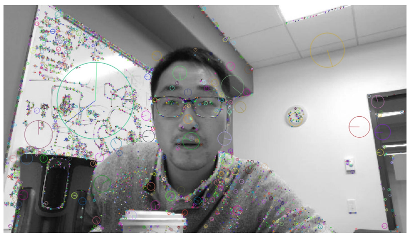
SIFT Feature Extration: Given an image (1a), we use `OpenCV` library to extract SIFT features. Image (1b) shows the keypoints with the circles of their size and their orientations.

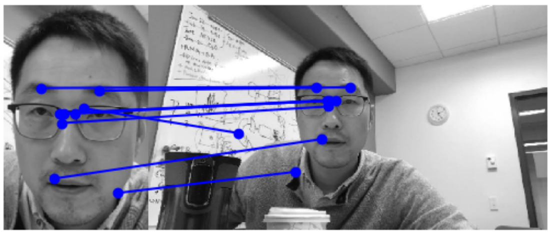
(1a) Image frame 1


(1b) SIFT

SIFT Feature Matching: We use the SIFT features to match between the template image and the target image. In specific, use two sets of keypoint descriptors / SIFT features from the template and target, find the matches using nearest neighbor with the ratio test, i.e., take the ratio of distances of first nearest neighbor and second nearest neighbor and drop those with ratio ≥ 0.7. Threshold 0.7 is chosen empirically to control the type I and type II errors simultaneously. Image (2a) and (2b) show the template and target image. Image (2c) shows the matching points between the template and the target. We can see most of the matches are correct but their are some noisy matches.

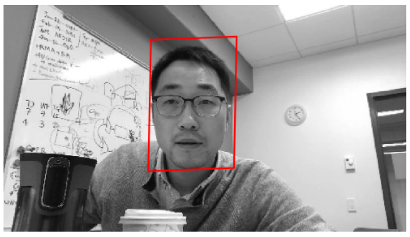
(2a) Template

(2b) Target

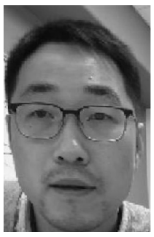

(2c) SIFT matches with ratio test

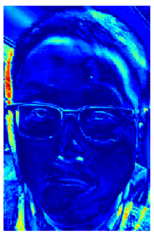
Feature-based Image Alignment: The noisy SIFT matches can be filtered by RANSAC with an affine transformation. Since affine transformation has 6 degrees of freedom, we need 3 pairs of correspondences in SIFT matches to determine it. The RANSAC procedues are as follow: We randomly sample 3 pairs of SIFT matches, determine the affine transformation, project the keypoints in template to the target domain, calculate the Euclidean distance between the projected points and matched points, and find the inliers based on a given thresholds (`ransac_thr=10`). The whole procedure is repeated `ransac_iter` = 1000 times (with the random seed set as 5561 for reproducibility). The transformation that gives the maximum number of inliers is chosen. Figure (3a) shows the projection of the template into the target domain using the RANSAC-filtered affine transformation.

Image Warping: Use the affine transform to warp the target image to the template using the inverse mapping. The inverse mapping method is applied to make sure the warped image does not produce empty pixel. For each pixel in the output image `img_warped`, project onto the domain of the input image `img` using the affine transformation `A`, then use bilinear interpolation to determine its value based on the input image `img`. We use `interpn` function imported from `scipy.interpolate` for bilinear interpolation. Figure (3c) shows the warped image. Figure (3d) shows the error map $|I_{\text{tpl}} - I_{\text{wrp}}|$.


(3a) Red: Boundary of the transformed template.

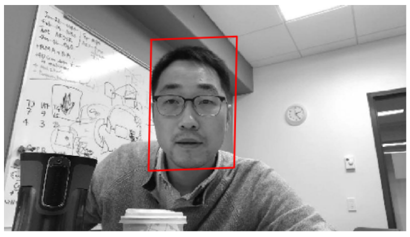
(3b) Template

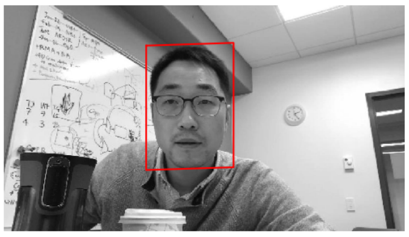

(3c) Warped image

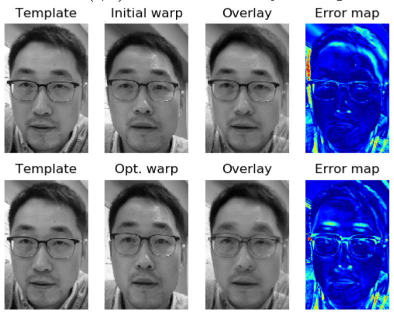

(3d) Error map

Inverse Compositional Image Alignment (ICIA): Given the initial estimate of the affine transform from the feature based image alignment, we can track the next frame image using the inverse compositional method. We reparametrize the affine tramsform and refine the affine transform using inverse compositonal image alignment. Image (4a) shows the template image `‘Hyun_Soo_template.jpg’` for the first frame image. Image (4b) shows the first frame image with the initialization of the affine transform from the feature based image alignment. Image (4c) shows the optimized affine transform using ICIA. Figure (4d) shows the comparison of the initial warp and optimized warp for frame image 1. Figure (4e) shows the error plot over iterations.

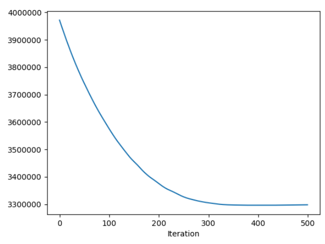
(4a) Template


(4b) Initialization for target1


(4c) ICIA for target1



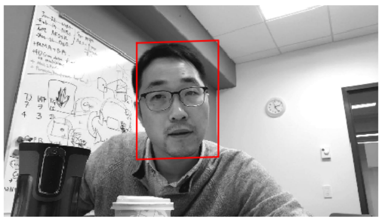
(4d) Template for target1

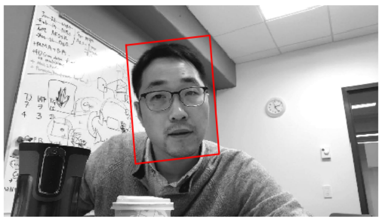


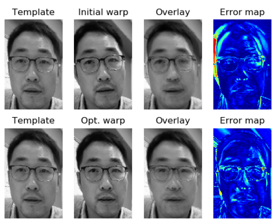
(4e) Error plot

Putting Things Together: Multiframe Tracking Given a template and a set of consecutive images, we will first initialize the affine transform using the featured based alignment and then track over frames using the inverse compositonal image alignment. The template image is updated at every frame, i.e., the ICA-warped image of frame image 1 will be used as the template for frame image 2, and so on so forth. The initial transform is set as the refined transform of the previous step.

(5a) Template


(5b) Initialization for target2


(5c) ICIA for target2



(5d) Comparison for target2

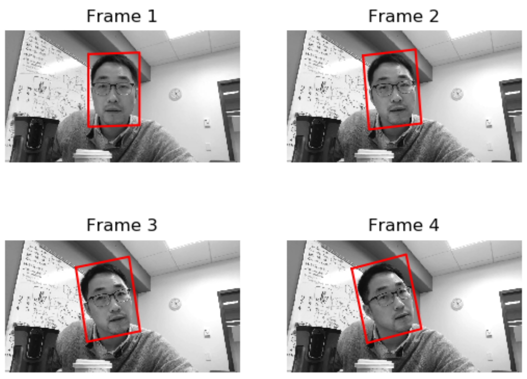


Image (5a) shows the aligned image from the frame image 1 as the template for frame image 2. Image (5b) shows the initialization of the affine transform for the frame image 2. Image (5c) shows the optimized affine transform using ICIA. Image (5d) shows the comparison between initial and optimized transform.

This left figure shows the tracking of the frames for four consecutive images using ICIA.