Improving Gaze Estimate Accuracy for Eye Tracking Device

Kien Tran, Yamin Tun

ABSTRACT

The goal of our project is to improve the accuracy of the gaze estimation provided by iShadow's neural network model by using saliency information offline. The gaze error could occur due to inaccuracy of the neural network itself, as well as calibration error. To reduce the gaze error, we intend to use the saliency map of the environment images that are captured by the outward camera of the glasses. Saliency map [2][3] is a topographically arranged map that represents the quality by which each pixel stands out relative to its surrounding. Using Saliency Toolbox and motion detection, we detect the positions in visual scene, at which human eyes tend to gaze. With the use of saliency information of the environment scene, we will then create a simple probabilistic model that uses intensity-and-motion-based saliency to correct the gaze estimate. According to the experimental results, we found out that the average error reduces approximately by half for a simple moving circle image after using the intensity-based saliency information. For complex images, intensity-based saliency alone does not significantly improve the accuracy of gaze estimation. However, according to the experimental results, the combination of motion-based saliency and intensity-based saliency improves the gaze estimation significantly for a series of image frames with moving object. In the experiment, iShadow user looks at the moving object. We conclude that saliency improves the accuracy of gaze estimation in general.

Key words: gaze estimation, gaze error, saliency map, affine transformation

I. INTRODUCTION

Our project is an extension of iShadow eye tracker, which is developed by a PhD candidate from UMass Sensor Lab, Addison Mayberry and his team. The iShadow project team [1] has previously developed an end-to-end system with novel ultra-low power computation eyeglasses that is capable of detecting eye movement and estimating eye gaze relative to the external environment. The system uses neural network model to predict gaze point based on eye tracking. The system's estimation error is roughly 3 degrees.

II. BACKGROUND AND MOTIVATION

1. Motivation

Gaze tracking is measuring points and estimating where a person is looking at in the physical world using eye movement information. Real-time gaze tracking can be useful in several applications such as detecting unsafe behavior such as tracking driver's gaze and detecting whether driver has attention on the road [1]. This can further prevent accidents and enhance road safety. To accomplish such intention, one of the most important characteristics of eye tracker is the accuracy of gaze estimation. In this project, we intend to improve the accuracy of gaze estimation of a previously developed framework: iShadow EyeTracker.

2. iShadow Glasses

Most commercial eye tracker in the market are computationally intensive, high-power and bulky devices. Unlike the existing eye trackers, the iShadow glasses consists of compact hardware and tracks gaze in real-time under low power. The iShadow glasses use eye images of user in real-time to estimate his or her gaze.

a. Hardware

The device contains two grayscale cameras: an eye-facing camera and an outside-world-facing camera (111x112 resolution at 10Hz ~5fps), which are connected to the STM32 Cortex F4 microcontroller on the right side of the glasses frame for real-time eye tracking[1]. The gaze estimate points are either stored in SD card on the glasses or streamed to a computer via USB.

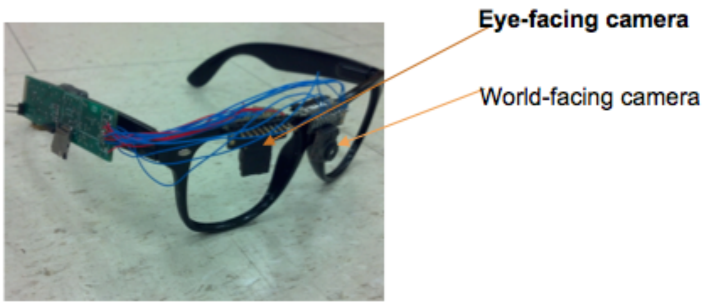


Figure 1. iShadow Gaze Tracker [1]

b. Software

The real-time gaze tracking is accomplished by using a neural network model. During the training process, the user wears the iShadow glasses and looks at the moving black circle in the white background on the computer screen for 10 minutes to collect the training eye image data and true gaze point data. The neural network model is constructed by using the training data for individual user. The code for neural network was written in C and loaded onto the microcontroller for real-time gaze tracking. This part of the project is already completed by iShadow team.

c. Limitation of iShadow

The previously developed iShadow model has several limitations that introduces error to gaze estimation. One factor that causes error is the performance of neural network on the glasses itself. In order to track gaze in real-time under low power, the glasses use sparse sampling to select a relatively low number of pixels from eye image and feed into the neural network to produce gaze estimate, instead of using all the pixels from the eye image. The selected pixels might not always be the best features that indicates eye movement. In addition, since the camera on the glasses is a low resolution grayscale imager that could be worsened by environment lighting, the eye images itself contain noises. This could severely affect sparse sampling process. As a result, the gaze estimate from the neural network is not as accurate as state-of-art eye trackers. Other calibration factors such as glasses position and lighting could also cause the error to gaze estimation. For example, the calibration error could be arisen due to different positions of the glasses relative to the user's eyes throughout the time when training and testing data were recorded. This is also unavoidable if the user adjusts the position of the glasses or if another user wears the glasses.

The difference in the position of glasses affects the angle of camera and eye images captured by the camera, which further impacts the performance of neural network. In addition, since the glasses do not detect blinks, the neural network still provides gaze estimate during blinks. The goal of our project is to increase the accuracy of iShadow's gaze estimate using saliency information of outside world scenes.

3. Saliency

Saliency map is a topographically arranged map that represents the quality by which each pixel in an image stands out relative to its surrounding [2][3]. Saliency map is constructed using different features of images that attracts users' attention such as color, intensity, orientation of objects, motion and the presence of face. Saliency map indicates the regions in the image where humans are more likely to focus their attention of sight. By utilizing such characteristic of saliency map, we intend to improve the accuracy of the gaze estimate provided by neural network of the existing iShadow framework. Below is an example of saliency map that is based on different features such as color, intensity and orientation. Although the saliency toolbox that we use does not provide saliency map that is as accurate as the following saliency map in figure 2, it was sufficiently good enough to correct the gaze estimate most of the time.

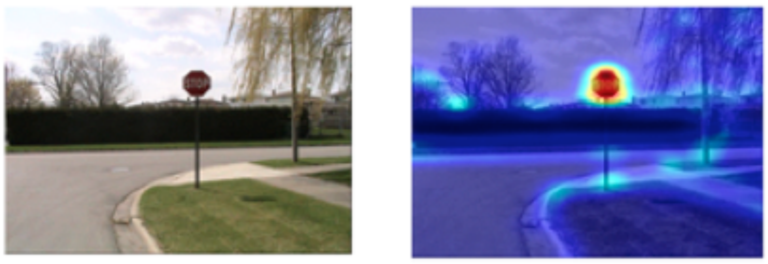


Figure 2. Original outdoor scene and saliency map of the scene [8]

III. METHOD

The iShadow glasses utilize only the eye-facing camera images to feed into the neural network and estimate gaze coordinates. Considering the fact that the information from the outward-facing camera is abundant and possible to utilize to improve the accuracy of gaze estimation, we use saliency information from the outward camera and build a simple probabilistic model to improve the gaze estimation of iShadow.

1. Overview

Our project is composed of two parts. In the first part of our project, we explored the way to increase the gaze accuracy by using saliency based on motion. In the second part, we used the intensity of grayscale image from the outward camera of iShadow as the feature to build saliency map and attempted to increase the accuracy of gaze estimate. Finally, we applied the combined saliency based on both motion and intensity to a series of video frames that was recorded on iShadow and observed the gaze estimate output.

2. Saliency Toolbox

The three common features of saliency map are color, intensity, orientation, motion and presence of human face. Out of these features, only intensity and motion were used to compute saliency map since the iShadow's camera is grayscale, and orientation does not provide much intuitive saliency information for the toolbox that we are using.

We used the matlab saliency toolbox created by Dirk B. Walther in the Koch Lab at the California Institute of Technology to produce the saliency map using image intensity [4]. We implemented the algorithm to locate a potentially more accurate gaze estimate coordinate using saliency information from the toolbox. In addition, we produced saliency map based on motion by referring to a matlab example of background subtraction [5]. However, the example uses hsv information to infer motion information. To accommodate the grayscale images, we have modified and extended the example implementation of background subtraction to account for camera movement (or head movement of iShadow user) and produce motion-based saliency map.

3. Data

The data sets that we used include continuous series of image frames recorded on the outward-facing camera on iShadow glasses, as well as grayscale images from online saliency database [6].

For the intensity-based saliency, the following datasets were used:

1. Image series of a black circle that is moving in a random path on the computer screen, recorded on iShadow [1]
2. Online saliency image database from MIT [6]
3. A balloon image downloaded online [7]

For motion-based saliency, we used the following dataset recorded by Addison:

1. Image series of a person walking in a room with a stable background, recorded on iShadow [1]

Along with the image data, Addison provided us with true gaze point and iShadow’s gaze estimate to us for moving dot dataset. The iShadow’s gaze estimates for the corresponding datasets were compared with our gaze estimates after using saliency information to observe if saliency information improves the gaze estimate. The true gaze points for the iShadow’s moving-circle dataset are provided by Addison. The true gaze points were produced by a simple matlab function, which locates a point that is somewhere inside the moving circle. We observed that the true gaze points do not always coincide with the center of the moving circle.

We have observed that iShadow’s gaze estimates for the moving dot on the computer screen has a significantly higher accuracy in estimating gaze than the gaze estimates for the walking-person dataset. It could be said that the walking-person dataset was coarse testing dataset that was collected in the wild since the iShadow user sometimes move the head. The true gaze points for the walking-person dataset were not labeled. Because the user was told to look at the person walking, it was assumed that the gaze points roughly lie in the area of the moving person’s head. It is good to note that although the gaze estimates from neural network was close to the head area most of the time, but sometimes wildly move away from the head area. This could possibly be due to blinks or glasses movement. We intend to correct such error using saliency.

4. Algorithm Outline: Finding New Gaze Estimate Using Saliency

The following figure illustrates the outline of our algorithm to produce a new gaze point based on gaze estimate produced from iShadow’s neural network. The selected image was converted to grayscale, which, in turn, produces saliency map using its grayscale intensity values and motion if it is an image frame from a video. Assuming that iShadow’s neural network produces reasonably accurate gaze points most of the time, the saliency values closer to iShadow’s gaze estimate must be weighed more than the values further from the gaze point. To create such effect, the gaze estimate point from neural network is overlayed on the saliency map. The gaussian kernel is applied to the saliency map with its center at iShadow’s gaze estimate coordinate. Details on how we choose the best sigma (standard deviation) value for gaussian kernel is explained in experiment section. The resultant map is the score map, in which the values in saliency map decrease radially as it moves away from iShadow’s gaze estimate point. Finally, we search for the coordinate in the image where the score is the highest in the entire score map. This coordinate is the new gaze coordinate after using saliency map. The following formula sums up our algorithm: $\boxed{\text{Gaze' s.t max (saliency\_score * g(}\sigma\text{))}}$

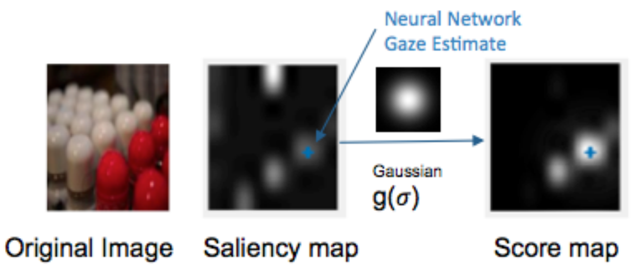


Figure 3. Algorithm outline for finding new gaze point using saliency

5. Intensity-based Saliency Map

We use the saliency toolbox developed by CalTech's Koch Lab to produce intensity-based saliency map. We observed that the saliency toolbox produces saliency maps at different detail levels for different image resolution. For example, in the first image below, the saliency map with large green blobs is produced for the input image at its original resolution. The man's face, the lady's face and some areas of their clothing have high saliency value in image (a). This is intuitive since people tend to look at the faces more compared to the background. As the resolution is increased in a factor of 2 by using imresize() function in matlab, the green blobs in saliency map become smaller. In other words, the saliency map shows the detailed area of the image where people tend to look at. For example, in the image (f), the saliency value is high in the area of the white button in the old lady's dark clothing background. Also, the guy's cheeks, teeth and the women's forehead show relatively high saliency values. These are small highlighted area with dark background, on which people usually pay attention when looking at the image. To conserve all the saliency information at different resolution level, we add all the saliency value with the weight levels that increases with scale. For example, the weight for the original resolution is 1, and the weight for the two times of original resolution is 2, and so on. The final saliency map is shown in the image below.

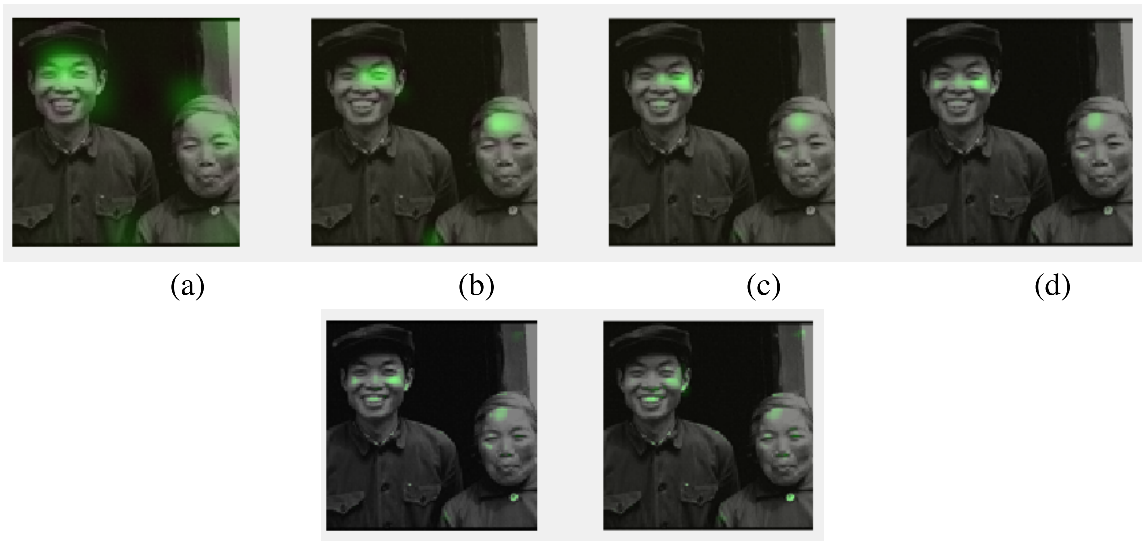


Figure 4. Intensity-based saliency map overlayed on the grayscale image for different resolution levels. Resolution increases by a factor of 2 from left to right

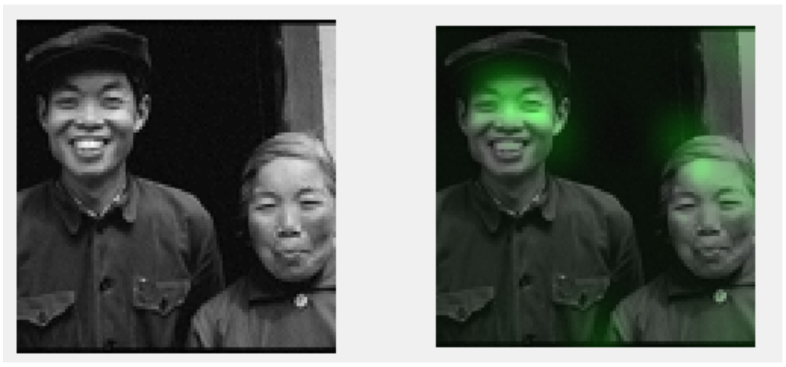


Figure 5. Final saliency map that combines saliency values at different image resolution

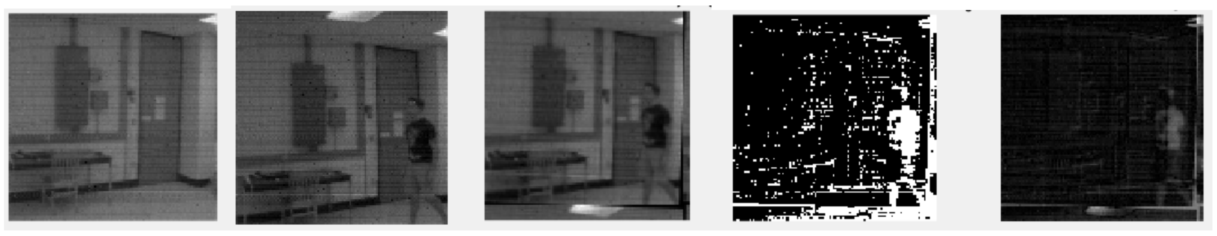
6. Motion-based Saliency Map

Background subtraction is implemented to produce motion-based saliency map. We have modified an example matlab script for background subtraction for colored images to use it for iShadow's grayscale images [5]. The first step for motion detection is to select the background frame with no moving object in the video. The background frame was selected manually in the series of images.

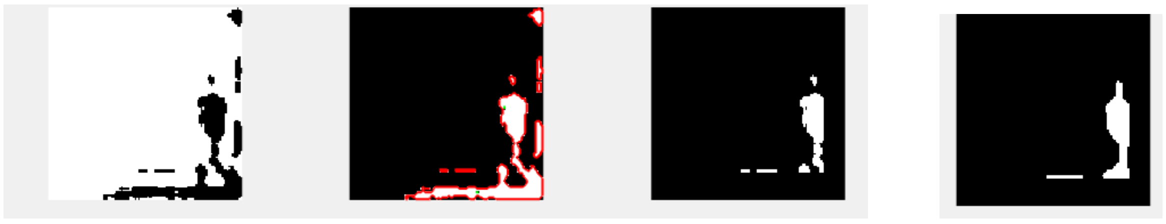
The challenge in background subtraction is the head movement of the iShadow user, which causes the outward-camera image frames to be not exactly aligned the entire time. In the images below 6(a) and 6(b), it can be seen that the background frame and the image frame with moving object are not perfectly aligned due to the head movement of the iShadow user. Since background subtraction assumes that the background and foreground frames are mostly aligned, the image alignment must be performed first before background subtraction. For image alignment, the best x and y values to shift and best rotation angle were exhaustively searched by finding the minimum difference between edges of background frame and edges of foreground frame. Canny edge detector was used for detecting edges in the images. Image 6(c) below shows the foreground image aligned with background image.

A binary image is produced by thresholding the difference of the background frame and each image frame in the video. Since the image is low resolution and noisy, subtracting image frame from background frame does not necessarily produce zero. Therefore, the best threshold for producing the binary image is selected to be a non-zero value through trial and error. The image 6(d) and 6(e) show the results of subtracting background frame from the image frame with moving object.

The next step is to determine the location of moving objects. Median filter was applied to remove most of the noises and merge some broken segments in the human body. The regionprops() matlab function was used to find different segments in the resultant binary image. The segments with area less than a selected threshold are considered as noises and eliminated. As shown in figure 6(f) and 6(g), the bottom and right borders of images were detected as moving objects, which are the artifacts of circshift() that was used for image alignment. In image (h), those artifacts were removed. In image (i), the broken segments were connected using 'close' matlab morphological operation, which is basically filling in the gaps between broken objects with disk-shaped blobs. In image (i), there still exists some artifacts of low resolution camera such as areas with intensity differences and pixelated lines in the image. To eliminate such artifact, regionprops() function was used for the second time to find different segments in the image. The biggest segment was chosen as the moving object. The resultant binary image is the motion-based saliency map as illustrated in figure (k).

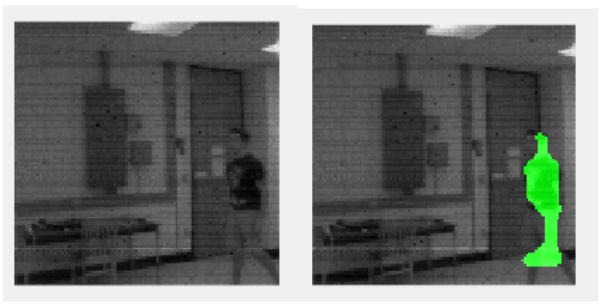

(a) (b) (c) (d) (e)

(a) Background (b) Original Image frame with moving object before aligning (c) Image frame with moving object after aligning (d) Thresholding result of subtracting background frame from aligned image frame (e) Result of subtracting background frame from aligned image frame


(f) (g) (h) (i)

(f) background detected and (g) blobs detected after applying median filter and eliminating small noise segments

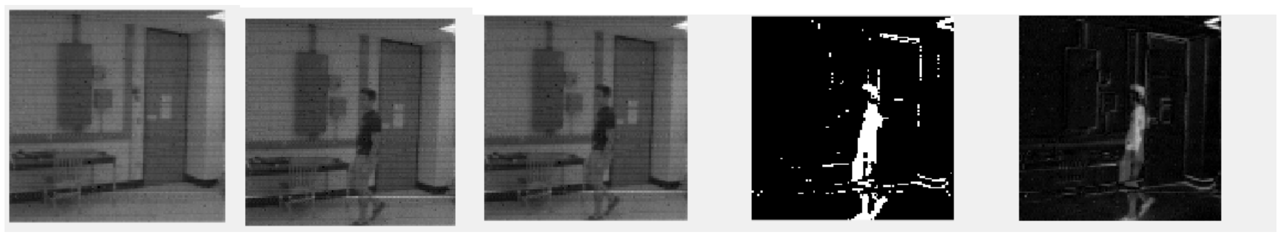
(h) motion saliency after removing the artifacts of aligning image (i) motion saliency after 'close' morphological operation


(j) (k)

(j) original image frame with moving object (k) motion-saliency map

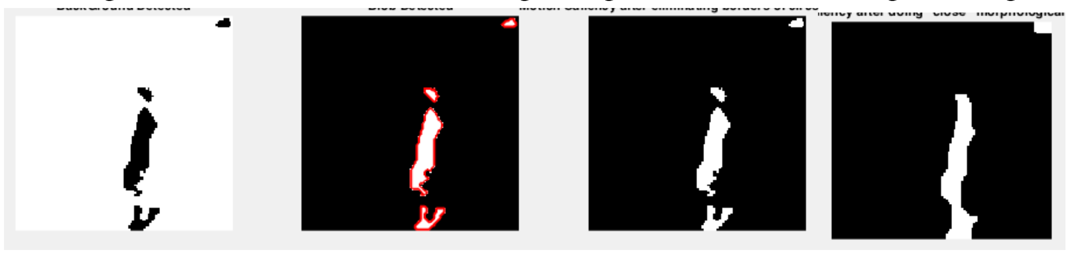
Figure 6. First Image Frame: Step-by-step procedure of producing motion-based saliency map and combined saliency map based on intensity and motion

Below is another example of processing to produce a motion-saliency map.



(a) (b) (c) (d) (e)

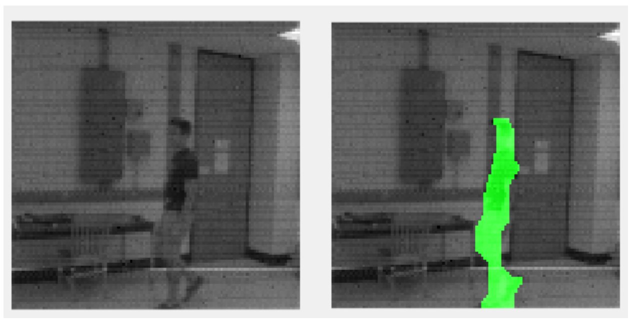
(a) Background (b) Original Image frame with moving object before aligning (c) Image frame with moving object after aligning (d) Thresholding result of subtracting background frame from aligned image frame (e) Result of subtracting background frame from aligned image frame



(f) (g) (h) (i)

(f) background detected and (g) blobs detected after applying median filter and eliminating small noise segments

(h) motion saliency after removing the artifacts of aligning image (i) motion saliency after 'close' morphological operation



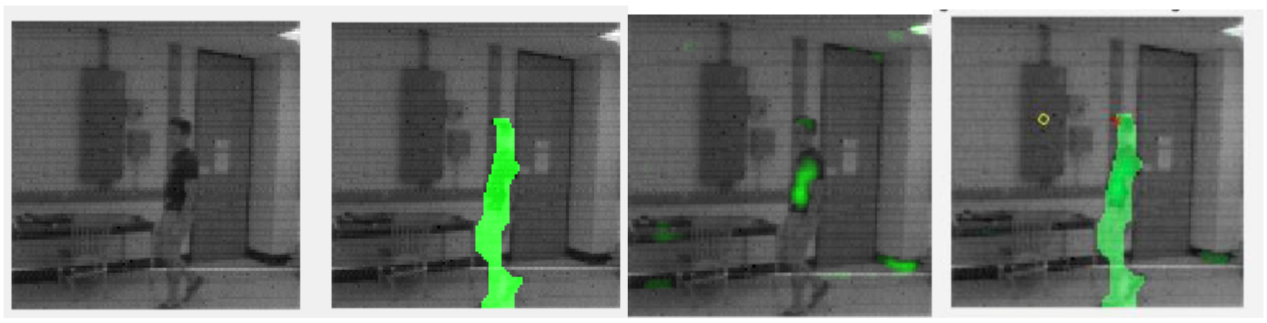
(j) (k)

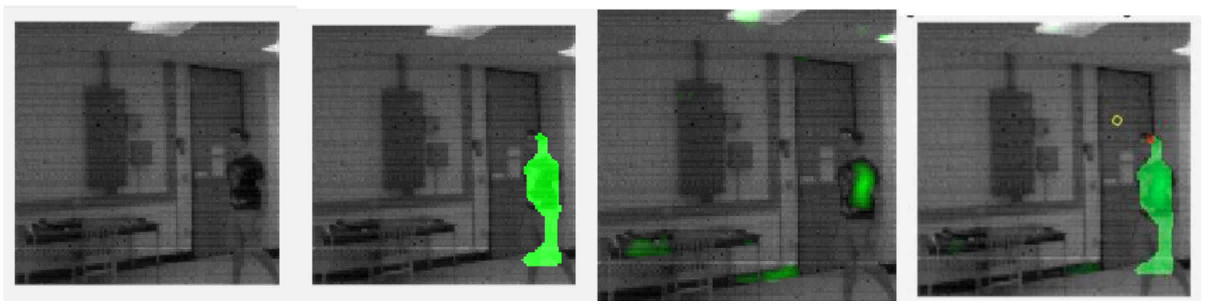
(j) original image frame with moving object (k) motion-saliency map

Figure 7. Second image frame: Step-by-step procedure of producing motion-based saliency map and combined saliency map based on intensity and motion

7. Combining motion-based saliency and intensity-based saliency

Figure 8(b) shows the motion-based saliency map and figure 8(c) shows the intensity-based saliency map produced from saliency toolbox [4]. To combine the motion-based saliency map and the intensity-based saliency map, the weight of each saliency map was determined as follows. Since humans' attention to motion is generally far more than their attention to stationary objects with high-contrasted intensity, we decided to weigh 0.8 for motion saliency and 0.5 for intensity saliency to produce the final normalized saliency map shown in 8(d).


(a) (b) (c) (d)
(a) original image frame with moving object (b) motion-saliency map (c) intensity-saliency map produced using saliency toolbox [4] (d) final combined saliency map based on intensity and motion- yellow circle represents gaze estimate before correcting with saliency, red cross represents gaze estimate after correcting with saliency
Figure 8. First Image Frame: Combining intensity-based saliency map and motion-based saliency map


(a) (b) (c) (d)
(a) original image frame with moving object (b) motion-saliency map (c) intensity-saliency map produced using saliency toolbox [4] (d) final combined saliency map based on intensity and motion- yellow circle represents gaze estimate before correcting with saliency, red cross represents gaze estimate after correcting with saliency
Figure 9. Second Image Frame: Combining intensity-based saliency map and motion-based saliency map

8. Limitation of Motion Detection

Our motion detection method still has limitations. Motion detection is sensitive to the difference between the intensity of the person's body and the background behind the body. For example, in the figure above, the person has dark hair and dark-colored shirt compared to the pale-colored background, which makes it easier to differentiate foreground and background. However, since the person's skin color has similar intensity level as the color of the door that is located behind the person in figure 6(c) and (d), the person's face and a part of his leg were not detected as moving objects.

Another limitation is that our motion-based saliency map does not indicate how fast the object moves between 2 consecutive image frames, but only the presence of the moving object. Since humans tend to look at fast moving objects compared to objects moving at a slower pace, the quantity of movement could be useful for inferring gaze. As a future work, the motion between consecutive image frames could be further explored.

Although our technique of image alignment works when the head of the iShadow user moves horizontally, vertically or rotates in an angle, our method of image alignment does not consider scaling. For example, the images below (figure 10) are some image frames where iShadow's user head moved closer to the moving object. If the user head moves closer or further away from the scene, the image could be aligned by trying different scales. The motion saliency map was not successfully produced due to such kind of head movement. This could be one of the future works.

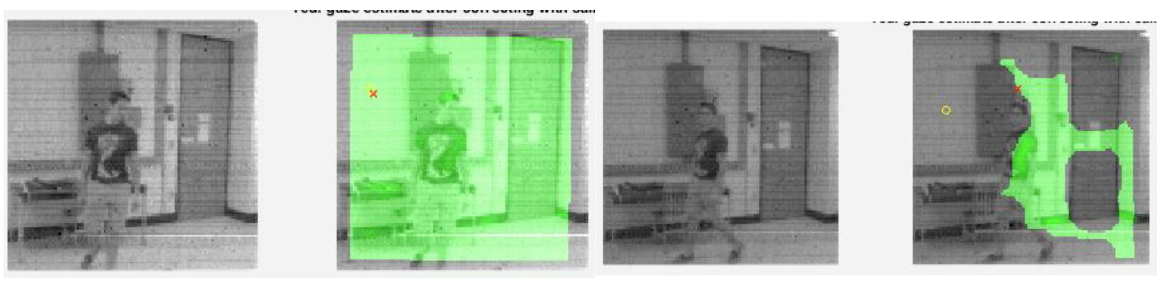


Figure 10. Motion saliency map when the head of iShadow user moved

IV. EXPERIMENTS & RESULTS

Part 1: Improving Gaze Estimation by Using Motion-and-Intensity-based Saliency

In method section III 6, we have constructed the combination of motion-based and intensity-based saliency map. In figures 8(d) and 9(d), iShadow's gaze estimate (yellow circle) is originally far off from the head area. By using the algorithm mentioned in section III 4, the gaze is corrected by moving it to local maxima of a selected standard deviation for gaussian kernel, which is 15 pixels (kernel size=31 pixels) shown in red cross. The new gaze estimate in red cross is relatively closer to the human's head compared to the original iShadow's gaze estimate in yellow circle. Therefore, motion-saliency significantly improves the gaze estimate.

The full video of the saliency map overlayed on consecutive 51 image frames that are recorded on iShadow is available at https://www.youtube.com/watch?v=_Q650MOkZs0. In all the image frames except the image frames where iShadow's camera moved closer to the moving object, iShadow's gaze estimate points are much closer to the moving person's head. Therefore, iShadow's gaze estimate is significantly improved by using motion-and-intensity-based saliency.

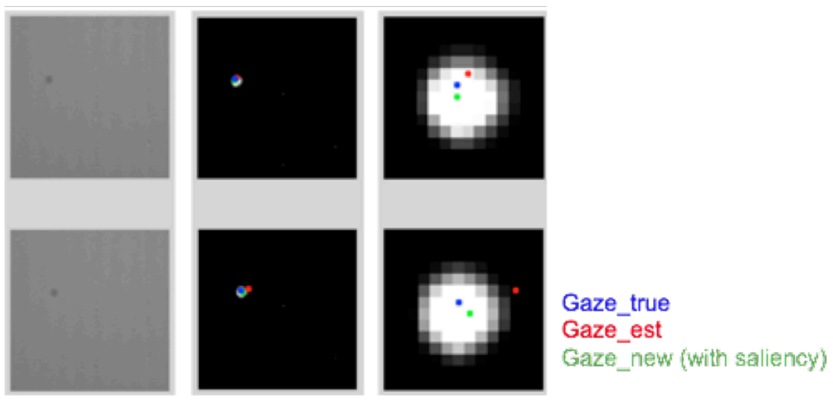
Part 2: Improving Gaze Estimation by Using Intensity-based Saliency

1. Gazing moving circle on screen

In this experiment, we attempted to detect the gazing position when looking at a small moving circle on computer screen. The pictures of screen were taken by the world-facing camera, then the centers of the circles were detected and considered as the true positions (blue points) that human eyes were gazing at. The pictures of human eyes were taken by the eye-facing camera and were used as input data for trained neural model. Gazing positions (red points) were estimated by the neural model. This iShadow data included 3469 grey scale images with 112 x 111 resolution.

As mentioned in method section, we constructed Gaussian filter with the center at the estimated position, then applied to the saliency map and selected the highest saliency value points as the new revised gazing positions (green points). We trained 150 moving-circle images to get the best sigma (standard deviation) value of Gaussian filter. The criteria for finding the best sigma is to have the new revised gazing position (saliency-maxima point) as close to the true point as possible. Such sigma value is exhaustively searched using 150 moving-circle images. We then used the best sigma on 100 different images and compared the new gazing points with iShadow's original estimated positions.

The following figure shows that the new gaze after using saliency (green) is closer to the true gaze position in the image (blue), compared to iShadow's gaze estimate (red). One thing to note is that the true gaze position provided by Addison is not very precise. As shown in the figure below, the true gaze positions are not close to the center of the circle while the new gaze point is closer to the center. This could cause some imprecision in the error calculation and finding the best sigma.



(a) (b) (c)

Figure 11. (a) Image of moving-circle on computer screen (b) Intensity-based saliency map and true gaze point, iShadow's gaze estimate and new gaze estimate after using saliency information (c) Zoomed-in version of figure(b)

Table 1 shows the average errors of neural model estimated results and new revised results, figure 12 is the distribution of errors. The unit of distance error was pixel. The best sigma value for gaussian kernel is 5. According to the results in table 1, applying saliency reduces the average error of gazing position estimation from 2.48 to 1.00. The standard deviation of gaze errors is lower, which means that gaze error is more stable. Therefore, we conclude that the intensity-based saliency map improves iShadow's gaze estimation. One observation is that the error distribution for the new gaze estimate is sparse, which means the error values are similar most of the time. In the figure 12, there are only 7 histogram bars in the error distribution of new gaze estimate while there are much more histogram bars in the error distribution of iShadow's original gaze estimate. This could occur possibly because a certain cluster of iShadow's gaze coordinates are shifted to the same saliency-maxima point, which is the center of the circle most of the time.

Table 1: Comparing error of original estimated result and new revised result of moving circle images

	error average	errors standard deviation
Original estimated result	2.48	1.63
New revised result	1.00	0.58

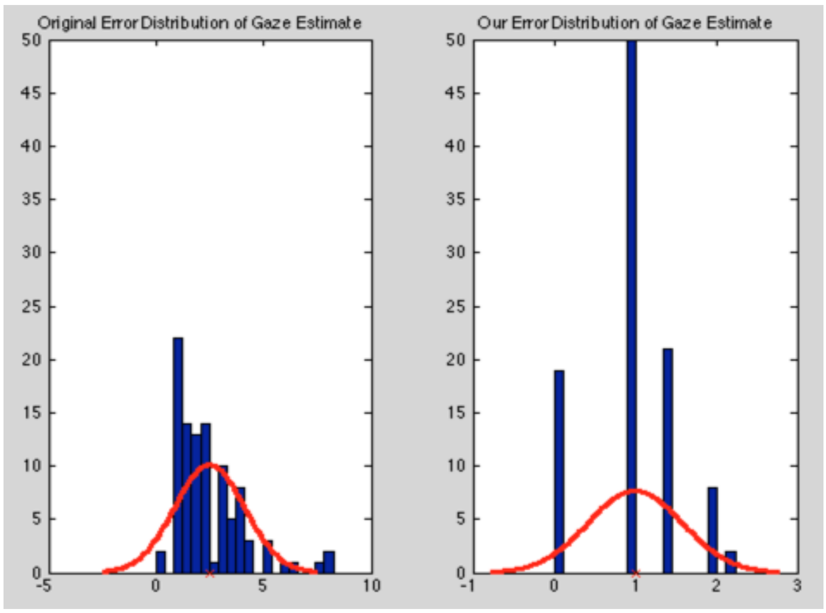


Figure 12: Error distribution of moving circle images for iShadow's original gaze estimates (left) and for the new gaze estimates after using intensity-based saliency

2. Complex images

In this experiment, we attempted to detect the gazing position when looking at different complex pictures on computer screen. Due to the time constraint and technical difficulties with glasses, we did not have a chance to run this experiment using iShadow glasses. Therefore, we decided to create a synthetic dataset and generating saliency map with synthetic gaze estimate by using the gaze estimate points from moving-dot dataset.

The process is finding the true gaze points in the original iShadow's moving-circle dataset that are close to the high saliency region on our new image (in this case, balloon image) within a selected distance. If such true gaze points exist in the moving-dot dataset (blue points), we take the responding gaze estimated points (red points) from the moving-dot dataset and use it in our balloon image. The main assumption in our experiment is that people tend to look at the regions where the saliency is reasonably high. This is the reason why we selected the true gaze points that are close to the high saliency regions in the image. In this way, we assume that if we had the glasses that works properly, when people look at the areas with high saliency values, iShadow's neural network will provide the estimated points that are close to the estimated points from iShadow dataset (since original neural network works independently on gazing images). The data is not authentic, but good enough for current experiments.

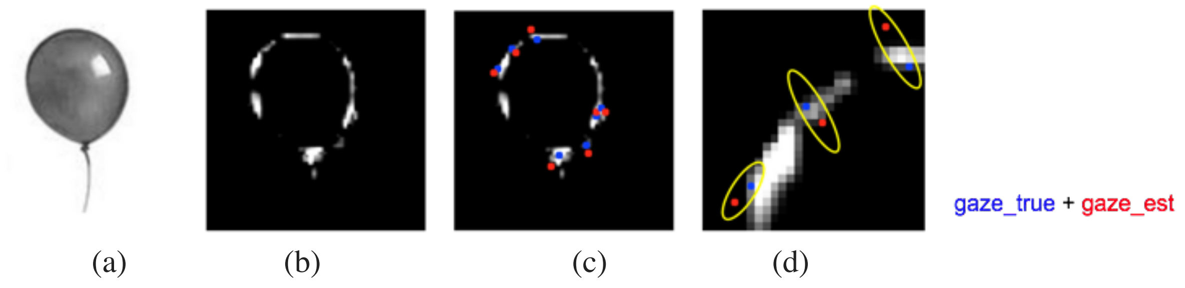


Figure 13 (a) Original grayscale image (b) Intensity-based saliency map at 1024x1024 resolution (c) true gaze points and gaze estimate points that are selected from iShadow's moving-dot dataset (d) Zoomed-in saliency map with true gaze points and gaze estimate points

The following figure shows the 3 scenarios in which (1) saliency helps improving the accuracy of gaze estimation (2) saliency reduces the accuracy of the gaze estimation (3) saliency does not make a difference in improving the accuracy of gaze estimation. According to the result below, the gaze estimation improves only if the true point is close to the high-saliency region relative to the iShadow's gaze point, which can be seen in figure 14(a). However, in figure (b) and (c) , true gaze points are not necessarily in the region with maximum saliency. When the iShadow's gaze estimation point is shifted to the saliency-maxima point, the resultant gaze point is either less accurate than iShadow's original gaze point or the same.

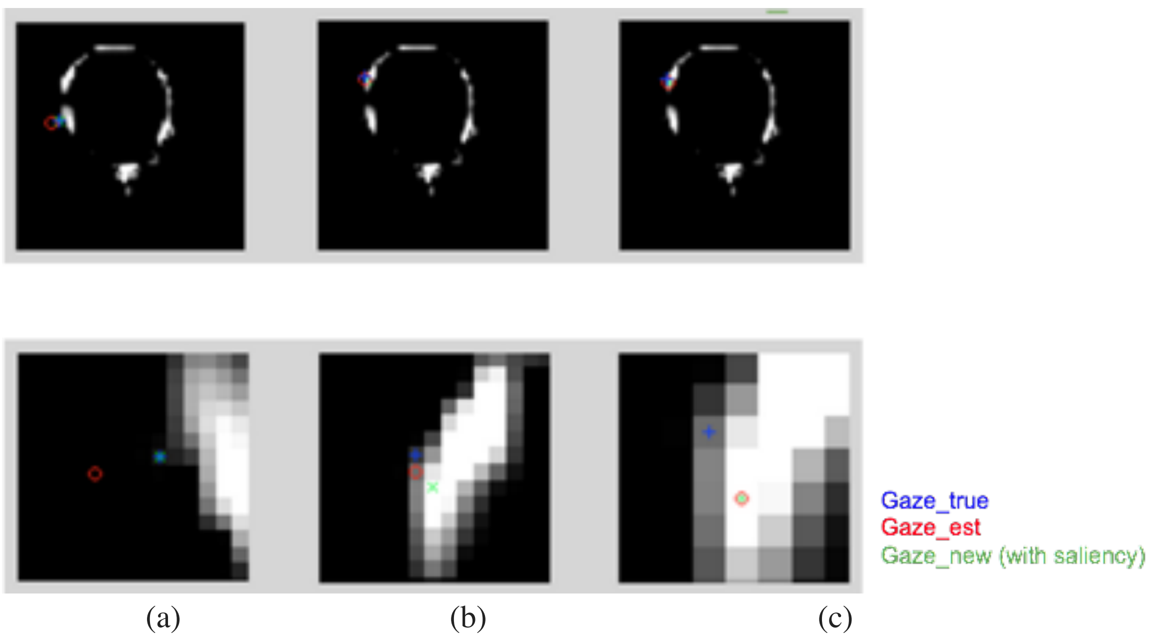


Figure 14. Saliency map with true gaze, iShadow's gaze estimate and new gaze estimate (bottom image is the zoomed-in version of the top image.) (a) true gaze coincides with new saliency gaze (b)

new saliency gaze is further away from true gaze than iShadow's gaze estimate (c) new gaze coincides with the iShadow's gaze estimate.

Table 2 shows the average errors of neural model estimated results and new revised results. Figure 6 shows the distribution of errors. Trained sigma value is 1.5. Applying saliency reduced the average error of gazing position estimation from 2.96 to 2.41. The errors standard deviation changed from 2.07 to 1.52. Unlike the moving dot image, the error does not reduce significantly for this balloon image possibly. One reason could be that for complex images, the saliency does not improve much for gaze estimate that are already close to the true points within a few pixels. In addition, there are several true gaze points that are not necessarily in the high-saliency region as shown in figure 14(b) and (c). As a future work, the fact that whether humans tend to look at points close to high saliency regions more could be confirmed by performing experiment using iShadow glasses and a human participant to ask to look at different images and collect authentic data.

Table 2: Comparing error of original estimated result and new revised result on a single balloon image

	error average	errors standard deviation
Original estimated result	2.96	2.07
New revised result	2.41	1.52

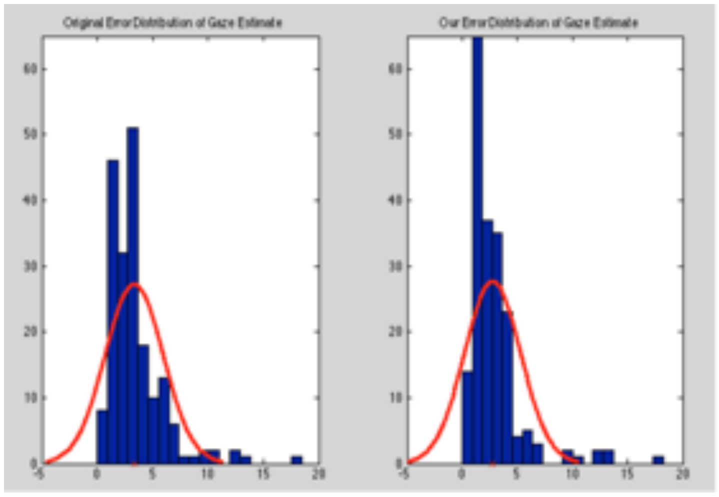


Figure 15: Error distribution of single balloon images

Similarly, the average error did not get reduced significantly for another image shown in figure 16. Below are result of outdoor scenery image that reduce error from 2.99 to 2.36.

Figure 16: Outdoor scenery image

Table 2: Comparing error of original estimated result and new revised result of scenery image

	error average	errors standard deviation
Original estimated result	2.99	1.84
New revised result	2.36	1.97

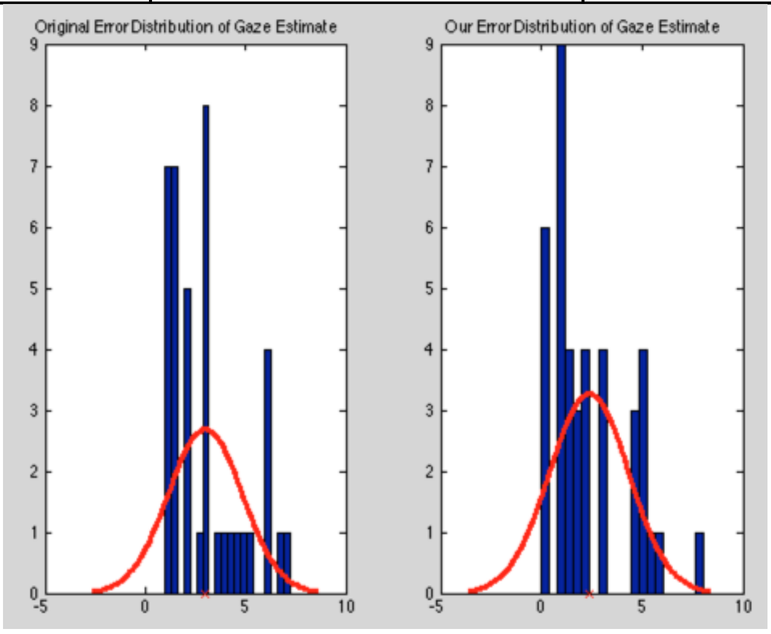


Figure 17: Error distribution of scenery images

V. CONCLUSION

In conclusion, we have used motion-and-intensity-based saliency map to improve the gaze estimate that is provided by the neural network of iShadow eye tracker. According to motion detection experiment, we concluded that motion saliency is useful in correcting gaze point. The intensity-based saliency lowers the error in gaze estimates significantly for simple image, but not for complex images. One thing to note is that iShadow's gaze estimates for this experiment are much closer to true points than the gaze estimates for the motion detection experiment. We can conclude that the saliency is useful in correcting iShadow's gaze estimate that are far off from the true point, but not for the gaze estimates that are already close enough to the true point.

VI. DELIVERABLE

The full video of the saliency map overlayed on consecutive 51 image frames that are recorded on iShadow is available at https://www.youtube.com/watch?v=_Q650MOkZs0.

VII. LIMITATIONS AND FUTURE WORK

Although color information is extremely useful in providing saliency information, the iShadow's grayscale camera do not provide color information. If the color saliency is incorporated to the existing motion-and-intensity-based saliency, the gaze estimation might be improved better. In the future, rgb camera could be used to explore color saliency. Other saliency features such as edge and corner detection and face detection could also be incorporated into the saliency map.

For future work, affine transformation method could be used to calibrate the coordinate system of both cameras on the glasses for each time when the glasses move relative to the eye positions. The affine transformation will be completed by finding a least square solution to compensate the effects of shearing, translation, rotation and scaling in the camera's coordinate system. This could be complementary way to eliminate calibration error and improve the accuracy of the gaze prediction.

The head movement requires the consecutive image frames to align them, which could be difficult to accomplish in real time. One way to tackle this would be to use the inertia sensor that is already on iShadow's circuit to detect head movement. Inertia sensor would detect movement not only when head moves but also when the user adjusts the glasses. To further differentiate glasses movement from head movement, the images from two cameras could be used. For example, if eye-facing camera shows a lot of changes (much more than pupil movement), it could be determined that the glasses moved, and otherwise, the head moved. This is assuming that glasses movement and head movement don't usually happen together.

Currently, the algorithm to improve gaze estimate runs offline on matlab. The algorithm could be simplified further to accommodate the iShadow glasses framework for real-time gaze tracking.

VIII. REFERENCE

1. Mayberry, A. 2014. iShadow: design of a wearable, real-time mobile gaze tracker
http://people.cs.umass.edu/~dganesan/papers/MobiSys14-iShadow.pdf
2. Niebur, Ernst. "Saliency Map." - Scholarpedia. N.p., n.d. Web. 16 Dec. 2014.
http://www.scholarpedia.org/article/Saliency_map

3. Schneider, W. and Shiffrin, R. M. (1977). "Controlled and automatic human information processing: I. detection, search, and attention". Psychological Review 84 (1): 1–66. doi:10.1037/0033-295x.84.1.1.
4. Walther, Dirk B. "Saliency Toolbox." Saliency Toolbox. California Institute of Technology, n.d. Web. 16 Dec. 2014.
http://www.saliencytoolbox.net/
5. "Matlab Code for Background Subtraction." Pantech Blog. N.p., n.d. Web. 16 Dec. 2014.
https://www.pantechsolutions.net/blog/matlab-code-for-background-subtraction/
6. "MIT Saliency Benchmark." MIT Saliency Benchmark. MIT, n.d. Web. 16 Dec. 2014.
http://saliency.mit.edu/
7. "Red Balloon Print: Digital Print of an Original Drawing Available 5x7" or 8x10"" Etsy. N.p., n.d. Web. 16 Dec. 2014.
https://www.etsy.com/listing/191918584/red-balloon-print-digital-print-of-an?ref=market
8. Meur, Le. "Predicting Saliency Using Two Contextual Priors: The Dominant Depth and the Horizon Line." Olivier Le Meur. N.p., n.d. Web. 16 Dec. 2014.
http://people.irisa.fr/Olivier.Le_Meur/publi/2010_icme/

IX. CODE SNIPPETS

Get saliency map

SaliencyToolbox return saliency map with resolution as about of original image, so we should first enlarge the image then resize back to original resolution.

```
%------------------------------------------------------------------------------------------------------------------------
function [ salmap,im_size ] = getSaliency(rgb_im)
[h w d] = size(rgb_im);
im = imresize(rgb_im,[h*5 w*5]);
imwrite(im,'tmp_image.jpg');
img_path = 'tmp_image.jpg';
img = initializeImage(img_path);
params = defaultSaliencyParams_nocolor;
salmap = makeSaliencyMap(img,params); %using SaliencyToolbox function
salmap = imresize(salmap.data,[h h]);
%------------------------------------------------------------------------------------------------------------------------
```

```
%-------Constructing Gaussian filter and get highest saliency position------------------------------------
function [x,y]=getMaxPosition(salmap,sigma,gaze_hat)
[h,w]=size(salmap);
gaze_hatX=gaze_hat(1);
gaze_hatY=gaze_hat(2);
maxWidth=max((w-gaze_hatX),gaze_hatX);
maxHeight=max((h-gaze_hatY),gaze_hatY);
hsize=2*(max(maxHeight,maxWidth));
gfilter = imresize(fspecial('gaussian',hsize+1,sigma),2*[ maxHeight maxWidth]);
[gh, gw]=size(gfilter);
%Bottom right corner patch
if gaze_hatX>=(w-gaze_hatX) && gaze_hatY>=(h-gaze_hatY)
    croppedfil=gfilter(1:h,1:w);
    %Top right corner
elseif gaze_hatX>=(w-gaze_hatX) && gaze_hatY<(h-gaze_hatY)
    croppedfil=gfilter((gh-h+1):gh, 1:w);
    %Top left corner
elseif gaze_hatX<(w-gaze_hatX) && gaze_hatY<(h-gaze_hatY)
    croppedfil=gfilter((gh-h+1):gh,(gw-w+1):gw);
    %Bottom left corner
else% gaze_hatX<(w-gaze_hatX) && gaze_hatY>=(h-gaze_hatY)
    croppedfil=gfilter(1:h,(gw-w+1):gw);
end
scoreMat = salmap.* croppedfil;
BW=logical(scoreMat==max(scoreMat(:)));
stat=regionprops(true(size(BW)), BW,  'WeightedCentroid');
x=round(stat.WeightedCentroid(1));
y=round(stat.WeightedCentroid(2));
%-------------------------------------------------------------------------------------------------------------------




%---------Trainning sigma value--------------------------------------------------------------------------------
function [ bestSigma ] = bestSigma( salmaps, gaze_points, true_points, num_of_train )
sigma_range = 1:0.5:100;
average = arrayfun(@(sigma) dist_cost(salmaps, sigma, gaze_points, ...
    true_points, num_of_train), sigma_range);
bestSigma = sigma_range(find(average==min(average)));
end
%-------------------------------------------------------------------------------------------------------------------

%--------------Distance code----------------------------------------------------------------------------------------
function [ dist_cost ] = dist_cost(salmaps, sigma, gaze_points, true_points, num_of_train )
distances = zeros(size(true_points,1),1); %n by 1 matrix
for i=1:num_of_train,
    true_point = true_points(i,:);
    true_point_x = true_point(1,1);
    true_point_y = true_point(1,2);
    [max_x, max_y] = getMaxPosition(salmaps{i},sigma,gaze_points(i,:));
    distances(i,1) = sqrt((max_x-true_point_x).^2 + (max_y-true_point_y).^2);
end
dist_cost = mean(distances);
end
%-------------------------------------------------------------------------------------------------------------------



%--------Generating simulated data-----------------------------------------------------------------------------
% read images, get saliency map, find local maxima, finding responding
% points from data set, get gaze_true and gaze_est data, save to mat files

% Loading data
data = importdata('y1sm4.mat');
% Loading images
allImage = data.Out;

% Loading gazed positions
gaze_est = round(data.gaze_est);
gaze_est = [gaze_est(:,1), gaze_est(:,2)];
```

```
%Loading true positions
gaze_true = round(data.gaze_true);
gaze_true = [gaze_true(:,1),gaze_true(:,2)];
new_folder = 'img_new/';
data_folder = 'img_data/';
srcFiles = dir('img_new/*');
sze = 2*1+1;
threshold = 0.1;
for j = 4 : length(srcFiles)
    filename = strcat(new_folder,srcFiles(j).name);
    I = imread(filename);
    [salmap,im_size] = getSaliency(I);
    mx = ordfilt2(salmap,sze^2,ones(sze));
    convolved = ((salmap(:,:)>threshold) & ...
        salmap(:,:)==mx(:,:));
    [c,r] = find(convolved);
    est = [];
    true = [];
    hold on;
    num = 0;
    for i=1:size(r,1)
        row = find(ismember(gaze_true,[r(i),c(i)],'rows'));
        if size(row,1) == 0
            row = find(ismember(gaze_true,[r(i)-1,c(i)-1],'rows'));
        end
        if size(row,1) == 0
            row = find(ismember(gaze_true,[r(i)+1,c(i)-1],'rows'));
        end
        if size(row,1) == 0
            row = find(ismember(gaze_true,[r(i)-1,c(i)+1],'rows'));
        end
        if size(row,1) == 0
            row = find(ismember(gaze_true,[r(i)+1,c(i)+1],'rows'));
        end
        if size(row,1) == 0
            row = find(ismember(gaze_true,[r(i)-1,c(i)],'rows'));
        end
        if size(row,1) == 0
            row = find(ismember(gaze_true,[r(i),c(i)-1],'rows'));
        end
        if size(row,1) == 0
            row = find(ismember(gaze_true,[r(i)+1,c(i)],'rows'));
        end
        if size(row,1) == 0
            row = find(ismember(gaze_true,[r(i),c(i)+1],'rows'));
        end
        if size(row,1) == 0
            continue
        end
        t = gaze_true(row,:);
        t = [t(1,1) t(1,2)];
        e = gaze_est(row,:);
        true = [true; t(1,:)];
        e = [e(1,1) e(1,2)];
        est = [est; e(1,:)];
        if mod(num,3) == 0
        scatter(t(1,1), t(1,2),'b','filled');
        scatter(e(1,1), e(1,2),'r','filled');
        end
        num = num + 1;
    end
    num;
    hold off;

    imdata.Out = I(:,:,1);
    imdata.gaze_est = est;
    imdata.gaze_true = true;
    filename = strcat(data_folder,srcFiles(j).name,'.mat');
    save(filename,'imdata');
end
%-----------------------------------------------------------------------------------------------------------------------



%-----------MotionSaliency----------------------------------------------------------------------------------------

function   [motionSal]=getMotionSaliency(Background,CurrentFrame,   plotOn,   maxShift,   maxtheta,
thres,areathres)
%citation: https://www.pantechsolutions.net/blog/matlab-code-for-background-subtraction/
method='edg';
[CurrentFrame,  predShift,  theta]  =  alignImages(Background,  CurrentFrame,  maxShift,maxtheta,
method);
CurrentFrame=(mat2gray(CurrentFrame));
Background=(mat2gray(Background));

Out=logical(abs(CurrentFrame- Background)>=thres);
%Apply Median filter to remove Noise
FilteredImage=medfilt2(Out,[5 5]);
%Boundary Label the Filtered Image
[L num]=bwlabel(FilteredImage);
STATS=regionprops(L,'all');
cc=[];
removed=0;
%Remove the noisy regions
for i=1:num
    dd=STATS(i).Area;
    if (dd < areathres)
        L(L==i)=0;
        removed = removed + 1;
        num=num-1;
    else
    end
end
[L2 num2]=bwlabel(L);

% Trace region boundaries in a binary image.
[B,L,N,A] = bwboundaries(L2);
%Display results

if plotOn
    figure;
    subplot(2,3,1),  imshow(1-L2);title('BackGround Detected');
    subplot(2,3,2),  imshow(L2);title('Blob Detected');
    hold on;
    for k=1:length(B),
        if(~sum(A(k,:)))
            boundary = B{k};
            plot(boundary(:,2), boundary(:,1), 'r','LineWidth',2);
            for l=find(A(:,k))'
                boundary = B{l};
                plot(boundary(:,2), boundary(:,1), 'g','LineWidth',2);
            end
        end
    end

end

motionSal=L2;
if num2>0
    motionSal(motionSal>0)=1;
    offset=6;
    if predShift(2)<0
        motionSal(:,end-(abs(predShift(2))+offset):end) =0;
    elseif predShift(2)>0
        motionSal(:,1:predShift(2)+offset) =0;
    end

    if predShift(1)<0
        motionSal(end-(abs(predShift(1))+offset):end,:) =0;
    elseif predShift(1)>0
        motionSal(1:predShift(1)+offset,:) =0;
    end

    if plotOn
        subplot(2,3,3),
        imshow(motionSal);
        title('Motion Saliency after eliminating borders of circshift result');
    end
    se = strel('disk',10);
    motionSal = (imclose(motionSal,se));

    if plotOn
        subplot(2,3,4),
        imshow(motionSal);
        title('Motion Saliency after doing "close" morphological operation');
    end
    Ilabel = bwlabel(motionSal);
    %unique(Ilabel)

    stat=regionprops(Ilabel,'all');
    nSeg=numel(stat);
    SegMat=zeros(nSeg,6);

    SegBW=cell(nSeg);

    if plotOn
    figure;
    end

    %Remove the noisy regions
    for x=1:nSeg
        SegMat(x,:)= [x stat(x).Area stat(x).BoundingBox];
        SegBW{x}=stat(x).Image;
        if plotOn
            SegLog=zeros(size(Ilabel));
            SegLog(Ilabel==x)=1;
            subplot(3,3,x);
            imshow(SegLog);
        end
    end

    if nSeg>0
        SegMat=sortrows(SegMat,2);
        bbox=SegMat(end,3:end);
        [h w]=size(Ilabel);
        BW=padarray(SegBW{SegMat(end,1)},[h-bbox(4) w-bbox(3)],'post');
        motionSal=circshift(BW,floor([bbox(2) bbox(1)]));
    end
    motionSal = imrotate(motionSal,-theta,'bilinear','crop'); % Try varying the angle, theta.
    motionSal=circshift(motionSal,predShift*-1);  %need to be a vector
end
end
%--------------------------------------------------------------------------------------------------------------------




%-----------ImageAlignment-------------------------------------------------------------------------------------------------------
--

function [im2Shift, predShift, thetafit] = alignImages(im1, im2, maxShift,maxtheta, method)
assert(all(maxShift > 0));
%Extract color channels
R=im1;  %Red Channel
G=im2; %Green Channel

[height,width]=size(R);
ycrop=round(height*(1/10));
xcrop=round(width*(1/10));

Rcrop=R;
Gcrop=G;

% 1. Using Edge Detection for Alignment
if method=='edg'

    R_edg = edge(Rcrop,'canny');
    G_edg = edge(Gcrop,'canny');

    %if the two color edges are correctly aligned,
    %sum(abs(R-G)) and sum(abs(R-B)) has to be close to zero. (minimum)

    minSumG=2147483647;
    for i=-maxShift(1):2:maxShift(2)
        for j=-maxShift(1):2:maxShift(2)

            for theta=-maxtheta:1:maxtheta

                    Gshift_edg=imrotate(G_edg,theta,'bilinear','crop');

                    Gshift_edg=circshift(Gshift_edg,[i j]);  %need to be a vector

                    diffG=abs(R_edg-Gshift_edg);

                    sumG=sum(diffG(:));

                    if sumG<minSumG
                        minSumG=sumG;
                        ig=i;
                        jg=j;
                        thetafit=theta;
                    end

                %end
            end
        end
    end

end

im2Shift=Gshift;
 %-----------------------------------------------------------------------------------------------------------------------
```

X. APPENDIX

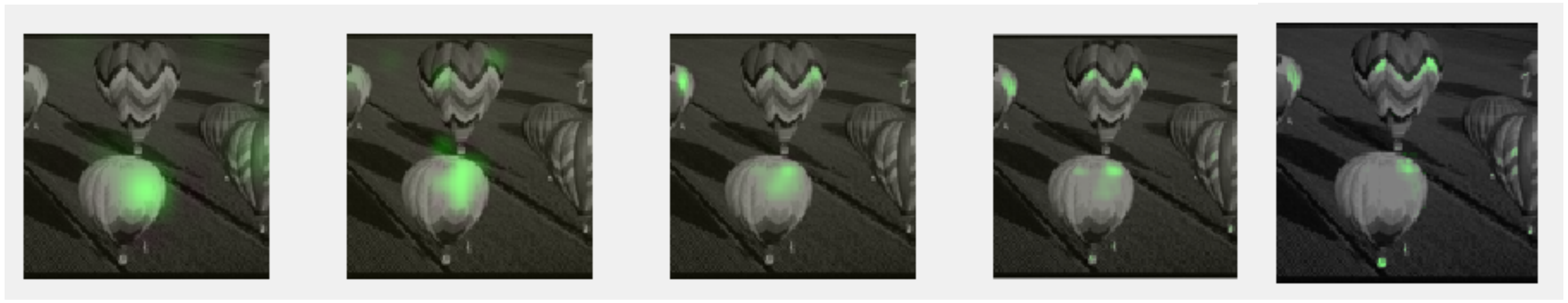


Figure 18. Intensity-based saliency map for images with different resolutions. Resolution increases by a factor of 2 from left to right

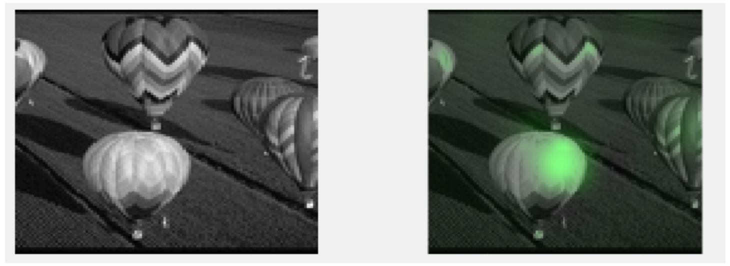


Figure 19. Final saliency map that combines saliency values at different image resolution

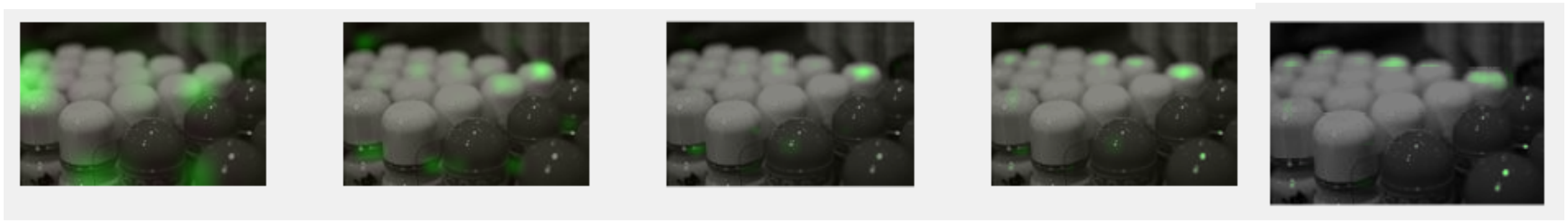


Figure Figure 20. Intensity-based saliency map for images with different resolutions. Resolution increases by a factor of 2 from left to right

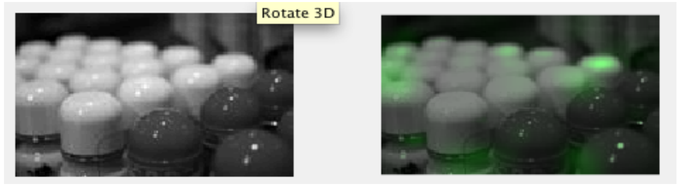


Figure 21. Final saliency map that combines saliency values at different image resolution

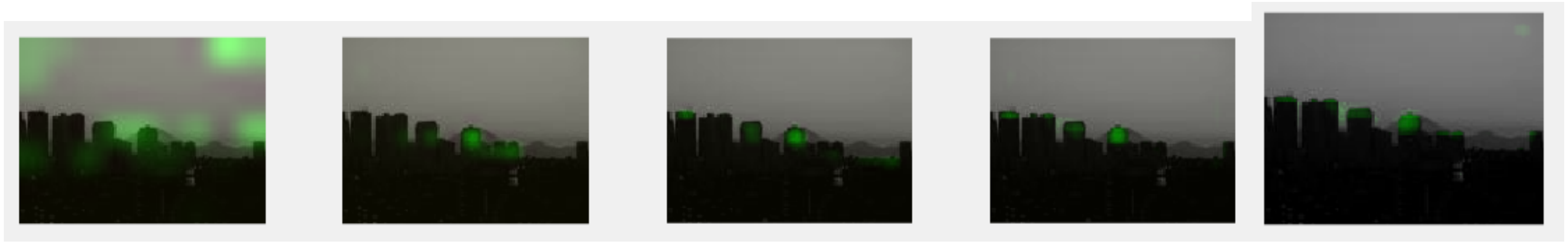


Figure 22. Intensity-based saliency map for images with different resolutions. Resolution increases by a factor of 2 from left to right

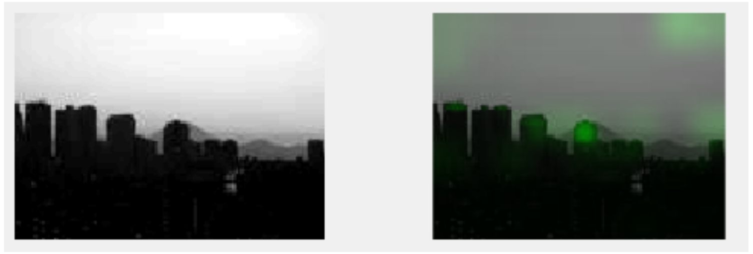


Figure 23. Final saliency map that combines saliency values at different image resolution

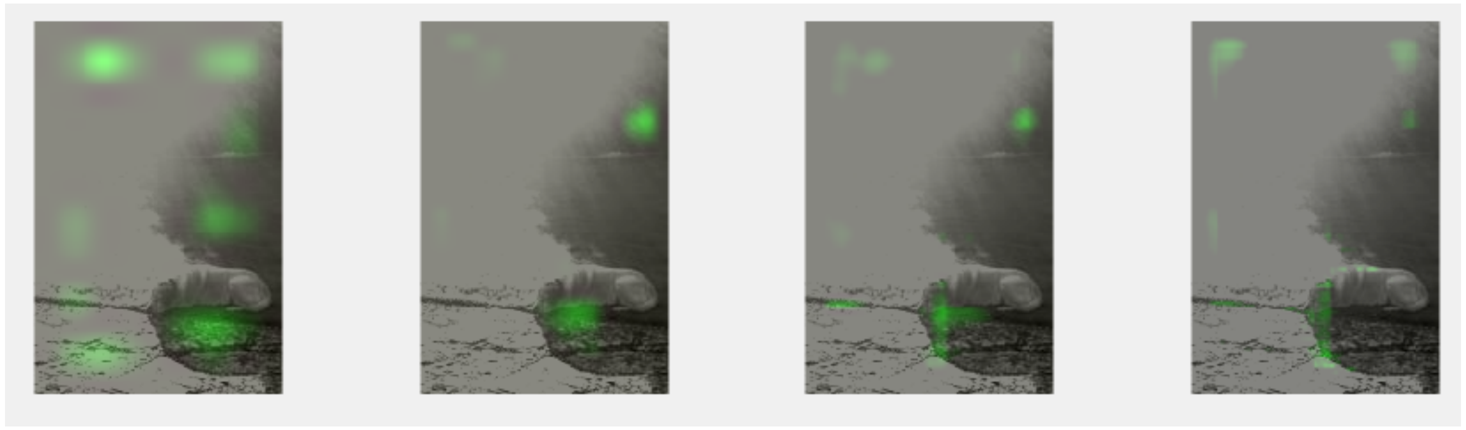


Figure 24. Intensity-based saliency map for images with different resolutions. Resolution increases by a factor of 2 from left to right

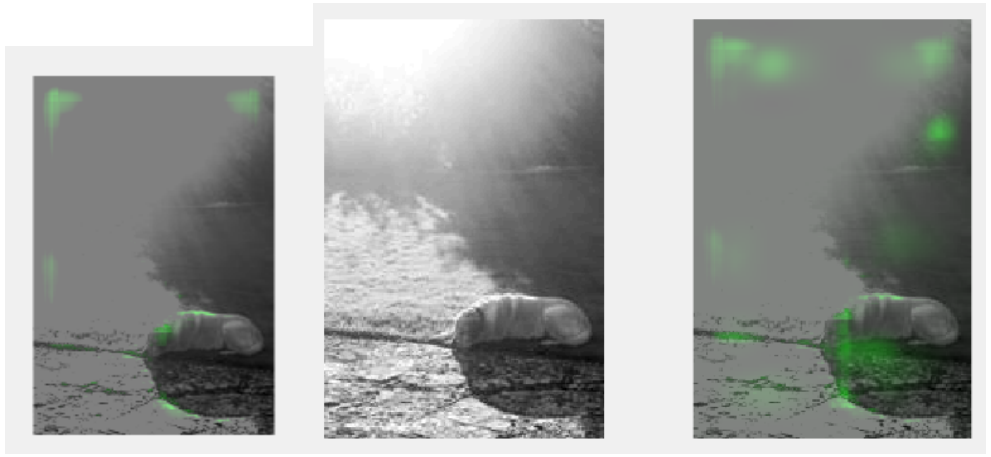


Figure 25. Final saliency map that combines saliency values at different image resolution

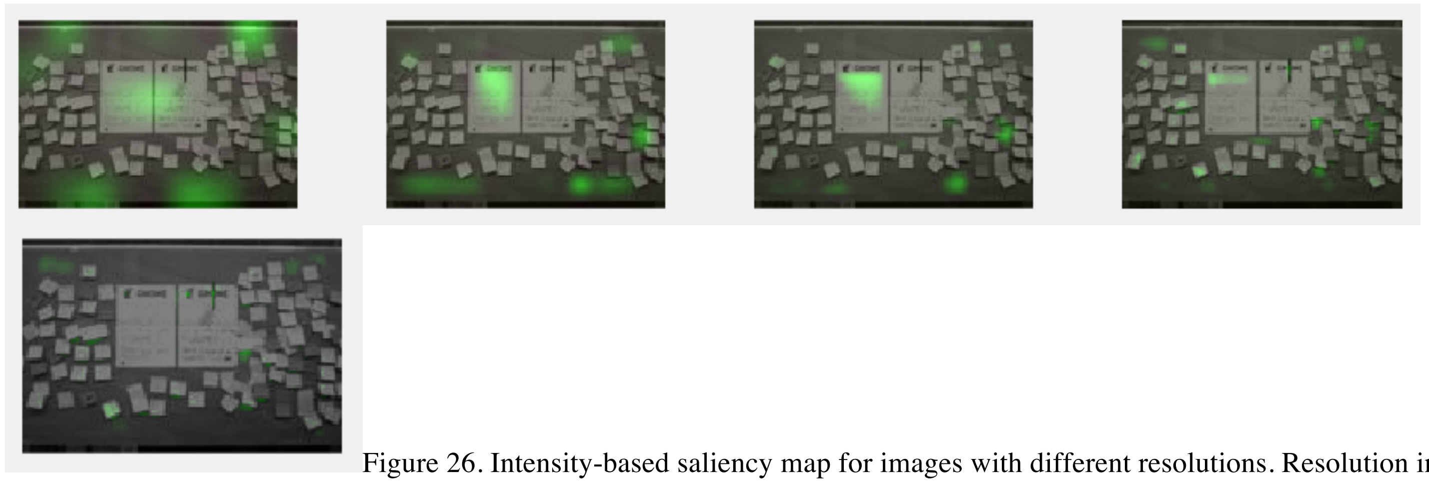


Figure 26. Intensity-based saliency map for images with different resolutions. Resolution increases by a factor of 2 from left to right

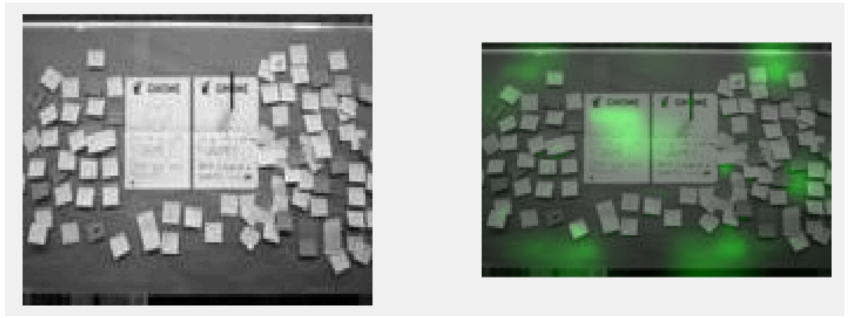


Figure 27. Final saliency map that combines saliency values at different image resolution

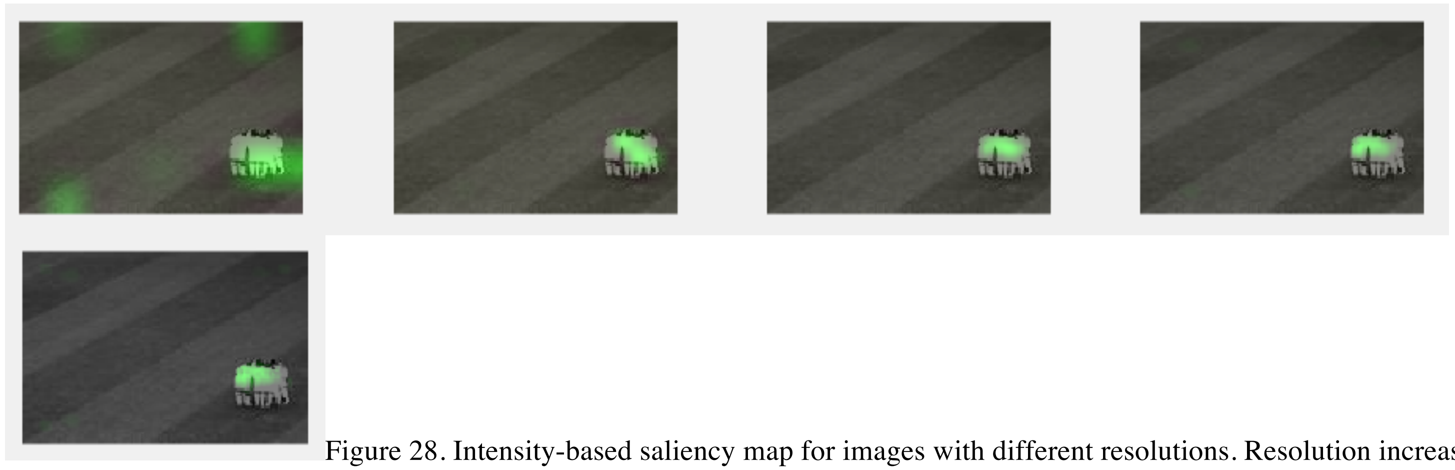


Figure 28. Intensity-based saliency map for images with different resolutions. Resolution increases by a factor of 2 from left to right

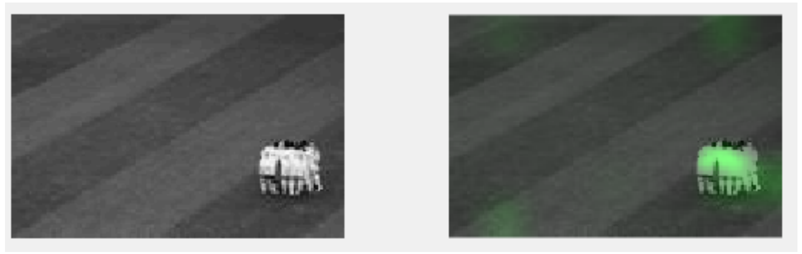


Figure 29. Final saliency map that combines saliency values at different image resolution

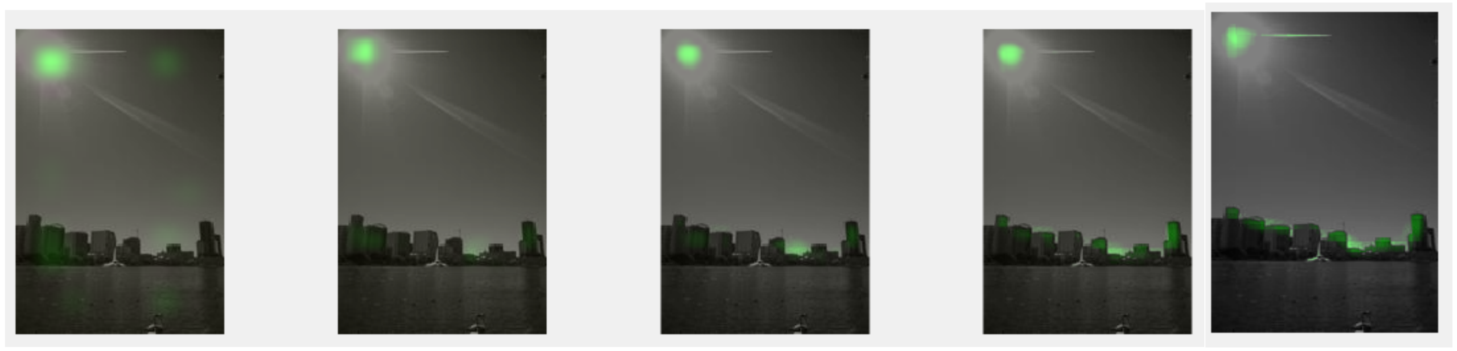


Figure 30. Intensity-based saliency map for images with different resolutions. Resolution increases by a factor of 2 from left to right

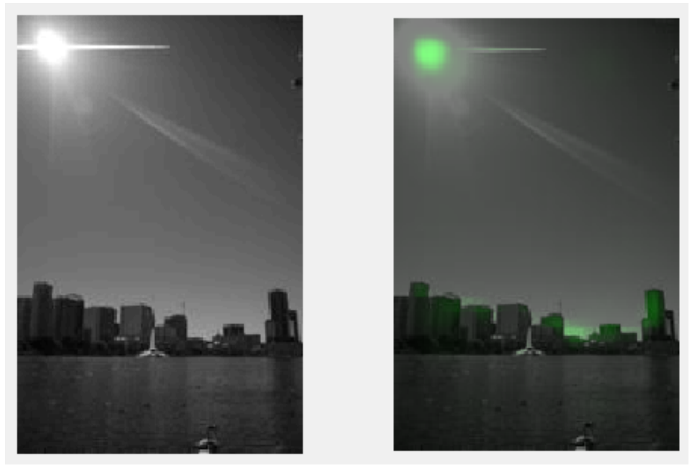


Figure 31. Final saliency map that combines saliency values at different image resolution