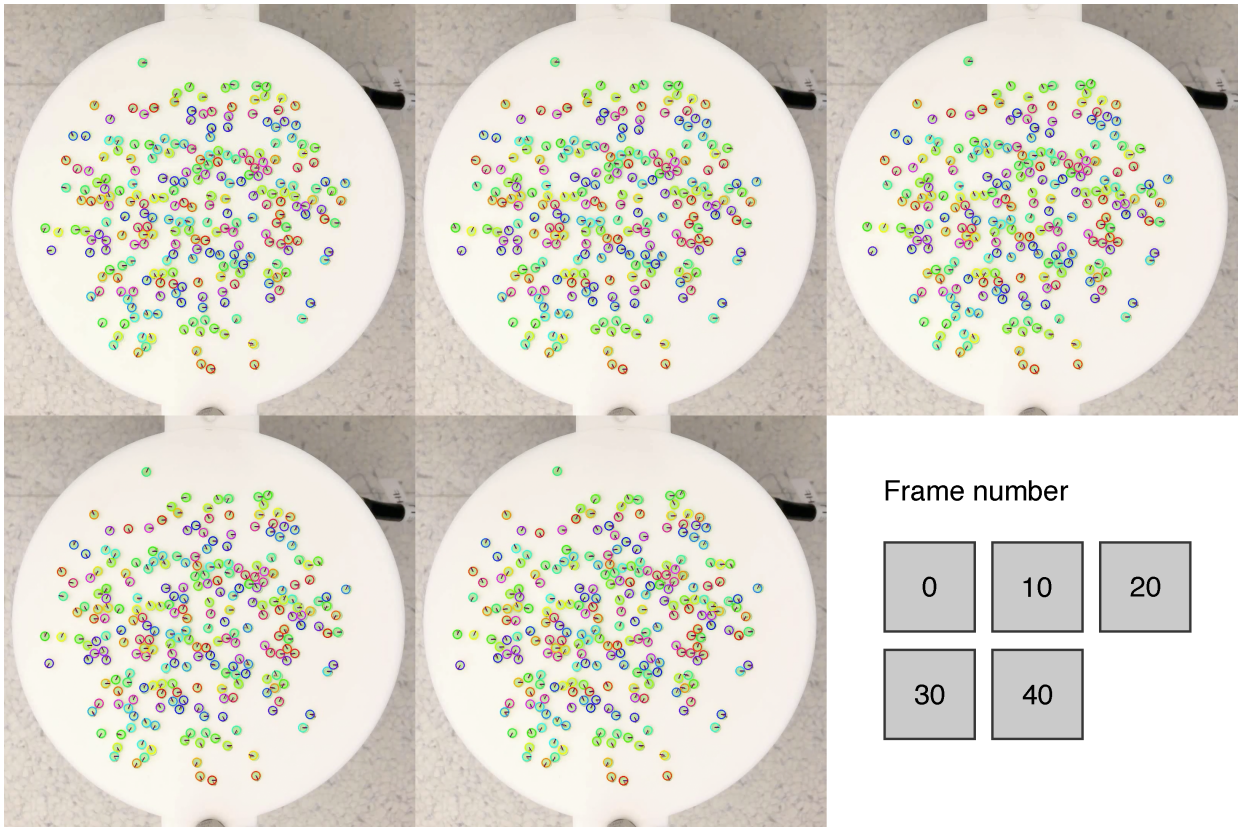

Fig. 8. Robot correspondence results from frame 0 to frame 40. Different colors of the detection circles illustrates different IDs.

VIII. Data Storage

During the execution of the program, the data is stored frame by frame. The data is stored as a **hierarchical JSON message** (Fig. 9). The three dimensions captured are time (frames), robot (IDs) and the innermost layer stores the data as an array of [timestamp, robot ID, X location, Y location, θ (angle)]. The redundancy of timestamp and robot ID makes sure corrupted and incomplete data is still usable.

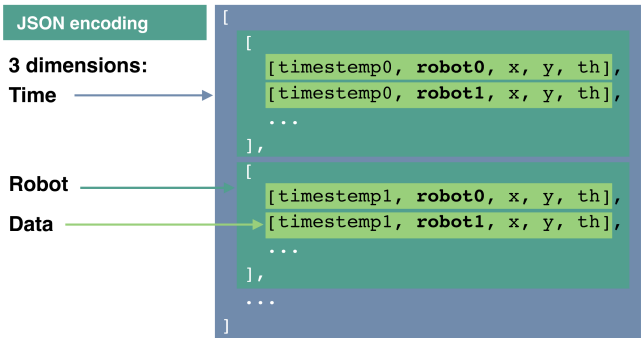

Fig. 9. Output file structure.

IX. Speed Optimization

To target real-time applications with a high frames per second (FPS), the speed needs to be optimized.

For each frame, the time complexities of template matching is:

$$\mathcal{O}(mnMN)$$

where m, n are sizes of the template, and M, N are sizes of the input image (assuming $M >> m$ and $N >> n$).

The time complexities of PCA angle detection is:

$$\mathcal{O}(R(dn_r^2 + d^3))$$

where R is the number of robots, d is the number of dimension (in our case $d = 2$), and n_r is the average number of red pixels of each robot.

All m, n, M, N, n_r can be reduced by down-sampling of the input frames. Thus, to achieve higher speed, both the original video and the template was **spatially down-sampled** with sampling period of 1-14 pixels. The down sampling results are shown in Fig. 10 and Fig. 11.

The results of run-time is shown in Fig. 12.

X. Results

The full system is composed of three main elements: robot localization (template matching), angle detection (PCA), unique robot ID (Manhattan distance minimization).

The detailed results of individual components are described in their corresponding sections. The key results are:

- Staged template matching can reliably detect the locations of all robots.
- PCA provides the fastest and most accurate way of measuring the angle.

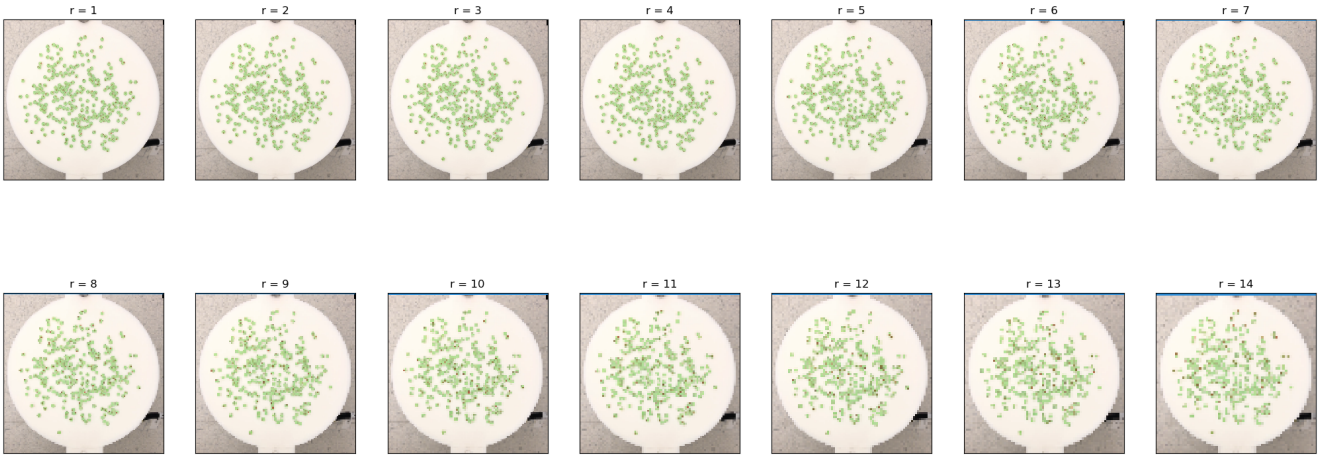

Fig. 10. The first frame of the video down-sampled at 14 different rates.

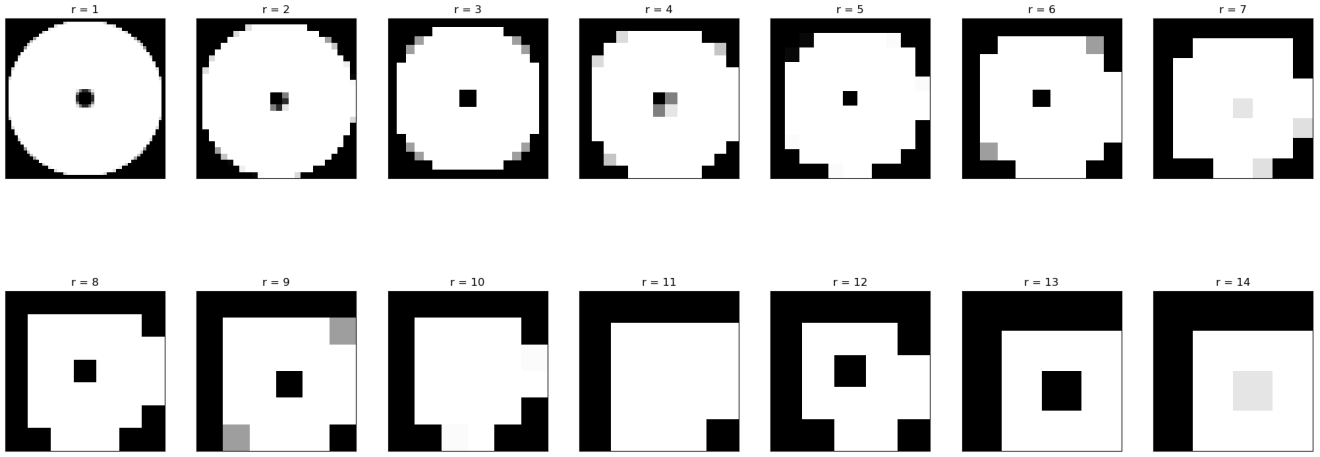

Fig. 11. The template down-sampled at 14 different rates.

- Unique IDs can be maintained for each robot by minimizing Manhattan distances of detections between frames.
- Down-sampling by a factor of 2 can shorten the run-time by 75%.

The entire program is capable to run in real time at a speed of 5 frames per second. A video is taken showing the program operating with the extracted data reconstructed and overlaid on the original input [see "251_robots_result.mp4" in Supplementary Materials].

The source code, raw input materials, and data analysis programs are included in the Supplementary Materials as well.

XI. Analysis

The detailed analysis of individual components are described in their corresponding sections. Here, the overall error analysis of the whole program is discussed.

As mentioned, by down-sampling the video and image inputs, the speed of the program can be enhanced. This enhancement, however, introduces a trade-off between speed and accuracy.

After analyzing the data, it was quickly noted that the original footage could not be down-sampled with a sampling period greater than 5 as the number of identified robots was no longer maintained, as shown in Fig. 13. This outcome may be a result of the PCA algorithm as it requires a certain number of red pixels forming a line to demonstrate confidence in the resulting robot orientation calculation. Down-sampling certainly reduced the total amount of pixels PCA could utilize for the calculation, leading to further error. The only error that existed with regard to this sampling rate was during angle detection where some angles; however, approximately 90% of the angles were detected (Fig. 15). The resulting run time after down-sampling was 25% of the original run time. Additionally, when observing the robots using this tool during some research project, angles between frames may be inferred from previous frames. This level of data automation is worth the reduction in angle accuracy. Robot IDs overall remained consistent and location; therefore, in these contexts, the program is still 100% accurate compared to the ground truth, obtained by executing these methods on the full resolution data.

Some evaluation metrics included mean squared error, num-

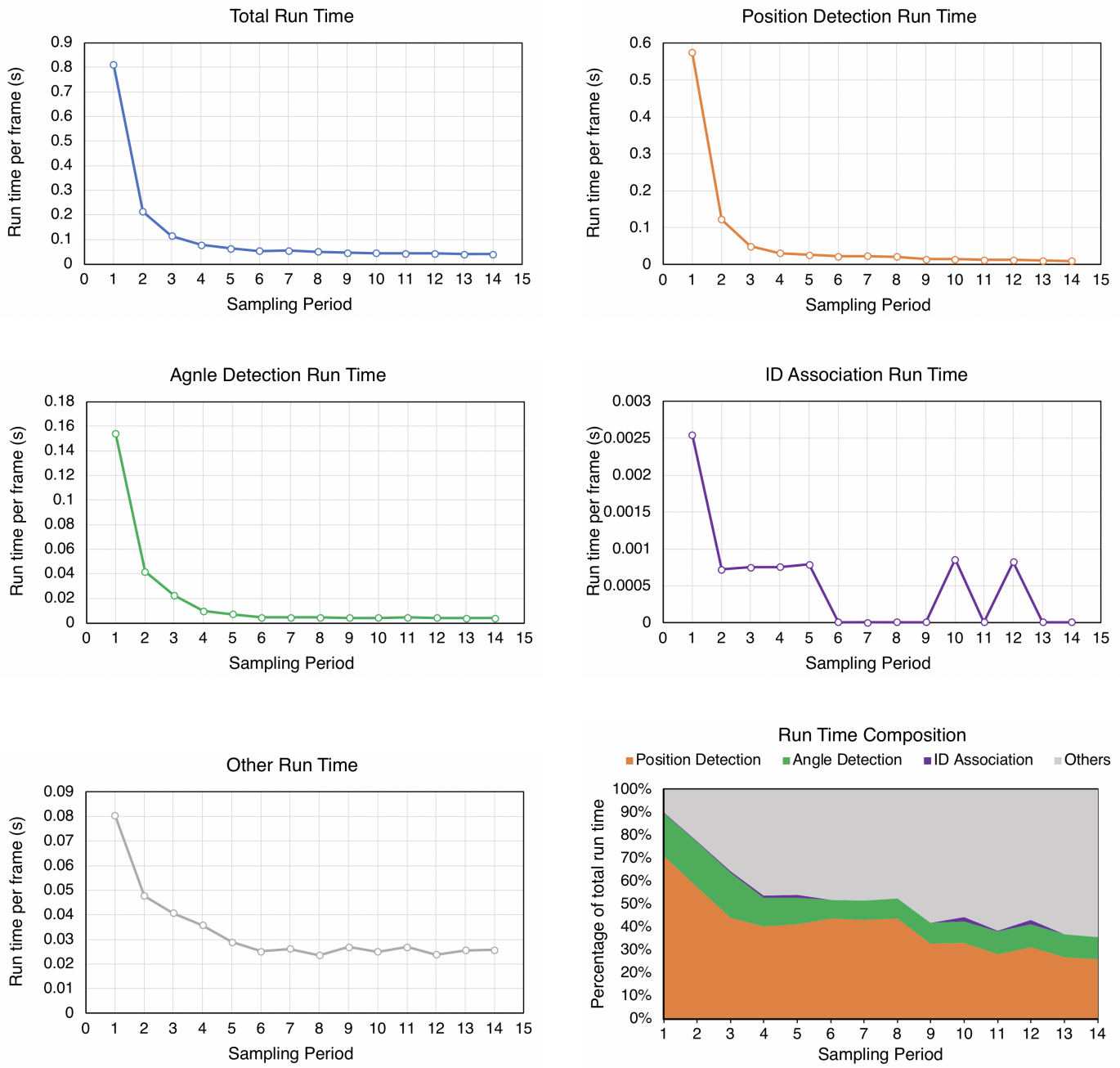

Fig. 12. Run times after down-sampling the first 100 frames and the template.

ber of angles that could not be confidently measured with PCA, number of robots detected, and run time over 100 frames.

Mean squared error between the down-sampled frames was calculated for the vertical and horizontal locations as well as angular orientation, which can be seen in Fig. 15. Each robot's x, y coordinates of location and angular orientation, θ, was determined in the original video; this data was used as the ground truth for comparison between results obtained when running the program on down-sampled videos. Generally the MSE is calculated by taking the square of the difference between each robot's original and down-sampled x, y and θ, looping over all frames and summing each of those squared differences, then averaging over the total number of frames. For θ measurements, the difference is subtracted from 360° so that the difference is always positive and between 0° and 360°. Additionally, there is a different MSE calculation performed if a bad angle is detected. The angular orientation MSE value for a bad frame becomes the max possible angle error: 180^2.

The down-sampling rate was determined based on which rate demonstrated most accurate results with the most reduced run time to approach real-time operation. It was found that after a certain sampling period (increasing the sampling period past every fifth sampled pixel), the results were not acceptable

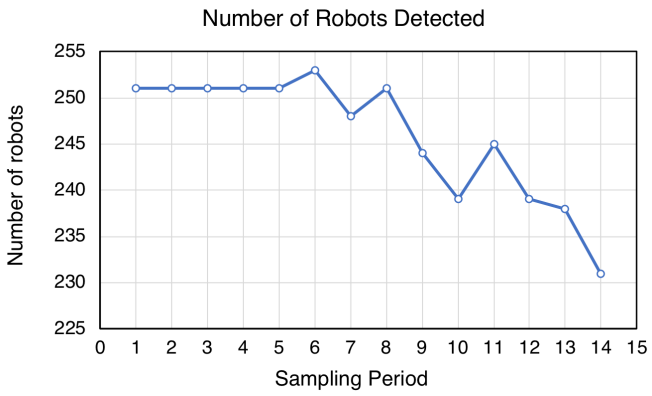

Fig. 13. Number of robots detected for 14 different down-sampling rates.

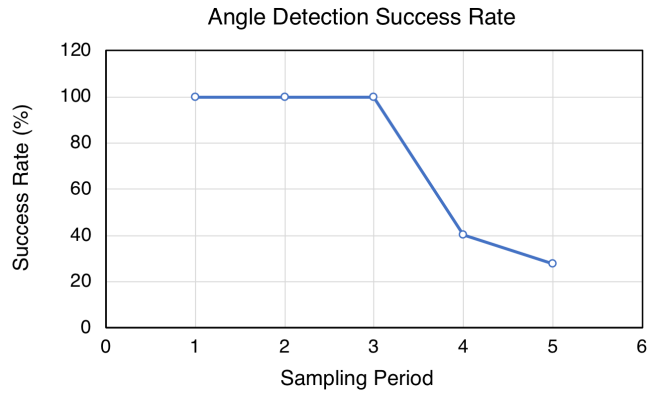

Fig. 14. Angle detection success rate for the first 5 valid down-sampling rates.

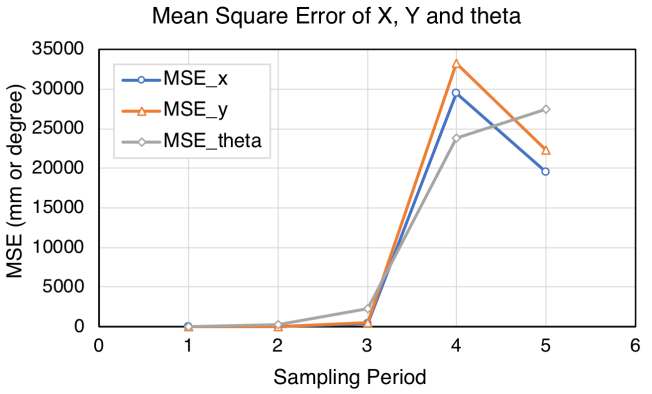

Fig. 15. Mean square errors of x, y, and theta.

as some robots were not detected at all. This error may have been caused by down-sampling the template. Another trial could be conducted with a manually constructed template of lower resolution. Upon observing run times, using a sampling period of $r = 2$ pixels demonstrated the least error while reducing the run time by 75% of the original run time.

XII. Conclusion

We presented a custom-built light-weight computer vision program, which is able to track both translational and rotational motion of a large microrobot swarm with more than 250 agents.

Multiple techniques, including color contrast filtering for image segmentation, staged template matching for robot localization, principal component analysis (PCA) for angle detection and Manhattan distance minimization for object correspondence, have been integrated to extract spatial locations and orientations of all robots in the swarm, while maintaining a unique identification (ID) of each robot.

The program is optimized for real-time application at up to 5 frames per second (FPS). Using spatial sampling techniques, the program's run time is reduced by 75% without introducing significant errors.

References

[1] R. Sahai, S. Avadhanula, R. Groff, E. Steltz, R. Wood, and R. Fearing, "Towards a 3g crawling robot through the integration of microrobot technologies," *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*

[2] M. Sitti, "Voyage of the microrobots," *Nature*, vol. 458, no. 7242, p. 1121–1122, 2009.

[3] D. Vogtmann, R. S. Pierre, and S. Bergbreiter, "A 25 mg magnetically actuated microrobot walking at ¿ 5 body lengths/sec," *2017 IEEE 30th International Conference on Micro Electro Mechanical Systems (MEMS)*, 2017.

[4] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim, "Multiple object tracking: A literature review," *Multiple Object Tracking: A Literature Review*, May 2017. [Online]. Available: https://arxiv.org/pdf/1409.7618.pdf

[5] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Computing Surveys (CSUR)*, vol. 27, no. 3, p. 433–466, 1995.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] H. T. Ho and R. Goecke, "Optical flow estimation using fourier mellin transform," *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[8] K. Gupta and A. V. Kulkarni, "Implementation of an automated single camera object tracking system using frame differencing and dynamic template matching," *Advances in Computer and Information Sciences and Engineering*, p. 245–250, 2008.

[9] S. K. Sahani, G. Adhikari, and B. Das, "A fast template matching algorithm for aerial object tracking," *2011 International Conference on Image Information Processing*, 2011.

[10] S. Yoshimura and T. Kanade, "Fast template matching based on the normalized correlation by using multiresolution eigenimages," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS94)*.

[11] Y.-S. Lee, H.-S. Koo, and C.-S. Jeong, "A straight line detection using principal component analysis," *Pattern Recognition Letters*, vol. 27, no. 14, p. 1744–1754, 2006.

[12] G.-J. Yoon, H. Hwang, and S. Yoon, "Visual object tracking using structured sparse pca-based appearance representation and online learning," *Sensors*, vol. 18, no. 10, p. 3513, 2018.

[13] A. Moga, B. Cramariuc, and M. Gabbouj, "An efficient watershed segmentation algorithm suitable for parallel implementation," *Proceedings., International Conference on Image Processing*, 1995.

[14] "Template matching." [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

[15] A. Manzanera, T. P. Nguyen, and X. Xu, "Line and circle detection using dense one-to-one hough transforms on greyscale images," *EURASIP Journal on Image and Video Processing*, vol. 2016, no. 1, 2016.

[16] "Manhattan distance," *Manhattan distance*. [Online]. Available: https://xlinux.nist.gov/dads/HTML/manhattanDistance.html