AUGMENTED REALITY IN LIVE VIDEO STREAMS USING POST-ITS

FINAL REPORT
JUNE 6TH, 2016

BOJIONG NI, JORIS VAN MENS
bojiong@stanford.edu, jorisvm@stanford.edu

Abstract

In this paper, we set the goal to use generic Post-it notes as fiducial markers for real-time augmented reality applications. We compare detection using SIFT, Template Matching and a custom designed "Color-Shape" method, and find that the latter approach delivers high quality results, allowing for robust and accurate projective pose estimation at low processing time.

1. Introduction

Augmented reality is an exciting experience where the virtual world meets the real. In a typical setup, virtual imagery is projected into live, real world video streams. Popular implementations of augmented reality often use fiducial markers that are specifically designed to be easily detected in video streams. In our paper, we aim to create a robust augmented reality experience that works with a colored Post-it note as fiducial marker, and which can run sufficiently fast on a single thread on a general processor.

With this approach, we hope to bring the augmented reality experience to larger audiences. A specific example would be the classroom, where teachers presenting to their students on a generic laptop could enrich their teaching with virtual experiences using only a Post-it note and our software application.

2. Review of previous work

On the specific problem of locating Post-it notes in real time, we were not able to find any literature (nor on Post-it note detection in general). There has however been extensive research focused on designing and finding fiducial markers that are easy to detect in video frames, using a variety of methods[9, 16].

Template matching is one of such commonly used methods for detecting markers in augmented reality. In Template Matching approaches, the marker (or template) tends to be carefully designed to enhance recognition and camera estimation[12].

Another frequently used method for object detection and tracking is SIFT. Previous work shows that it delivers strong results for a wide range of detection goals, yet it is a computationally expensive approach[2].

Literature on color detection has often focused on the problem of skin color detection[13]. Detection in Red Green Blue (RGB) space is common, as is detection in Hue Saturation Value (HSV) space. It is noted that a potential downside of the latter space is that it can cause hue discontinuities. In addition, the value (brightness) dimension does not relate well to human perception of brightness. An alternative Tint Saturation Lightness space is similar in nature. A YCrCb space, where Y relates to brightness and Cr and Cb relate to the color hue, is also found to exhibit favorable color detection properties for various purposes[3].

A typical, fast approach to filtering color is done by defining an explicit cuboid bounding box in the chosen three dimensional color space[5]. This method can be augmented by normalizing the image for brightness[6]. The boundaries can be found empirically by taking samples of the object in different scenes. Elliptical boundaries have also been tested[7], as have various probability-based models[17].

For edge detection, Canny edge detection is often regarded as one of the most accurate methods, with decent performance, although various other edge detection algorithms, such as Laplacian of Gaussian, may be less computationally expensive[1, 10].

For line detection, typical methods are the Radon and Hough transform. The two methods are similar in nature, where the Hough transform can be considered a discretized version of the more general Radon transform[19]. Various speed optimized Hough transform have also been proposed[18, 14, 4].

2.1. Contribution of our work.

We provide a comparison of several methods to solve the problem

of robust, accurate and fast Post-it detection for augmented reality, which had not yet been done in previous literature.

In addition, we show that it is indeed possible to use generic Post-it notes as fiducial markers for augmented reality, robustly obtaining full pose estimation at high accuracy and high speed. We do so using a combinatory approach of various low-level algorithms (color filtering, noise reduction, edge detection, line detection and several logical elements) specifically tailored to the use case, providing results superior to general methods such as SIFT and Template Matching.

3. Technical approach

Using Post-its for augmented reality imposes two important constraints. First, the Post-it is an object with very few distinguishing features, implying some general feature detection methods might not work well. Second, the speed requirement implies a further restriction on the methods being available to use, and creates a focus on minimizing execution speed. We aim to consistently render 30 frames per second, which implies the full algorithm must take less than 33 milliseconds to execute on our hardware.

We will test a manual "Color-Shape" approach, a SIFT approach and a Template Matching approach and compare their applicability to solving our problem.[1]

3.1. Experimental setup.

The hardware we use for recording and processing is a MacBook Pro 2013. The sticky notes we use are the original Post-it brand, in various color variants. For testing accuracy and speed we wave a Post-it in circular motion at 2 feet distance from the webcam and capture 100 pose estimations. We test our methods in rooms well lit by either daylight or artificial light.

3.2. Color-Shape Approach.

In the color-shape approach, we make use of a number of properties of the colored Post-it under projective transformation. These assumptions are as follows:

(1) It has 4 exact edges and vertices
(2) In the absence of radial distortion, the edges are straight
(3) Due to its square shape and small size compared to the camera distance for expected scenes, the opposing edges will be of similar size (near-affine transformation)
(4) It is a solid, non-porous object
(5) For bright colored variants, the saturation level is high compared to most surrounding scenes
(6) For various color variants (e.g. pink), the hue is uncommon in most scenes

The Color-Shape method aims to make use of all of these properties to achieve an optimal solution.

While creating this method and choosing parameters, we aim to optimize several factors. First, we aim for a high detection rate, which we define as the percentage of frames that return valid vertices (as opposed to frames that return no vertices). Second, we aim for high accuracy, defined as the percentage of detected vertices that are accurate. Third, we aim for speed, as measured in milliseconds of execution, while also keeping the standard deviation in mind. High standard deviation can result in video stutter even when mean speed is low, as a single frame with high processing time will halt the video until processing is completed.

We use a combination of color masking, binary noise reduction, edge detection, line detection and various logic steps to estimate the Post-it's location and calculate the transformation matrix. We rely on Python Opencv3.0.0 implementations of mentioned algorithms, and NumPy for other image-wide calculations, given both run optimized machine code to provide optimal performance. Figure 1 shows the main elements of the Color-Shape pipeline.

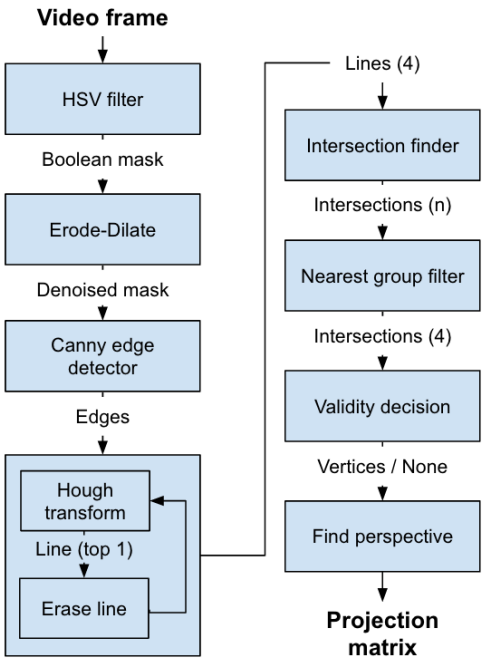


Figure 1. Color-Shape pipeline

[1]Code can be found at github.com/Bojiong/cs231a

3.2.1. *HSV filter.* We create a mask on the image by filtering for specific hue, saturation and value ranges as such:

$$HSVmask = \begin{cases} & [h_{min} < H < h_{max}] \\ 1 \quad \forall & [s_{min} < S < s_{max}] \\ & [v_{min} < V < v_{max}] \\ 0 & \text{otherwise} \end{cases}$$

To find relevant $h_{min}, h_{max}, s_{min}, s_{max}, v_{min}$ and v_{max} values, we sampled Post-it camera captures under various lighting conditions (figure 2), and captured their mean HSV values (figure 3).

FIGURE 2. Post-it color samples under various lighting conditions

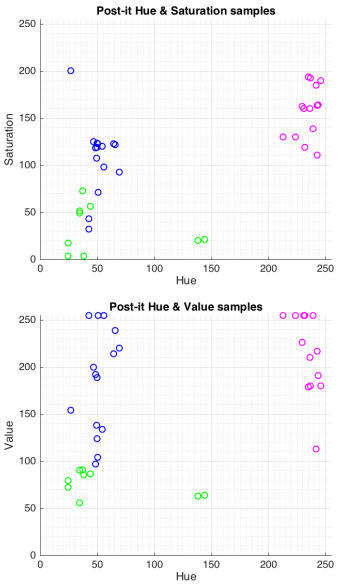


FIGURE 3. Mean hue-saturation (left) and hue-value (right) distribution of samples. Green markers correspond to pale yellow Post-its, blue to bright yellow and pink to pink.

We take minimum and maximum values for every dimension as below (example for hue H), and apply the mask.

$$h_{min} = min(H_{samples}) - stdev(H_{samples})$$
$$h_{max} = max(H_{samples}) + stdev(H_{samples})$$

3.2.2. *Erode-Dilate.* We use erosion and subsequently dilation to remove small patches (noise and false positives) from our mask while keeping larger patches intact. For erosion, we move a pixel kernel K (10x10 matrix of ones) over the mask M (which holds all non-zero pixels). We then keep only those pixels p for which all surrounding pixels covered by the kernel at that pixel (K_p) are also part of the mask:

$$M \ominus K = \{p | K_p \subseteq M\}$$

This returns pixels surrounded by patches of ones, while removing any smaller patches for which at least one pixel covered by the kernel was zero. For dilation the process is similar, but reverse: any mask pixel for which at least one of the surrounding pixels within the kernel is one, we return one. This effectively "grows" single pixels into patches of 10x10, undoing the "shrinking" caused by the erosion. The final effect can be seen in figure 4.

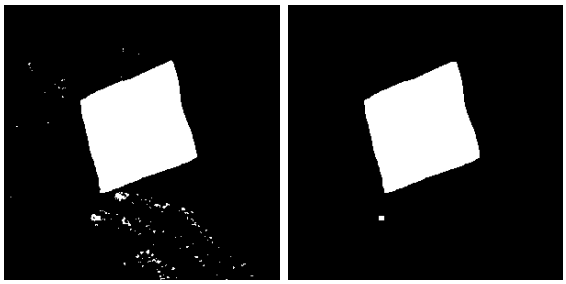


FIGURE 4. Effect of erosion and subsequent dilation (right) on noisy mask (left)

3.2.3. *Canny edge detector.* We apply the Canny edge detection algorithm to find edges in the mask. The edge detector first applies a Gaussian filter to the mask:

$$M_{ij} = \frac{1}{2\pi\sigma^2} e(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2})$$

The size of the filter is given by (2k + 1) in both x and y direction. Subsequently, it finds the gradients throughout the image as such:

$$\mathbf{S} = \sqrt{{\mathbf{S}_x}^2 + {\mathbf{S}_y}^2}$$
$$\mathbf{\Theta} = \text{atan2}\left(\mathbf{S}_y, \mathbf{S}_x\right)$$

Where S represents the size of the gradient and theta the angle. As a next step, non-maximum suppression is applied to remove multiple signals for the

same line ("thin the edges"). After this, a thresholding mechanism is applied to find the most likely true edges and remove ones more likely caused by noise.

Given the relatively simple mask provided by the previous steps, the edge detector provides good results as expected. Experimentation with the threshold values within reasonable bounds caused no significant difference in the results.

3.2.4. *Iterative Hough transform & line erase.* On the edge detected output, we will apply a Hough transform to find lines. The Hough transform translates Euclidean x-y coordinates into curves in the polar space, representing lines with different distances from the origin (r) and angles (θ). Cells in Hough space with votes above a certain threshold will be accepted as lines and converted back into x-y coordinate space, where points at extreme x-y coordinate values on the specified line will be used as endpoint estimates. Figure 5 shows the initial output.

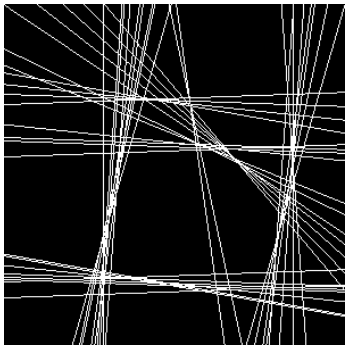


FIGURE 5. All Hough lines found

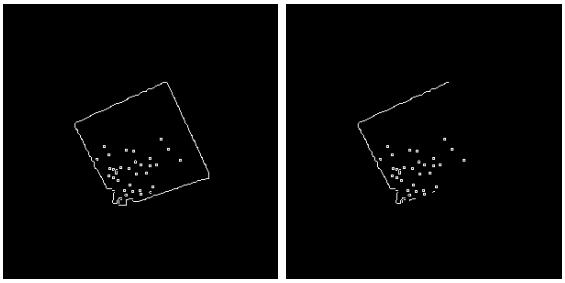


FIGURE 6. Initial edge image (left) and edge image after 2 Hough and Erase Line iterations (right)

One problem with the Hough transform is that it will fit multiple lines on a single Post-it edge. As a robust method for finding the most promising lines corresponding to the unique Post-it edges, while removing duplicate Hough matches for a single edge, we use only the highest-voted line from the Hough transform. We subsequently erase all pixels on the edge image corresponding to this line with a 3-pixel boundary radius. On the new edge image, we reapply the Hough transform. We iterate through this method 4 times. The result on the edge image after 2 iterations can be seen in figure 6. As an alternative to the iterative approach, we have also tested an approach that aims to filter out multiple line detection per Post-it edge by filtering lines with similar rho and theta values. While this gave decent results (see experiments section), we found this to be less robust than our iterative erase-line approach.

3.2.5. *Intersection finder.* To find intersections, we use the lines' polar coordinates and solve the following equation:

$$\begin{bmatrix} cos(\theta_1) & sin(\theta_1) \\ cos(\theta_2) & sin(\theta_2) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix}$$

3.3. **Nearest group filter.** Given the detected intersections, we find all intersections that lie within (or just outside of) the image. Of those, we filter for the 4 intersections in the closest group by using only the 4 intersections with minimum distance to the full group's geometric center (filtering out intersections at largest distance from the rest).

3.3.1. *Validity decision.* Given the resulting intersections, we perform several validations to verify whether or not the intersections correspond to a valid Post-it transformation. Specifically:

(1) There must be exactly 4 vertices
(2) There cannot be 3 vertices on a single line
(3) Opposing edges must be of similar length (near-affine transformation)

For the third rule, we apply a minimum to maximum line length range as such:

$$\mu = l_1 + l_2/2$$
$$l_{min} = \mu * (1 - \delta)$$
$$l_{max} = \mu * (1 + \delta)$$

3.3.2. *Find perspective.* Subsequently, we will find the projection matrix M that allows us to project pixels of the overlay image into the position of the Post-it: $P_{trans} = M * P_{overlay}$. Projective matrix M has 8 degrees of freedom, and every matching pair of points gives us 2 equations, so we can find the matrix using 4 matching points. For the $P_{overlay}$ coordinates (x & y), we use the 4 vertices of the square image we want to overlay. For the P_{trans} coordinates (x & y), we use the 4 vertices of the Post-it in the video frame. We can now solve the linear system (using Direct Linear Transformation):

$$\begin{bmatrix} t_1x'_1 & t_2x'_2 & t_3x'_3 & t_4x'_4 \\ t_1y'_1 & t_2y'_2 & t_3y'_3 & t_4y'_4 \\ t_1 & t_2 & t_3 & t_4 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (1)$$

Where M is only defined up to scale, allowing to normalize to $h_{33} = 1$.

3.3.3. *Output.* Typical frames are shown in figure 7. The top frame shows a noisy environment (with a similar-colored object in background) and the bottom frame shows occlusion of one corner. The left images show all Hough lines (thin) with the 4 chosen Hough lines plotted thick. The quadrilateral corners are identified by white circles, and a projected animation (globe) is shown on the original input on the right.[2]

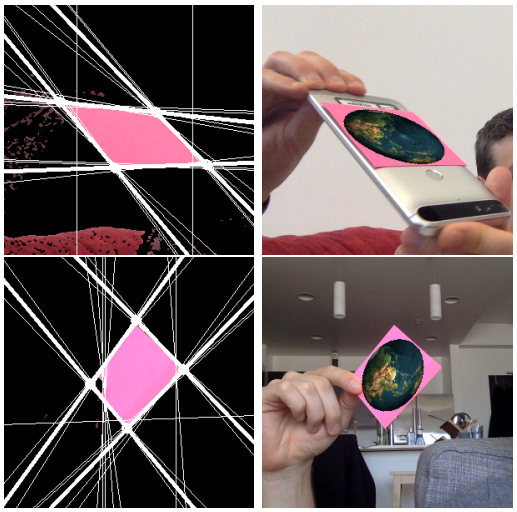


FIGURE 7. Debug images with applied mask, Hough lines and detected vertices (left) and original captures with projected overlay (right)

3.4. **SIFT approach.** Scale-invariant feature transform (or SIFT)[8] uses techniques of Difference of Gaussians, scale-space pyramid and orientation assignments to ensure the features are scale and rotation invariant. It also resamples the local image orientation planes in order to achieve full affine invariance.

3.4.1. *Find matching key points.* As SIFT is frequently used for object detection, it is worth trying to see how SIFT applies to our use case. In order to use SIFT, we first need a reference image and extract local features from it. Then we extract the SIFT features from each video frame. SIFT features are scale, rotation and affine invariant[8], thus we do not need to explicitly account for the transformation between reference camera frame and video camera frame.

Once we have the features in both reference image and the video frame, we match the features in the two images using a K-Nearest Neighbors approach. Here we will find the closest two matching points for each feature in reference. Each of the two matching points will have a distance value indicating how close it is to the corresponding feature in the reference image. To eliminate false positives, we will only accept a match if the closest distance is 70% or less than that of the second closest point. The rationale behind this is that for a true match, the second closest matching point's distance will be much larger than the closest (true) one's. When there is a false positive, various closely matching points may have similar distances.

We used the Python Opencv3.0.0 2D feature library to extract features from images and find the closest neighbors. We have experimented with both colored images and gray scale images for detection. For the reference picture, we tested with plain Post-its, Post-its with patterns drawn on them, and a rectangle directly drawn from an array. We will compare the results in the experiments section.

3.4.2. *Find transformation matrix.* After the key points are matched, we estimated the transformation of the Post-it in video from the original feature location and the matched feature location. Since we know that SIFT is affine invariant, we can assume the matched points are the source and destination of an affine transformation. This transformation matrix will later be used for transforming the animation to be overlay on the Post-it. Let (x, y) be the key point in the original template, (x', y') be the matching key point in the video frame. We have the following relation:

$$\begin{bmatrix} tx' \\ ty' \\ t \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

The transformation matrix has 6 unknowns (defined up to scale) and each matched key points will give one two equation, we need at least 3 matching points pairs to solve the problem. In case the system is over ranked (more than 3 pairs of independent key

[2] A video example can be found at youtu.be/f16gHGPc3wE

points are found) or rank deficient, we can use least square fitting to find the estimate.

3.5. **Template Matching approach.** In Template Matching, a small image is used as a template to see if a matching version can be detected in the larger image. This approach takes the template as the convolution mask and perform a convolution with the search image, sliding a window of the same size of the template over the search image. Then we compare the pixel intensity difference between the search image in the window and the corresponding pixel in template. We sum the difference over the window. The window with lowest difference sum gives the best match.

3.5.1. *Find the best matching window.* Let us give a formal definition[15]. Suppose coordinates (x_s, y_s) at the search image has intensity $I_s(x_s, y_s)$ and the coordinates (x_t, y_t) at the template has intensity $I_t(x_t, y_t)$. Define the absolute difference in the pixel intensities as $Diff(x_s, y_s, x_t, y_t) = |I_s(x_s, y_s)I_t(x_t, y_t)|$.

Define *Sum of absolute differences* measure as

$$SAD(x, y) = \sum_{i=0}^{T_{row}} \sum_{j=0}^{T_{col}} Diff(x + i, y + j, i, j)$$

We loop over the entire search image and calculate the corresponding $SAD(x, y)$. The pixel with lowest SAD is the best match.

Note that Template Matching is not scale invariant. I.e. we do not know how big the object (Post-it) is in the search image (video frame). We will resize and rotate the original template to create a series of templates with different sizes and rotations. Then we perform a search of all those templates in the search frame and return the best match.

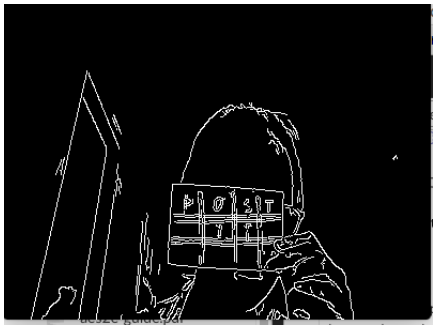


FIGURE 8. Edge image for Template Matching

In order to reduce noise and improve speed, we transform the image from RGB to gray scale and performed Canny edge detection to only work on the edges of the image (see figure 8).

We use the Opencv Python library for Template Matching. In the Opencv library, the sum of differences can be calculated in different ways, and we will compare results.

3.5.2. *Find the rotation and scaling matrix.* In this case, we can only find the rotation and scaling of the matched window instead of the full perspective transformation. As we iterate through different scale s and rotation angle θ, the corresponding transformation matrix is:

$$P' = sR(\theta)P + p$$

where p is the location of the left upper corner of matched window box.

3.6. **Overlay.** Once we have the applicable homography from either of above methods, we apply it to consecutive frames of an animation to overlay the animation into the video (as described in the Color-Shape section). We remove pixels with zero alpha values to allow for basic transparency (allowing us to overlay e.g. a spherical globe instead of only rectangular pictures).

4. Experiments and Results

4.1. Experiments using Color-Shape.

4.1.1. *Hough transform parameters.* To find the best rho, theta and threshold values for the Hough transform, we have done various experiments as shown in table 1.

Threshold	15	30	15
Rho	1	1	2
Theta	$\pi/45$	$\pi/45$	$\pi/45$
Speed (μ, ms)	4	1.8	3.6
Speed (σ, ms)	3.2	1.2	2.4
Accuracy	High	High	Mid
Detection rate	High	Low	High

Threshold	15	15	15
Rho	5	1	1
Theta	$\pi/45$	$\pi/90$	$\pi/22.5$
Speed (μ, ms)	3.6	6.1	3.7
Speed (σ, ms)	2.8	5.3	2.5
Accuracy	Mid	High	Mid
Detection rate	Mid	High	High

TABLE 1. Hough transform parameter results

Per these results, we have chosen 1, $\pi/45$ and 15 as our respective optimal parameters.

4.1.2. *Speed of Color-Shape.* An analysis of execution time for the various elements of the Color-Shape method can be seen in figure 9. Total execution time is well within our target range (below 33 ms) at 12 ms mean.

We find that the iterative Hough transform takes most computation time. This triggered us to also design a non-iterative method for finding the four most promising lines using only one Hough transform. This alternative method filters lines with similar sigma and rho (as described in the Hough transform section of the technical approach). While the method saved 3 ms of mean execution time, we found it to be less robust than the iterative approach (0.91 accuracy vs. 0.98 baseline). Given total execution time is already low, we decide to trade execution time for higher robustness.

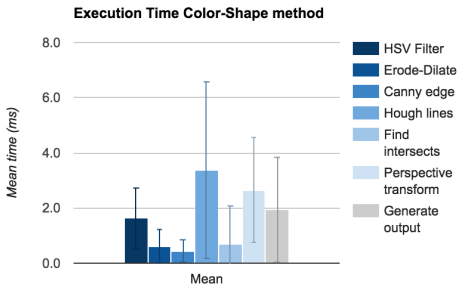
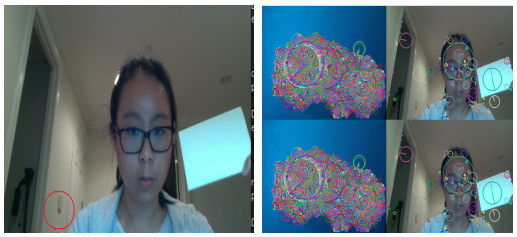


FIGURE 9. Color-Shape element-wise execution time

4.2. **Experiments using SIFT.**

4.2.1. *Colored reference with colored video frame.* In this approach, we take a picture of the Post-it, detect the features using SIFT and try to match the key points in video frame.

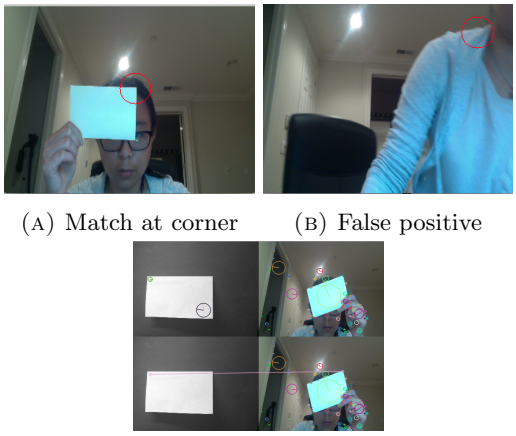


(A) False positive (B) Key points match

FIGURE 10. Colored image with colored reference

We can see from figure 10 that the colored Post-it picture contains too many features and key points, making the matching process slow and noisy. The Post-it is detected inaccurately.

4.2.2. *Gray scale reference and colored video frame.* In this experiment, we use a gray scale plain Post-it as reference. Different from the above, the key points in the reference image are greatly reduced and the matching is more accurate. However, the detection is always at the corner of the Post-it and the results still occasionally contain false positives. The result can be seen in Figure 11.

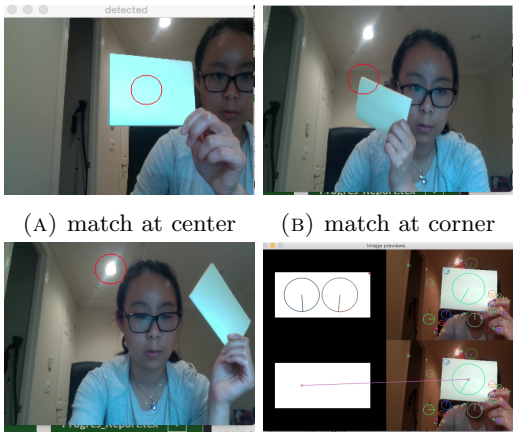


(A) Match at corner (B) False positive

(C) Key point matching

FIGURE 11. Colored image with gray Post-it

4.2.3. *Binary rectangle and colored video frame.* In this experiment, we use a rectangle drawn from an array as reference image, and use colored video frames for detection.



(A) match at center (B) match at corner

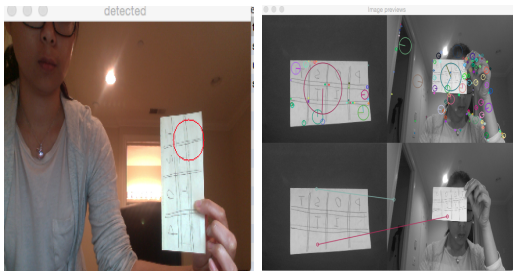
(C) false positive (D) key point matching

FIGURE 12. colored image with rectangle

The value within the rectangle is 255 and 0 outside. This time, the features and key points from the reference image are far fewer. From the results in Figure

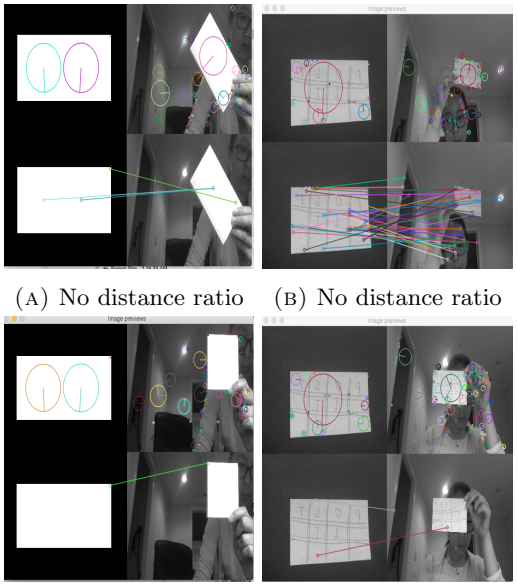
12, we can see that it can successfully find matches between the reference rectangle and the video frame. However, we still see some false positives.

4.2.4. *Post-it with pattern.* In this approach, instead of using a plain Post-it, we use a Post-it with pattern drawn. For simplicity, we will use gray scale for both reference and video frames. Figure 13 shows a result. Here the matching key points are within in the Post-it instead of at the corners as in previous cases. The lack of features for plain Post-its gives little interesting matching key points within the center of the Post-it. A patterned Post-it, however, has richer content and gradient variation within the area of Post-it, making it easier to match local feature inside.



(A) False positive (B) Key point match

FIGURE 13. Post-it with pattern



(A) No distance ratio (B) No distance ratio

(C) With distance ratio (D) With distance ratio

FIGURE 14. Distance Ratio Effect

4.2.5. *KNN distance ratio effect.* We experiment with two different schemes for KNN matching. The first is to find the nearest neighbor for key points and directly take it as a matching. The second is the method described above where we only accept a match if the closest distance is within 70% of the second closest distance. We can see from figure 14 that when the reference is simple, e.g. a binary rectangle, the approach without distance ratio gives more matches within the Post-it. We can see from the image that there are two key points within the Post-it (and the binary rectangle). Those two key points do not have much difference in terms of color intensity or texture. The gradients around those two key points are very similar. Thus they might have similar distance to the same key point within the reference. This is an example where distance ratio can eliminate false negatives.

However, when the reference image has richer features, i.e. with the patterned Post-it, the first approach generates many more false positive than the second approach.

4.2.6. *Speed of SIFT.* Our experiments show that our SIFT method takes on average 59 ms per frame to detect the location of the Post-It, which is much higher than our target maximum of 33 ms. This result is expected as SIFT is a computational heavy feature detector.

4.2.7. *Accuracy of SIFT.* From the above analysis, we found that SIFT works best for patterned Post-its. Both the Plain Post-it and digital rectangle template have too few interesting feature key points, making the perspective transformation between key points rank deficient. For SIFT approach, a patterned Post-It gives best results for our use case. We manually reviewed 100 video frames to measure how accurate the algorithm is in finding matches. The result shows that 94 out of the 100 frames had the Post-it correctly detected. For the other 6, it either identified no match or identified a point outside of the Post-it area (false positive).

4.3. **Experiments using Template Matching.** For the Template Matching method, we rotate and resize the template and do a matching process for each resized and rotated template. This causes latency to increase as we try finer granularity of size and angle rotation. Table 2 shows how processing time varies depending on the parameters. For processing times greater than 70 ms (i.e. surpassing our maximum goal of 33 ms), we experience significant delay in video rendering.

#rotation	#sizes	processing time (ms)
1	1	6
2	2	18
2	3	23
3	5	70
5	10	195

TABLE 2. Template Matching Speeds

Similar to the SIFT case, we tested with both plain Post-Its and Post-Its with patterns on it. In order to maintain low latency, we restricted the number of sizes to 2 and number of rotations to 3. It turns out Template Matching works well on both the patterned and plain Post-its with such rotation and scale iteration. Here, if the box includes $> 50\%$ of the region of the Post-it, we categorize it as a correct match.

In both figure 15 and figure 16, the three images in the first row (from left to right) are the original template, the template after Canny edge detection and the actual resized/rotated template giving the best match.

The three images in the second row (from left to right) are the video frame with best matched region (the rectangle enclosed region), the Canny edge detection of the video frame and the visualization of the convolution between template and the best matched region.

One drawback of the Template Matching approach is that we cannot estimate the full perspective transformation. We can only estimate similarity: we get scaling and rotation by iterating through different scales and rotations of the template, and translation from the box center location of the best matched region.

4.3.1. *Plain Post-It.* Figure 15 shows one frame where we use Template Matching for plain Post-it detection.

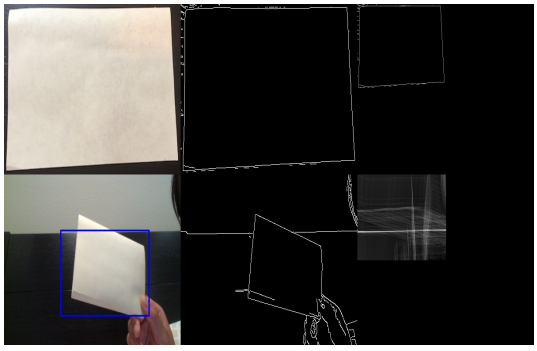


FIGURE 15. Plain Post-it

We reviewed 100 video frames and found 92 to have the correct bounding box for the best matching region. For the rest, the boxes also includes a part ($< 50\%$) of the Post-it, but the majority of the Post-it fell outside of the box.

4.3.2. *Post-It with Pattern.* We reviewed 100 video frames and found 94 to have the correct bounding box for the best matching region. Some of the false matches still have part ($< 50\%$) of the Post-it region in the box, while some completely missed the Post-it.

Figure 16 shows one frame where we use Template Matching with a patterned Post-it.

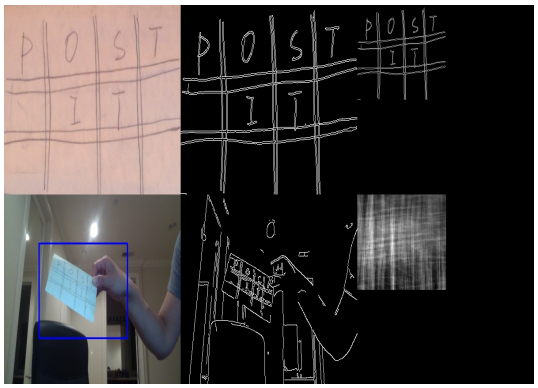


FIGURE 16. Post-it with pattern

4.4. **Comparison of methods.** Comparing the most important metrics for the 3 approaches, we find the manual Color-Shape approach performs much better than both SIFT and Template Matching. Table 3 shows the high-level comparative overview.

	SIFT	Template Matching	Color-Shape
Time (ms)	59	23	12
Accuracy	0.94	0.94 (pattern) 0.92 (plain)	0.98
Notes	Pattern	2 scale 3 rotation	Bright color

TABLE 3. Comparison of Methods

In addition to processing time and detection accuracy, the Color-Shape method provides a more accurate perspective transformation matrix that we use to transform the animation to overlay on the Post-it. For SIFT, the affine transformation can be rank

deficient when insufficient key points are matched. For Template Matching, we can estimate a similarity matrix but cannot estimate the full perspective transformation.

SIFT and Template Matching are designed for general, rich pattern detection, while the Color-Shape method we use is designed specifically for our Post-it use case. SIFT and Template Matching can be used for a much broader range of use cases, whereas the Color-Shape method is limited in scope to Post-it detection (and highly similar use cases).

5. Conclusions

We find that typical methods such as SIFT and Template Matching do not sufficiently meet our goals, as they do not provide accurate pose finding for the plain-faced Post-it, and require too much computation time to operate.

We find the manually designed Color-Shape approach, exploiting the saturated color and square shape properties of the Post-it, to work well and meet our goals. The full Color-Shape method takes an average of 12 milliseconds to execute (with 4.5 milliseconds standard deviation) on a 3-year old MacBook Pro laptop, indicating the maximum 33 milliseconds goal should be achievable on a wide range of laptops. We achieve high accuracy projective pose estimation with a reasonable robustness to noise and lighting variation.

6. Future Work

A number of improvements could be considered for future work:

- Interest region: the Color-Shape approach could be made significantly faster by searching only in an interest region derived from the previous location of the Post-it
- Multiple Post-its: the Color-Shape approach could be generalized to find multiple Post-its in one image
- Dynamic color filter: the HSV color filter could be made more accurate by dynamically adapting parameters to the scene
- Wearables support: the source code could be ported to relevant augmented reality hardware such as smartglasses

References

[1] Canny, J. A Computational Approach to Edge Detection IEEE Trans. on Pattern Analysis and Machine Intelligence, 8(6), pp. 679-698, 1986.

[2] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, Dieter Schmalstieg Real-time detection and tracking for augmented reality on mobile phones IEEE Transactions on Visualization and Computer Graphics (Volume:16 , Issue: 3), 2009

[3] Charles A. Poynton Frequently Asked Questions about Colour Available at https://engineering.purdue.edu/ bouman/info/Color-FAQ.pdf

[4] Fernandes, Leandro AF, and Manuel M. Oliveira. Real-time line detection through an improved Hough transform voting scheme Pattern Recognition 41.1, 2008

[5] Fleck, Margaret M., David A. Forsyth, and Chris Bregler. Finding naked people Computer VisionECCV'96. Springer Berlin Heidelberg, 1996

[6] Fleyeh, Hasan. Color detection and segmentation for road and traffic signs Cybernetics and Intelligent Systems, 2004 IEEE Conference on. Vol. 2. IEEE, 2004

[7] Lee, Jae Y., and Suk I. Yoo An elliptical boundary model for skin color detection Proc. of the 2002 International Conference on Imaging Science, Systems, and Technology, 2002

[8] Lowe, David G. Object recognition from local scale-invariant features Proceedings of the International Conference on Computer Vision. pp. 11501157. doi:10.1109/ICCV.1999.790410, 1999

[9] M. Fiala Designing Highly Reliable Fiducial Markers IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 7

[10] Maini, R. et al. Study and Comparison of Various Image Edge Detection Techniques International Journal of Image Processing (IJIP), Volume (3) : Issue (1), 2009

[11] Matas, J. et al. Robust Detection of Lines Using the Progressive Probabilistic Hough Transform CVIU 78 1, pp 119-137, 2000

[12] Nipat Thiengtham and Yingyos Sriboonruang Improve Template Matching Method in Mobile Augmented Reality for Thai Alphabet Learning International Journal of Smart Home Vol. 6, No. 3, July, 2012

[13] P. Kakumanu, S. Makrogiannis, and N. Bourbakis A survey of skin-color modeling and detection methods Pattern Recogn. 40, 3, March 2007

[14] Palmer, Phil L., Josef Kittler, and Maria Petrou An optimizing line finder using a Hough transform algorithm Computer Vision and Image Understanding 67.1, 1997

[15] Roberto, B. Template Matching techniques in computer vision: theory and practice 2009

[16] S. Garrido-Jurado, R. Muoz-Salinas, F.J. Madrid-Cuevas, M.J. Marn-Jimnez Automatic generation and detection of highly reliable fiducial markers under occlusion Pattern Recognition, Volume 47, Issue 6, June 2014

[17] Schumeyer, Richard P., and Kenneth E. Barner Color-based classifier for region identification in video Photonics West'98 Electronic Imaging. International Society for Optics and Photonics, 1998

[18] Singh, Chandan, and Nitin Bhatia A Fast Decision Technique for Hierarchical Hough Transform for Line Detection arXiv preprint arXiv:1007.0547, 2010

[19] Van Ginkel, Michael, CL Luengo Hendriks, and Lucas J. van Vliet. A short introduction to the Radon and Hough transforms and how they relate to each other Delft University of Technology, 2004