The Spotlight's On You!

Ryan Krawec Matthew Morrissey Rohan Chandy Kyle Liebler Ahree Hong

Abstract

We aimed to make a face tracker that keeps a spotlight on a user. We accomplished this by obtaining an initial bounding box, applying masks, and then using the mean-shift algorithm to update the bounding box. This report aims to describe our methods in approaching, implementing, and testing this project.

1. Introduction

COVID has made 2020 a strenuous year for many people. Most individuals have been isolated from loved ones and solitude has had a negative effect on the mental health of many. In a time of great melancholy and uncertainty, there is a need for being brought together without physically being brought together. That is why we built The Spotlight's On You.

Schools, workplaces, and even friends and family can lose sight of each other when they use virtual communication. Seeing the face of the person you are talking to has become extremely important in these times of social distancing. That is why we've built a system that tracks a user's face and always shines a spotlight on him or her. This way, your friend, colleague, or team member can be the only thing shining on your screen.

Our spotlight was implemented to always move with the user. We accomplished this by getting an initial bounding box of a face and then using the mean shift algorithm to alter the bounding box's location. No matter where on-screen the user goes, the spotlight will always be on them.

Similar projects have been implemented before. For example, Snapchat has software that tracks a users face and applies different styled filters to it. All the filters enhance the emotion sent over virtual communication. We were inspired by the happiness that these filters bring to so many people, and so, we created a unique filter that generates utility by simply making people's days brighter.

With the pandemic, we have to find new, creative ways to enjoy each other's virtual presence. Although our project is not directly practical, we believe increasing the number of smiles in the world is a great mission to strive for.

2. Approach

To get the initial location of the face in the video input, we used OpenCV's pre-trained Haar-Cascades classifier [1]. This model returns the location of the face in the form of a bounding box. Our initial approach was, given the initial bounding box, to use a Kalman filter to update the position of the bounding box to track the person's face as they move throughout the image. Upon further research, we decided that a mean-shift algorithm was the simpler and more efficient solution to our face tracking task.

2.1. Backprojection vs. HSV Thresholding

One key component of our algorithm was the preprocessing of the image by converting the current frame from the RGB to HSV colorspace. From here, we tried two different methods to best track the face between frames. Our first method, backprojection, began by creating a histogram of all the hue, saturation, and value pixels within the bounding box as well as the entire image. Then, we identified the region in the image whose histogram most closely matched that of the bounding box. This was accomplished by dividing the bounding box's histogram by the entire image's histogram. This yielded an image where the brighter the pixel value, the more likely it was to be the face in the image. Upon testing this method, we found that it was too noisy, and as a result, our mean shift algorithm was converging to areas other than the face. This can be seen in Figure 1.

The second method we tried, and ultimately chose, was HSV-color thresholding. This method is very simple. All it does is identify whether a pixel's hue, saturation, and value are within a certain range that covers all the possible values for a human face. This allowed us to create a fairly reliable mask that we could input into our mean shift algorithm. Similar to the previous method, the higher the pixel value, the more likely it was to be a face. This method created a much better mask for our mean shift algorithm and allowed the bounding box to converge much more reliably. This can be seen in Figure 2.

2.2. Mean Shift

We used mean-shift as our main algorithm for our face tracking. From our pre-processing step, we acquired a ma-

Figure 1. Backprojection (Left: OpenCV, Right: Ours)

Figure 2. HSV Thresholding

trix where higher values indicate pixels that were associate with a face. This, along with an initial bounding box of the face that we want to track, is everything we needed to implement our mean-shift algorithm.

The first step of the algorithm is to calculate the row and column inside our bounding box that corresponds to the mean. We accomplish this by taking the sum of all pixel values multiplied by their respective row or column. With the center of mass of the bounding box acquired, we can shift the center of our bounding box to the new center of mass.

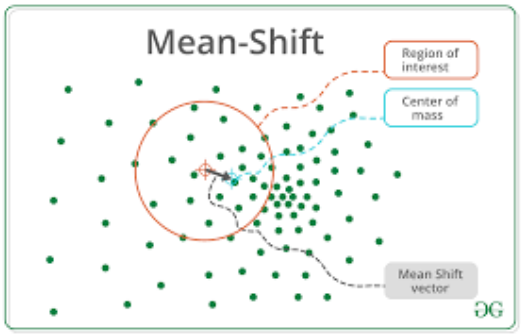

Figure 3. Meanshift [4]

As seen from Figure 3, this does not guarantee that we will be at the true center of mass of the entire frame. Because of this, we run the mean-shift algorithm ten times every frame. This allows us to successfully keep someone's face in the center of the bounding box at all times.

2.3. Spotlight Rotation

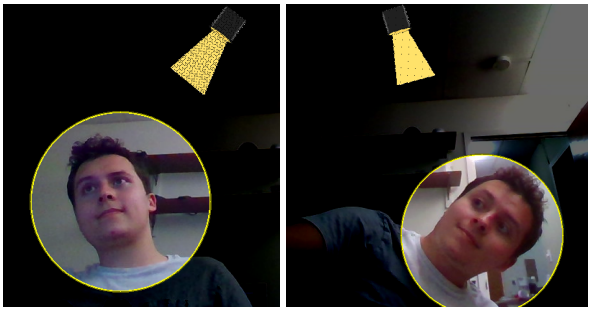

Figure 4. Spotlight

With mean-shift completed, we had to implement an image of a spotlight that rotates along with the individual. These steps are detailed below, describing the setup and continuous application of the spotlight.

During startup, it was necessary to scale and position the spotlight relative to the size and resolution of the screen. After this, we used the alpha channel in the PNG image to find out what pixels were completely transparent in the image. Then, we stored the nontransparent points in a matrix. We used these points to rotate the spotlight in the next frame.

To do this, we generated a 3x3 rotation matrix which we then applied to the set of points corresponding to the original spotlight image. For every iteration, to get the correct position, we calculate a distance and theta from the center of the spotlight to the center of the users face. Once we have these values, we can input the theta into our rotation matrix and multiply it by the indices representing our spotlight. This will correctly transform the spotlight between frames so that it is always pointing to the center of the users face.

3. Experiments

3.1. Data

To validate the proposed approach, 301 frames of a video containing a moving face are used to evaluate quantitative metrics. Live video feed is otherwise used for supplemental qualitative testing. This specific data is chosen because it resembles the typical use case of the approach and contains a reasonable amount of noise to ensure robustness.

3.2. Metrics

Success is measured qualitatively by visually assessing the accuracy of the bounding boxes and quantitatively through an intersection over union (IOU) comparison with OpenCV's implementation of mean-shift.

3.3. Results

After researching and attempting different algorithms for obtaining the features needed as input to the mean-shift algorithm, we ultimately decided to implement two different approaches. These two methods are both explained and qualitatively and quantitatively tested below. The first consists of generating a skin tone HSV thresholded mask and then calculating a histogram from the first frame of video. This is then used in backprojection for each new video frame which probabilistically determines the output features. Both OpenCV and our mean-shift algorithms are run on the output from backprojection and bounding boxes are updated for each frame. The bounding boxes in this method were jittery and had some issues tracking the face. IOU scores were not very stable, but still substantial with an average of 88.4% IOU over the whole video using OpenCV's backprojection and 95.1% using our own implementation. This suggests a possible issue in the feature-space rather than in the mean-shift implementation.

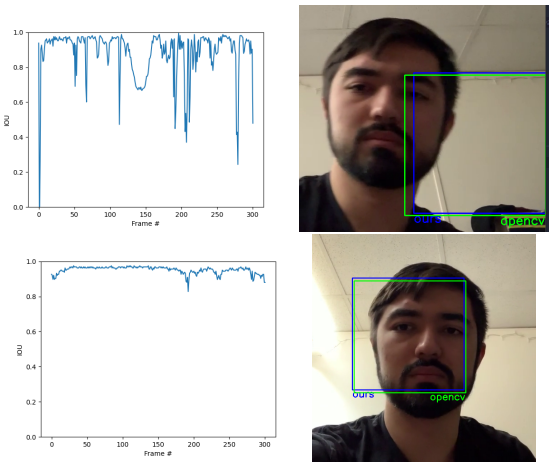

Figure 5. With Backprojection (Top: OpenCV, Bottom: Ours)

The second method involves a similar approach but removes backprojection. It was noticed first qualitatively that backprojection allowed for too much background noise when taking into account camera quality and instability. Thus, this approach generates a new HSV thresholded mask for each frame and directly uses the mask for the features being input to both OpenCV and our mean-shift algorithms. The output of this method was instantly noticed to be much smoother and tracked the face very well. It gave stable IOU scores and an average of 99.0%. A possible downside to this method is that artifacts are introduced into the masked image when skin tone colored objects are in the frame. Given properly sized bounding boxes, this should not present a significant issue.

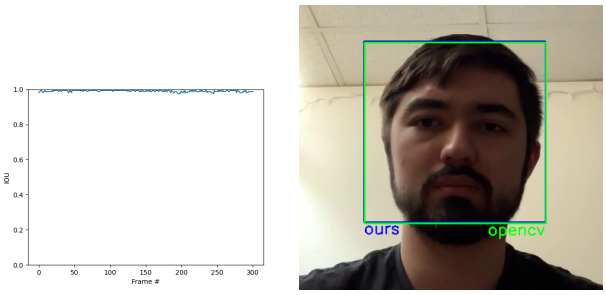

Figure 6. With HSV Thresholding

4. Implementation

We utilized OpenCV's CascadeClassifier and detectMultiScale function to get our initial bounding box around a face so that we could focus our efforts on tracking [1]. During experimentation we used OpenCV's backprojection and eventually created our own based on it [2]. We wrote our own mean-shift algorithm inspired by OpenCV's implementation and online resources [3][4]. Also, we wrote the algorithm used to rotate the spotlight from scratch.

5. Special Instructions for Special Situations

Our spotlight operates with two different modes - normal mode and party mode. Normal mode simply reduces the brightness of everything outside the spotlight. We accomplish this by simply subtracting a constant from all 3 RGB channels. Party mode cycles through different colors in differing intensities on top of doing everything that normal mode does.

References

[1] Face detection using haar cascades. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html. [Online; accessed 27-April-2021].

[2] Histogram backprojection. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_histograms/py_histogram_backprojection/py_histogram_backprojection.html. [Online; accessed 27-April-2021].

[3] Meanshift and camshift. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_meanshift/py_meanshift.html. [Online; accessed 27-April-2021].

[4] Ankur Tripathi. Mean shift clustering. https://www.geeksforgeeks.org/ml-mean-shift-clustering/. [Online; accessed 20-April-2021].