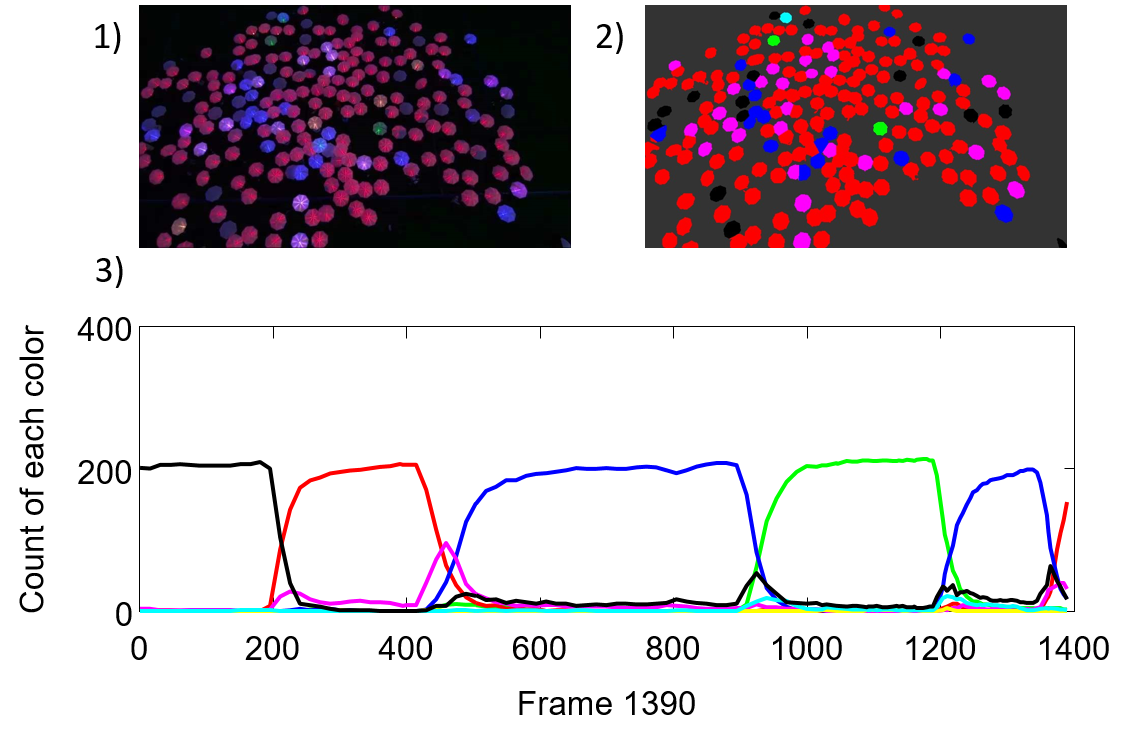
­­­­­

Wei Yao, Aaron T. Becker, [others/]

**Metrics on Crowd Control with Overhead Video and Vocal Commands**



*Abstract*— This paper presents an agent-tracking framework for semi-structured, crowded video. This framework is used to investigate how large numbers of people respond to vocal commands with local feedback and an overhead camera video. This paper analyzes a video showing an overhead view of more than 200 people, each holding an umbrella. Each umbrella is equipped with red, blue, and green LED lights. While individual’s movements appear stochastic, the crowd’s motion under the vocal command formed a variety of patterns. The problem is challenging because umbrellas often overlap, frequently change color, and occasionally leave the camera frame. This paper uses K-means clustering to separate umbrella from each other in each frame. Kalman filtering is used to estimate how each umbrella moves and track their motion path. This paper investigates the response time of the crowd and their accuracy in responding to vocal commands. In particular, we present results on: (1.) automatic segmentation and classification of each umbrella. (2). analysis of the response time of a human swarm to a simple vocal command. (3.) Measuring the accuracy of human swarm movement (4.) Calculating the learning rate of a human swarm, and (5.) documenting the position memory of a human swarm.

Keywords—K-means clustering, vision tracking, Kalman filter

# Introduction and Related Work

This paper presents a method to track the motion paths of multiple agents in semi-structured crowded scenes and analyze data from a crowd controlled by vocal commands.

The solution for vision tracking is coded in Matlab, and is available at [1]. The analysis uses video data from *UP: The Umbrella Project* [2],a beautiful experiment conducted at night on a football field in which more than two hundred people were each given an instrumented umbrella equipped with an RGB LED. Using vocal commands from a director on an elevated platform, and an overhead camera view projected on a large screen, the participants were divided into several groups according to their major, gender, or grade and then directed to form different shapes in various colors.

In an unstructured crowded scene, the motion of the crowd appears to be random, with different participants moving in different directions at different times [3]. This scenario has some structure because it is controlled by one person, the voice giving the vocal commands, but errors cannot be avoided completely. Moreover, tracking is challenging because the umbrellas switch colors rapidly and often overlap. Fig.1 shows a representative screenshot, the results of our analysis software, and a plot showing umbrella color counts as a function of time.

This paper analyzes an overhead video showing illuminated

umbrellas.

(1) raw data, captured from overhead video.

(2) classified umbrellas in the processed image.

(3) umbrella color count as a function of time is one form of data that is

generated.

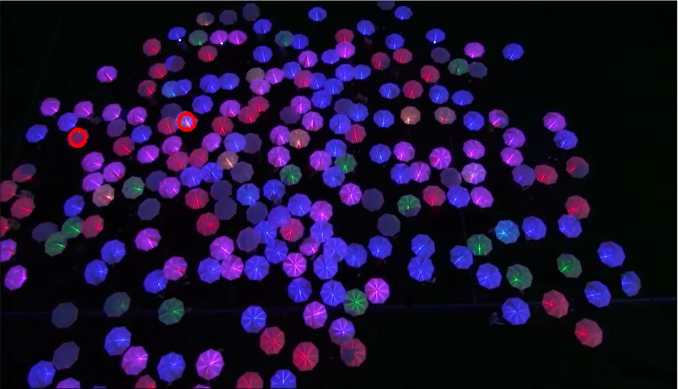
Object tracking is a key problem in the field of computer vision, and is especially challenging when tracking multiple objects in unstructured, crowded scenes. Tracking moving objects in video has a variety of applications, including automated surveillance, military guidance, traffic management system, robot vision and artificial intelligence [4].

This original video is available at [5]. Tracking multiple objects is more difficult than tracking one object for several reasons. *Data association*, the matching between targets and observations from frame to frame in a video sequence, is one difficulty [6]. Because objects are continually moving, they often overlap partially or completely. Sometimes the objects disappear and occasionally new objects enter the frame. To address these problems, this paper uses Kalman filters to track multiple objects [7].

The first challenge is to segment individual umbrellas. The solution employed is to erode all components to shrink to points. These points will not overlap and denote the centroid of each object. In this paper we apply data clustering to verify the centroids of each object. Data clustering is frequently used in many fields, including data mining, pattern recognition, decision support, machine learning and image segmentation [8]. In this paper we adapt one of the most widely used formulations to solve this problem, the K-means clustering algorithm. Given a set of *n* data points in real *d*-dimensional space, *Rd*, and an integer *k*, the problem is to determine a set of *k* points in *Rd*, called centers, so as to minimize the mean squared distance from each data point to its nearest center [9].

The umbrellas in this project are not moving aimlessly. At one frame, all may be the same color, but later the all umbrellas may change to another color and later form a colorful image. In the video, umbrellas form a smiley face, and later change to a snake, and finally form a word. All these transformation occurred under the direction of a vocal command. This paper presents an analysis on how the agents response to the vocal command.

This section, describes the approaches used: *K*-means clustering, Kalman filter algorithm, and techniques to, monitor the transformation of umbrellas’ color and pattern.



## Data clustering

[4] Since the square root is a monotone function, this also is the minimum

Euclidean distance assignment.

The data is a video recorded by an overhead camera showing how umbrellas respond to a vocal command. The first step is to identify the umbrellas, and record their positions However, both the numbers and positions of umbrellas are not a constant, this number changes as umbrellas enter and leave the field of view or lose battery power.

The aim of the *K*-means algorithm is to divide *M* points in *N* dimensions into *K* clusters so that the within-cluster sum of squares is minimized [10]. K-means returns a locally optimal clustering solution, and is dependent on a trustworthy initial value. We use manual identification for the first frame, and use the centroids from the previous frame as initial seed values for each successive frame.

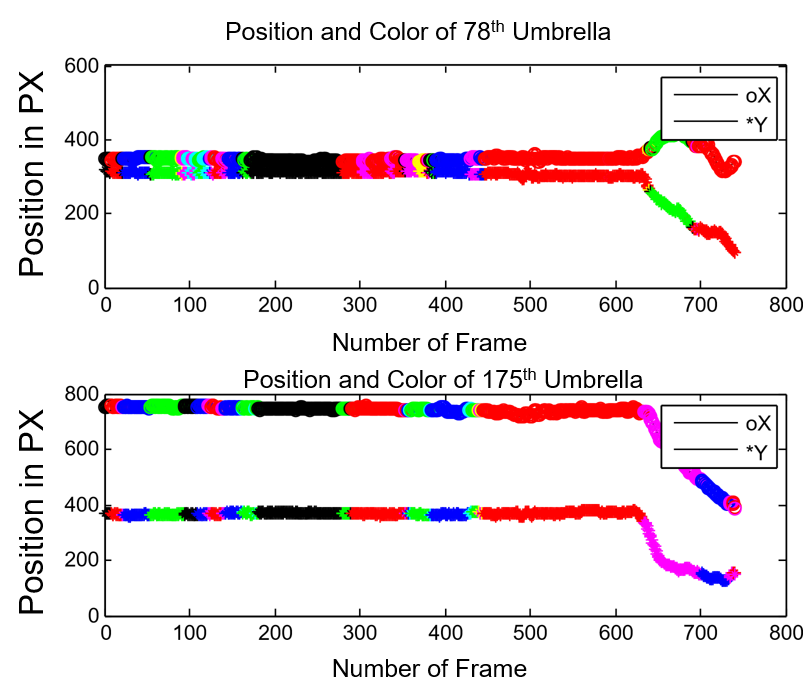
As shown in Fig. 1, each umbrella is marked by at its centroid. The centroid data and the average color of pixels in its neighborhood are stored to a file.

# Experiments and Results

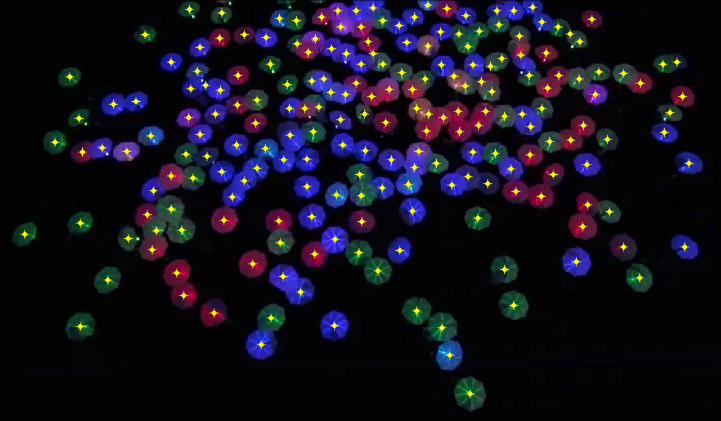
This section describes measurements obtained from *UP: The Umbrella Project*.

## Verify each umbrella and collect data information

Our software analyzed every 15th frame. In the first frame, each umbrella’s centroid was manually marked and those centroids were used as seeds for the next frame. In the next frame, *K*-means was used refine the seeds’ position to ensure they are in the middle of each umbrella.



This analysis is implemented using Matlab, code is accessible at [11]. The resulting data, saved as a video, is available at [12].



Umbrellas’ position data was first collected and stored, then it can be used in

by a Kalman Filter to track the motion paths.

## Tracking umbrellas’ motion path

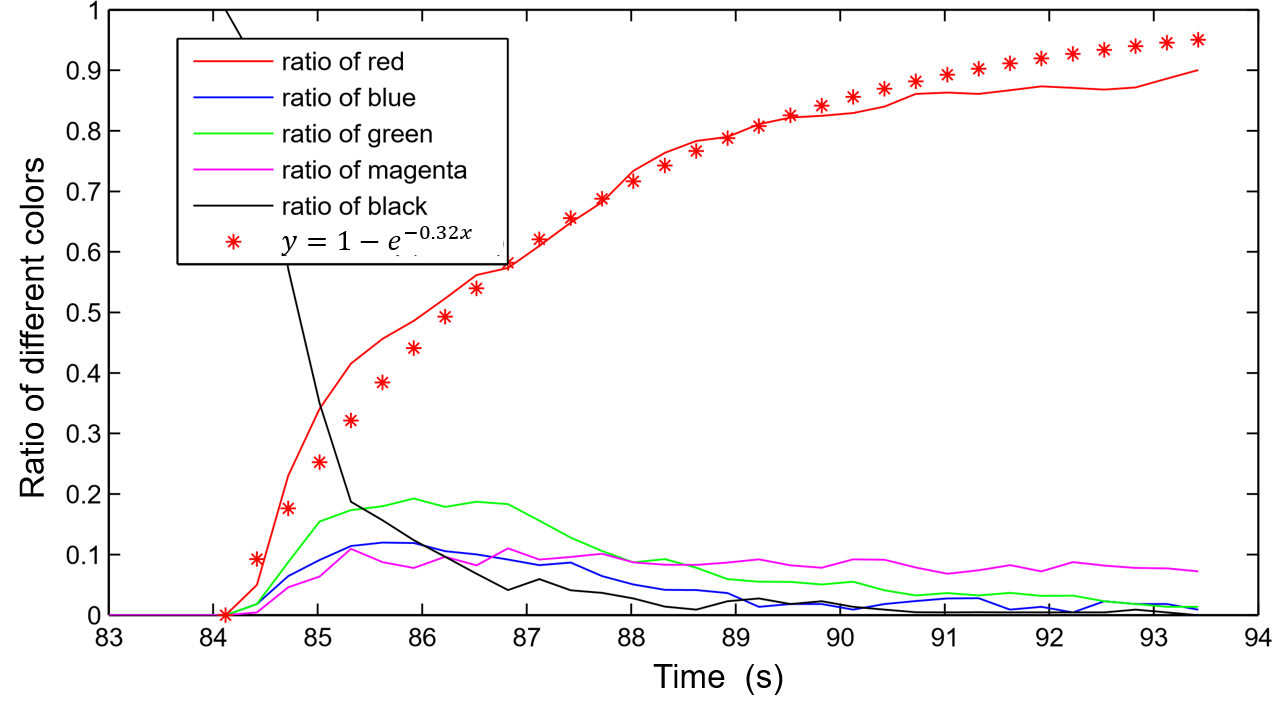
To detect umbrella positions and show how they moved, umbrellas are represented as a set of circles with three parameters, the center of a circle, and the circle radius. The user interface uses two circles to evaluate how the umbrellas move that are green and red. The green circle is the observed position of umbrella and the red circle is the estimated position which is calculated after applying the Kalman Filter, giving the result shown in Fig. 3, which shows that the two circles overlap well, which indicates the algorithm is working.

Applying Kalman filter tracking umbrella

Fig. 4 shows of *x* and *y* position of two umbrellas as a function of time with the lines drawn in the correct colors of the umbrellas, Fig. 5 shows the *x* and *y* velocity as a function of time with the lines drawn in the correct colors of the umbrellas. From the plot of *x* and *y* position, we used the umbrella’s position data, every time the umbrella moves from this frame to another, there will be a displacement distance on both *x* and *y* direction, so the *x* and *y* position are updated.

Change of two umbrellas’ position and color as a function of time

Similarly, the *x* and *y* velocities, is obtained by approximating motion as a straight-line movement between successive frames and dividing the distance travelled by time.

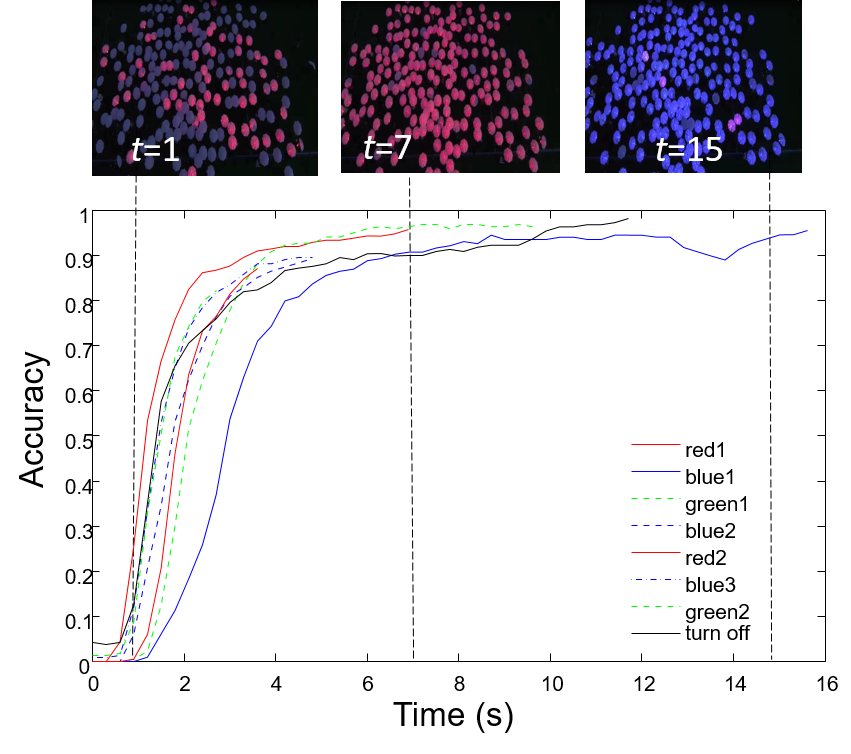


## Data analysis

Given the motion path of each umbrella, we want to know more details about the crowd, whether they performed well under the vocal command, and whether the swarm learns to better follow the directions of the vocal commands.

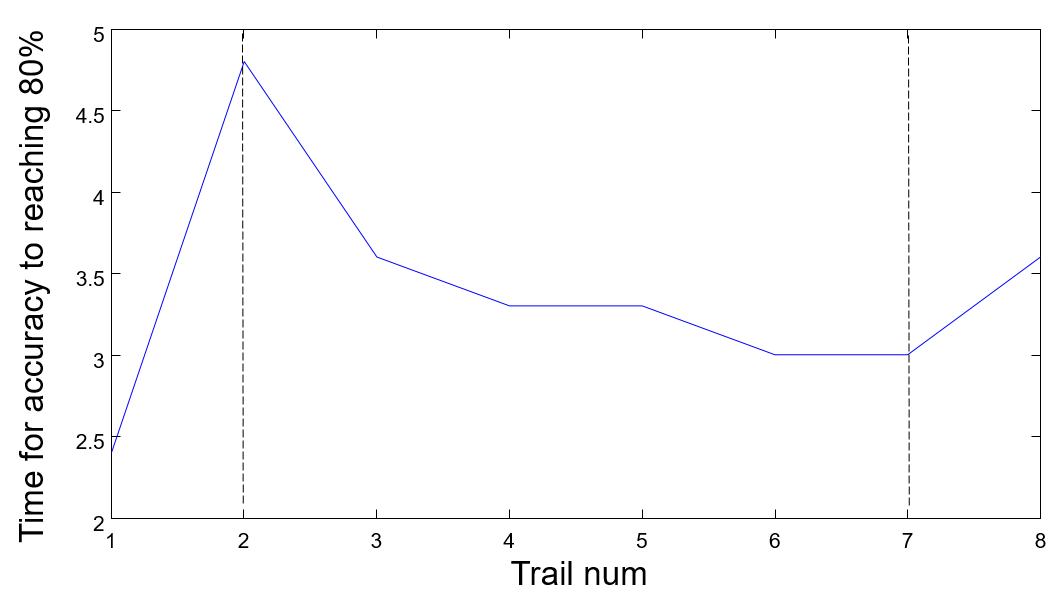
### Crowd response to a command to change color

In the video, there are several vocal commands which required people to change their umbrellas’ color. Commands included “*I want everybody to turn them red*”, and “*Now turn the red off. Turn the blue on!*” This test analyzes how people responded to eight color change commands. Results are summarized in Fig. 5



How crowd response to vocal command such as “*I want everybody to*

*turn them red*”



Eight vocal color-change commands were recorded. All data is aligned so the command begins at *t*=0s. For example, after the vocal command *“I want everybody to turn them red”*, people began to switch their umbrellas’ color at *t*=1s, and at *t*= 7 s, 90% of the umbrellas’ color changed to red. We define the response time as the time of the first response to when more than 90% of the umbrellas’ color change. In this case, the time constant is 6 seconds. During successive color change commands, it takes less time for 90% of the swarm to turn their umbrellas to one color. This means the swarm’s performance is increasing.

### The time constant for a harder vocal command

For simple color-change vocal command, people were able to achieve the goal in a short time. This section analyzes the time response to more complex commands, For example, at time *01:23*, the vocal command was *“When I say go I want you to turn them on and I want this whole group, this group that's gathered tonight to be one color but I'm not going to tell you what color that is”.* Fig. 6 shows the response and the time constant.

Response time for command *“when I say go I want you to turn them*

*on and I want this whole group, this group that's gathered tonight to be one color but I'm not going to tell you what color that is”,* people start to switch colors around *t*=82s, and by *t*=94s, all people turn to same color. The blue line shows an exponential fit with a time constant of 5 s.

This experiment is a classic distributed consensus problem. In this experiment, all people in the crowd must adjust their own color with their neighbors, but since the vocal command, is not specific on which color they need to turn, the process takes about 10 seconds. For this analysis, color umbrellas’ amount every frame, then we can find out which color the human swarm going to change. At the same time we can get the ratio of major color. An exponential function is fit to the data, giving 1-e-0.22 *t*.

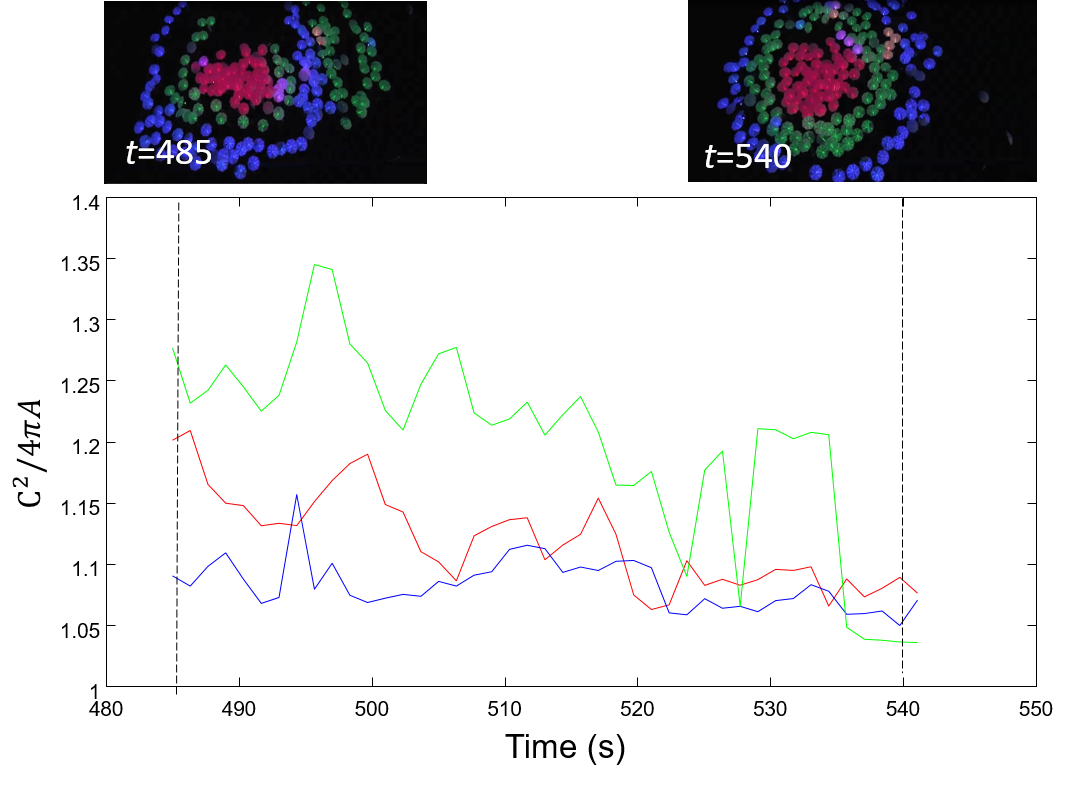
### Learning rate of the human swarm

The human swarm’s was asked to change colors eight times. Fig. 8 displays the time required for 90% of the swarm to achieve the desired value.

For color-change vocal command such as “*I want everybody to turn*

*them red*”, “*turn the red off turn the blue on*”, or “*let's go to green*” the swarm respond time tends to reduce, demonstrating that the swarm is learning.

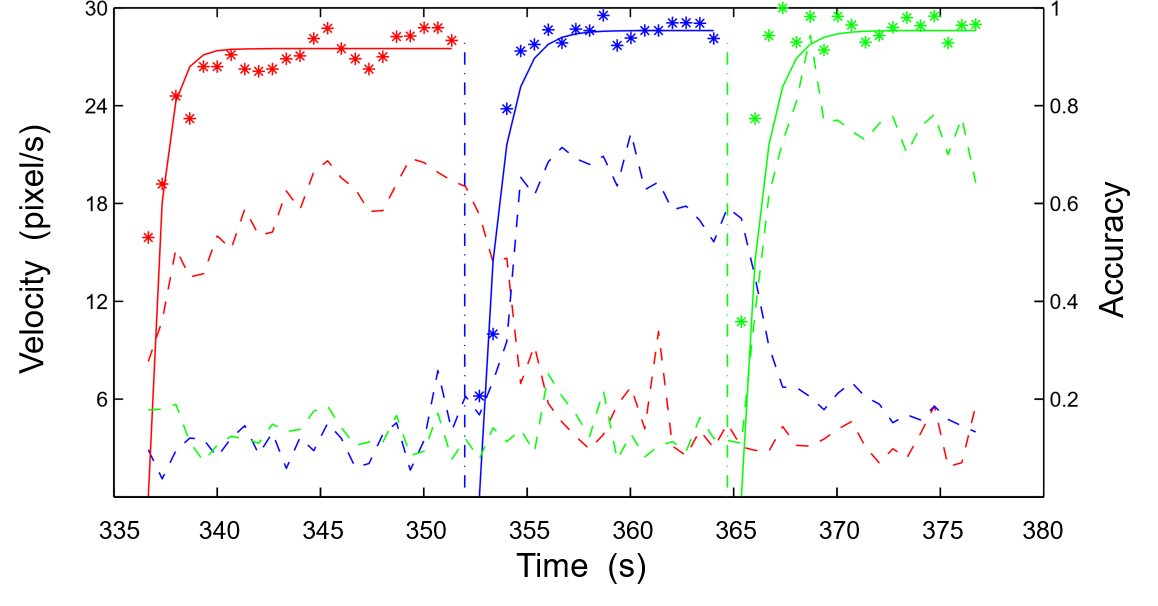
In Fig. 8 shows the time for accuracy to reach 80%. One experiment inserted a new vocal command before 90% of the human swarm achieved the desired color, so this analysis defined a successful convergence as when the ratio of major color reach 80%. The overall trend for human swarm is to take less time, but in the first trial, we can see that it only took *2.5* seconds for human swarm to accomplish the command, which is extremely faster than others, and at No.8 trail, it takes a longer time. There are several reasons. In the first trail, people had not turned on their lights. When they heard the vocal command *“I want everybody to turn them red”* they responded quickly without confusion. For next command, the initial umbrella colors were mixed, and the people required more. The last trail asked the human swarm to turn off the light. This was a novel command and it required more time to complete.



If we exclude the special events, we can conclude that human swarm responded more quickly each time.

### Comparing accuracy between similar commands

There are some vocal commands, very similar, so we want to know in those kinds of commands, which one they performed better. For example, for the command *“if you're red Move!”* how accurate were they? Compared to *“When I say go I want the red to freeze and the blue's to move … Go!”* And *“Let's try that with the green Go!”* we got the result in Fig. 8



Comparing the accuracy of two vocal command *“If you are red move.”*

and *“When I say go I want red to freeze and the blues’ to move.”* In every 20 frames, if the umbrella’s new position is more than a quarter of its radius with it’s original position, we define this umbrella is moving. Umbrellas’ velocity under commands *“If you are red move.”* and *“When I say go I want red to freeze and the blues’ to move.”*

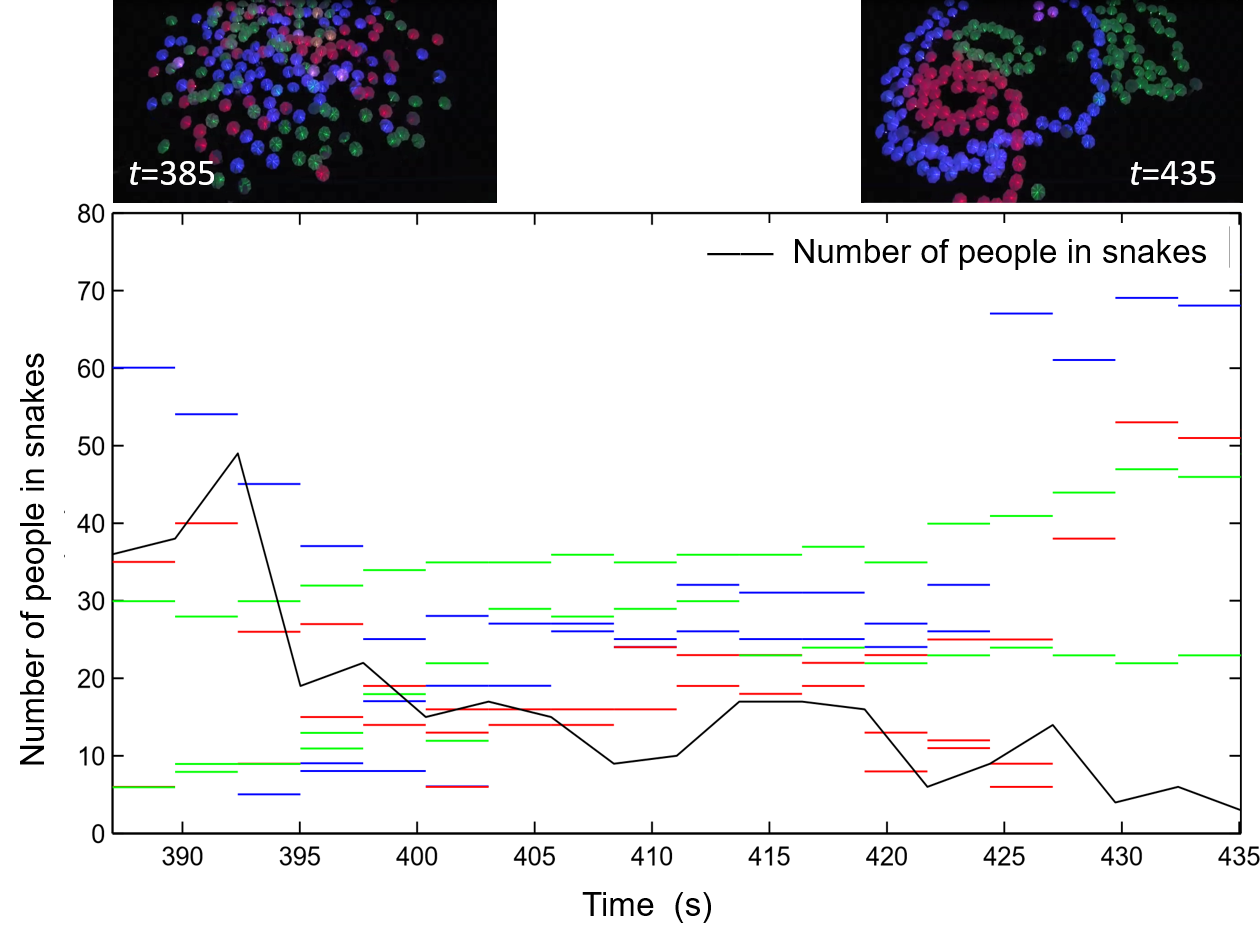


Fig. 8 shows that the swarm accuracy increased in response to *“When I say go I want red to freeze and the blues’ to move”.* An exponential function fit shows when the swarm achieved 0.9 accuracy, which means more than 90% of the umbrellas are following the vocal command.

Through this figure, we can see the difference between the red’s velocity and blue’s velocity. Before the vocal command *“Go”* red umbrellas are moving, blue umbrellas are not moving, after that the command, reds freeze and blues move instead.

### Shape-matching abilities

Besides changing their color, the human swarm were given harder commands including to form circles. The accuracy of circle formation of the human swarm is shown in Fig. 9.

Calculating the circularity of the human swarm when commanded

*“**Red stop and bunch up.  See how round you can be, keep circling around them greens.”.* There are three circles with different colors, values closer to one means the swarm shape is closer to a true circle.

From the figure we can see that at *t*=485 second, the human swarm was given the vocal command *“Red stop and bunch up.  See how round you can be, keep circling around them greens.”* and then they began to move. The human swarm followed this command till *t*=540 seconds, when a new command was announced. During this time the, circles became increasingly round. The equation is used to evaluate how round the circles are. Values close to 1 indicate a more accurate circle. The smaller the value, the more round the circle is. Three circles were formed by three different colors, and each color became increasingly more round.

### Forming a human swarm into a “snake”

The human swarm was told to form a “Snake”, which means they were divided into three different groups based on their color, asked to connect with their neighbors, and move, just like a snake. To evaluate whether the “snake” is good or not, we plotted the number of umbrellas are in the snakes as a function of time, and how many umbrellas were not in a snake. See the following Fig. 10

This plot shows how long it takes to form snakes, *x*-axis shows the

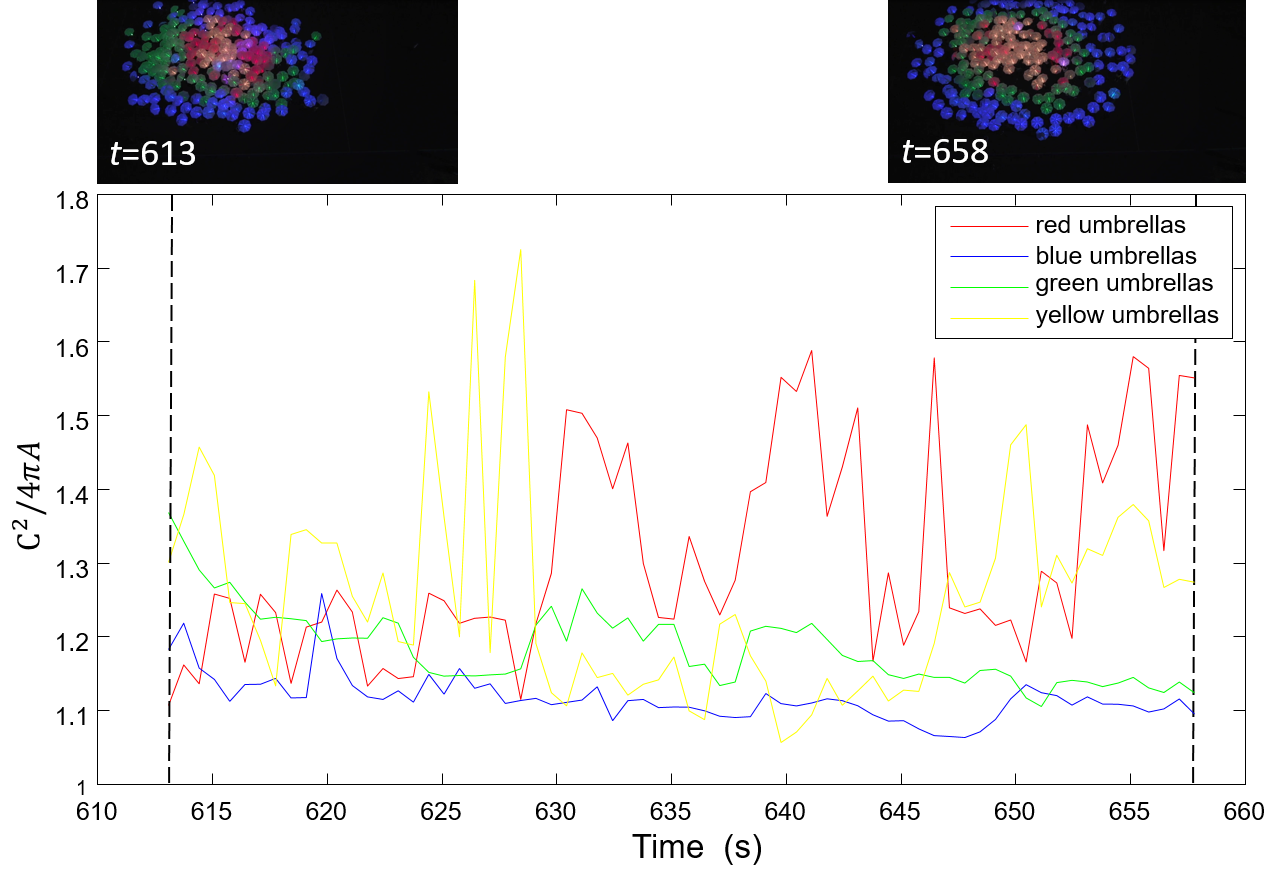
time it takes, and *y*-axis shows how long the snake is, the number of umbrellas are in (or not in) each snake. There are three different colors of snakes in the video. At time t=390, there are three green snakes with 9, 10, 28 people, one 40-member red snake, one 53-member blue snake, and 38 unaffiliated people.

The human swarm began to form snakes at the moment they heard the vocal command, at *t*= 385 seconds, and this command completed at *t*=435 seconds. During this period, the number of people in snakes are increasing, while the number not in a snake decreased. At the end there are four snakes in the image.

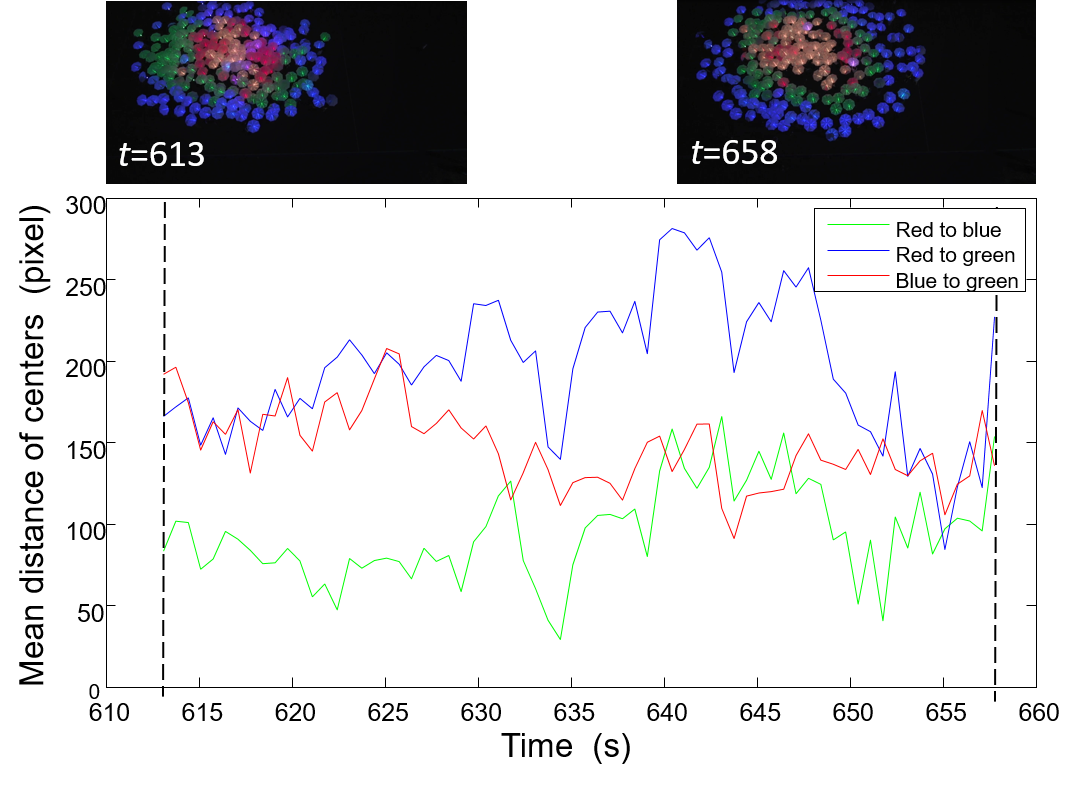
For this experiment, a *snake* is defined as at least five same-colored umbrellas, where each successive umbrellas is within six umbrella radius of a neighbor.

### Accuracy of the “Bullseye” configuration

At *t* =613 the human swarm was directed to form a “Bullseye”. To evaluate their performance, we analyzed how round each circle is, and calculate the mean distance between each circle’s centers. In a perfect bullseye, all circles’ are concentric. Figs. 11 and 12, shows the results of each.



Evaluste how round each circle got in bullseye when the command



as *“I would like to see three stripes, You know, like in the middle one color”*

Calculate the distance between each two circles to see whether

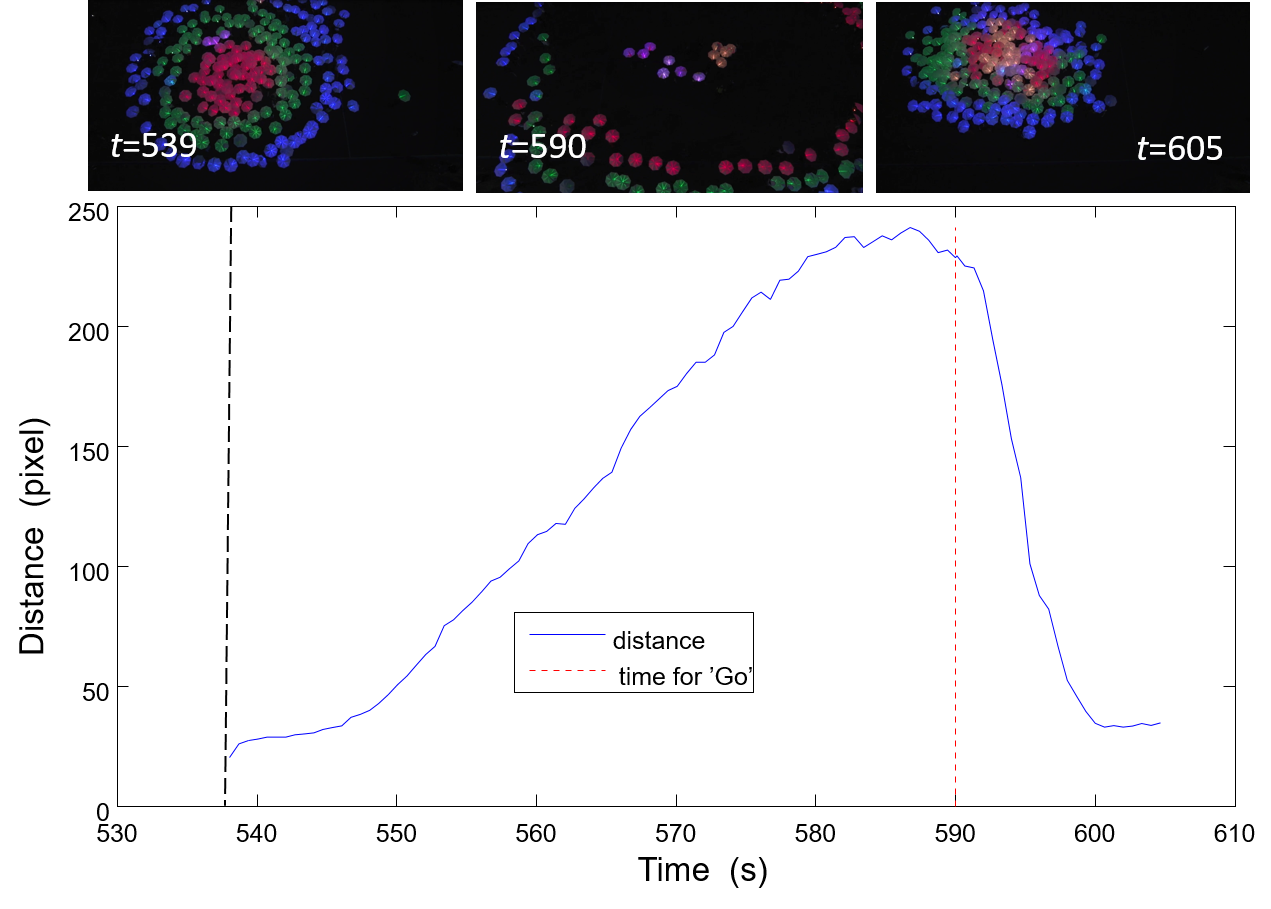
they formed a good bullseye

From two figures above we can conclude that the human swarm were unable to form a good bullseye. At first, according to the vocal command, it was required to form three kind colors of circles, which is red, green and blue. However, as we can see from the figure, since *t= 610* seconds, there are four different colors: red, green, blue and yellow. This situation may be caused by the LED light itself, so it may not be human swarm’s fault.

However, in the second figure, we did not count in the yellow circle. But the mean distance between each two circle did not reduce or keep a constant, this can be approved from the video capture too, we can see that bullseye had not been formed very successfully.

### The accuracy of the human swarm’s memory

This experiment analyzes the accuracy of people when they commanded to return back to known position. Results are shown in Fig. 13



This figure shows the accuracy of a swarm’s position memory. It

compares the mean distance between everyone’s “original position” and the “returned position”. Smaller distances indicate better memory. When they heard the vocal command “*When I say go I want you to move back, with the yellow right in the middle. Right back to where you were that same formation as quickly and safely as you can. As smoothly as you can.*” they returned to their original position.

In this figure we can see that from *t=539,* human swarm were spreading out slowly, and at *t=590* they were commanded to go back to their original position, during this period, the distance between current position and original position is increasing, which reflects the reality. After that moment, human swarm began to go back, so the distance is decreasing. Finally till *t=605* before next command, the distance is almost same with original one.

# Conclusion

Since the main tracking is based on Kalman Filter, the accuracy is very important. If the observed data matches the estimated data, this model for tracking umbrella seems good. While tracking the objects, the initial state and noise covariance influence a lot, maybe more than that, we need to tune the estimation functions to speed up our tracking system cause when tracking many objects, speed up is important. And tuning of Kalman Filter refers to estimation of covariance matrix, if it is not tuned properly, it leads to divergence of expected value from the actual value [9] In this project, the number of umbrellas are not constant all the time, it may change, and we need to track all the umbrellas.

In this project, we devote to find out a more effective way to track multi objects based on the existed achievement. Kalman Filter actually has a very wide application in many fields, in the object tracking it works well too. We set up a tracking system which should be able to track multiply objects which have similar appearance, it can track several objects maybe more than two hundred at the same time, when some object just disappeared or added, it can be detected quickly and keep going tracking. More than that, we focused on the data analysis.

# Appendix

## K-means algorithm support

The *K*-means method uses *K* prototypes, the centroids of clusters, to characterize the data [8]. K-means seeks to minimize:

(1)

Here is the data matrix, is the centroid of cluster, and is the number of points in

K-means, has two steps: assignment and update. The first assignment step uses observed data to assign data points to the cluster which yields the minimum within-cluster sum of squares. The sum of squares is squared Euclidean distance, so this is the nearest mean [4]:

(2)

(3)

Equation (2) is used for assign objects, (3) is used to calculate new means to be the new centroids of the observations in the new clusters.

## Other equations applied

To predict where the umbrella is in the next frame, my thought is to calculate the distance between two centroids of the umbrella, the first centroid is the observed location of umbrella at this frame, the second centroid is the estimated *x*, *y* location of umbrella, which will be updated. What we need is to calculate the distance between two central points, pick up the one which is nearest with observed umbrella’s position, then this is the next position of the umbrella in the next frame. Because in a very short time between each frame, umbrella’s moving could be seen as move towards a straight line and with a constant velocity. After that, the estimated umbrella’s position at this frame can be used as observed position to estimate the next position of umbrella at next frame.

(4)

## Kalman filter algorithm support

In this section, describes tracking umbrellas using the Kalman Filter algorithm. It is a recursive algorithm so that new measurements can be processed when they arrived, then a new round of calculating begin [2]. It can filtering out the noise during the time finding out the best estimate data, and a Kalman Filter not only just clean up the data measurements, but also projects those measurements onto the state estimate.

The Kalman Filter maintains both an estimate of the state:

Estimate of given measurements , ,…

Estimate of given measurements , ,…

And the error covariance matrix of the state estimate:

-covariance of given,,…

-estimate of given,,…

The Kalman Filter recursive processing is separated into several stages. The first part consists of two equations is called “Time Update (Predict)”:

(5)

(6)

Equation (4) represents the predicted state, (5) represents the error covariance ahead. And the second part can be seen as “Measurement Update (Correct)

(7)

(8)

(9)

Equation (6) represents the Kalman Gain computed, (8) means update the estimate with measurement, (9) represents the update of error covariance.

When apply the Kalman Filter in this project trying to track the umbrellas’ motion path, we need to do is initialize the algorithm at first, and also need to define the main variables that will be used in the equation. According to the practical situation, umbrellas are moving in the whole video with an un-constant velocity, the noise should be considered. Here I define the measurement noise *R* in the horizontal direction both *x* axis and *y* axis, and the process noise covariance *Q*, the estimate of initial umbrella position variance. Then we defined the update equations which also is the coefficient matrices, can be seen as a physics based model, so that we can make an estimation where the umbrella will be for the next moment.

In the update equations, all matrix variables need to be defined:

Initialize *A* represents the state transition matrix; *B* represents the input matrix, which is optional; H represents the observation matrix, *K* represents the Kalman Gain. After that, we can call the Kalman Filter. As mentioned before, each iteration of Kalman Filter will update the estimate of state vector of a system based upon the information in a new observation. In this project, the data which had already been collected is the *x*, *y* location of each umbrella at each frame. Although these location data have some error but they are reliable enough and they are used as measurement data. To track the motion path of the umbrella, we set an empty matrix “centroids” to store the *x*, *y* locations of each umbrella, so this matrix can represents the real locations of umbrella.

# Reference

[*https://www.dropbox.com/s/0zvompid7dnzbxf/detectupdate1121\_3.m?dl=0*](https://www.dropbox.com/s/0zvompid7dnzbxf/detectupdate1121_3.m?dl=0).

https://vimeo.com/56647019

M. Betke et al., Tracking Large Variable Numbers of Objects in Clutter, IEEE CVPR, 2007.

Yang, Hanxuan, et al. "Recent advances and trends in visual tracking: A review." *Neurocomputing* 74.18 (2011): 3823-3831.

[*https://www.dropbox.com/s/97h3jp4ist5tlvv/First10Min.mp4?dl=0*](https://www.dropbox.com/s/97h3jp4ist5tlvv/First10Min.mp4?dl=0)

Jeong, Jong-Min, Tae-Sung Yoon, and Jin-Bae Park. "Kalman filter based multiple objects detection-tracking algorithm robust to occlusion." *SICE Annual Conference (SICE), 2014 Proceedings of the*. IEEE, 2014.

Cuevas, Erik V., Daniel Zaldivar, and Raul Rojas. "Kalman filter for vision tracking." (2005).

Jain, Anil K., M. Narasimha Murty, and Patrick J. Flynn. "Data clustering: a review." *ACM computing surveys (CSUR)* 31.3 (1999): 264-323.

Kanungo, Tapas, et al. "An efficient k-means clustering algorithm: Analysis and implementation." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.7 (2002): 881-892.

Hargrave, P.J, “A tutorial introduction to Kalman filtering”, Kalman Filters: Introduction, Applications and Future Developments, IEE Colloquium on, 1989, pp. 1-6

1. [*https://www.dropbox.com/s/n0xiimufsp552pl/colorizeUmbrellaData.m?dl=0*](https://www.dropbox.com/s/n0xiimufsp552pl/colorizeUmbrellaData.m?dl=0)
2. [*https://www.dropbox.com/s/e3f22ouoqj5r17p/ProcessedUmbrella.mp4?dl=0*](https://www.dropbox.com/s/e3f22ouoqj5r17p/ProcessedUmbrella.mp4?dl=0)

Wei, Lai, Weiming Zeng, and Hong Wang. "K-means Clustering with Manifold." Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on. Vol. 5. IEEE, 2010.

Esteves, Rui Máximo, Thomas Hacker, and Chunming Rong. "Competitive k-means, a new accurate and distributed k-means algorithm for large datasets." *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*. Vol. 1. IEEE, 2013.

Rodriguez, M, Ali, S, and Kanade, T, “Tracking in Unstructured Crowded Scenes”, Computer Vision, 2009 IEEE 12th International Conference on, 2009, pp.1389-1396.

J. Hartigan, M. Wang, A K-means clustering algorithm, Applied Statistics 1979, 28:100-108

Zhao, Tao, and Ram Nevatia. "Tracking multiple humans in complex situations." Pattern Analysis and Machine Intelligence, IEEE Transactions on 26.9 (2004): 1208-1221.

Y.Cai et al., Robust Visual Tracking of Multiple Targets, ECCV, 2006.

Thrun, Sebastian, Wolfram Burgard, and Dieter Fox. Probabilistic robotics. MIT press, 2005.

Ramakoti, Nimmakayala, Ari Vinay, and Ravi Kumar Jatoth. "Particle swarm optimization aided Kalman filter for object tracking." Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT'09. International Conference on. IEEE, 2009.