# Shopping Cart Item Tracker

Dian Huang
dhuang05@stanford.edu

*Abstract*— **This paper presents a shopping cart item tracker. It uses background subtraction to detect the keyframes. To extract the foreground pixels in the keyframe, a double-threshold image subtraction technique has been proposed. It allows the threshold of image subtraction to be less strictly defined. The subtracted image is then matched to one of the 50 different items with Scale-invariant Feature Transform (SIFT). Linear Discriminant Analysis (LDA) is also used to recognize if the user puts in or takes out an item. The algorithm has been verified with two videos with 9 randomly selected items.**

***Keywords—image subtraction, background subtraction, LDA, SIFT***

## I. Introduction

Consumers often forget what the items they put in the shopping cart, so they have to check if all the items in the shopping cart are correct, which can be very time-consuming when there are too many items. Although unmanned shopping store can resolve this problem, they have not been popularized as it needs a large number of sensors and cameras to track the motion of the customers and the products. Technical-wise, a store like Amazon Go cannot support more than 20 people shopping simultaneously, at least by 2017 [1].

Therefore, it is necessary to develop an application to track the customer's shopping cart item record in real time. The user can put the phone on top of the shopping cart to capture the item he puts in or takes out. He should also hold the item for about half a second and make sure that the front view of the item is facing the camera. This paper presents the implementation of this application, which involves with keyframe detection, background subtraction, feature matching, and action recognition. It also proposes a novel simple image subtraction technique for static single object extraction.

The paper is organized as follows. Section II presents the initial setup and an overview of the algorithm. Section III describes how to identify the keyframes from the video with background subtraction. Section IV proposed a novel simple image subtraction technique for static single-object extraction. Section V shows the SIFT matching result. Section VI presents a simple chromaticity-based action recognition method that uses LDA as the classifier. Section VII describes the test setup and result. Section VIII includes the conclusion and potential improvement.

## II. Initial Setup & Algorithm Overview

Fifty different items are collected as samples and resized to 300x300 for feature matching. The descriptors of each item are precomputed with Scale Invariant Feature Transform (SIFT). The smartphone is placed on top of the shopping cart to capture the items user puts in or takes out. Each frame captured by the camera is resized to 400x800 to accommodate different screen resolution.

Then Gaussian-mixture model (GMM) background subtraction is used to detect 3 keyframes: background, entering, and foreground. Background frame is the frame before the user puts in or takes out an item. Entering frame is the frame when the user's hand enters the scene. Foreground frame is the frame when the user holds the item in the cart.

Then the foreground frame is subtracted by the background frame to extract the foreground pixels for feature matching. The entering frame is also subtracted by the background frame for action recognition with Linear Discriminant Analysis (LDA), which identifies if the user is putting in or taking out an item, as shown in figure 1.

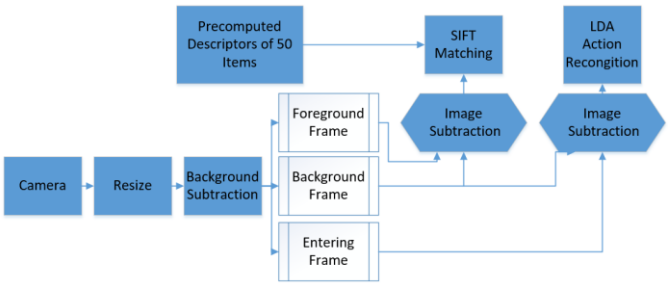



Figure 1: Flow chart of the algorithm for item tracker

## III. Keyframe Detection

Background subtraction with Gaussian mixture model has been very popular in visual surveillance. This technique identifies the intruding object as the foreground and models its probability with a Gaussian mixture model. It was first proposed by Fridman and Russell in 1997 [2], and later on widely popular and improved. This implementation uses the technique proposed by Zivkovic[3] to detect the keyframes.

For the background subtraction with Gaussian mixture model, the probability that each pixel at time t within a period of $T$ classified as the foreground (FG) or background (BG) is expressed by Gaussian mixture model with M components:

$$p(\vec{x}^t|X_T, FG + BG) = \sum_{m=1}^{M} \pi_m N(\vec{x}^t; \mu_m, \sigma_m^2 I) \quad (1)$$

$$\pi_m \leftarrow \pi_m + \alpha(o_m^t - \pi_m) \quad (2)$$

where N denotes the Gaussian distribution, $\alpha = \frac{1}{T}, . o_m^t$ is set to 1 if the newly joined pixel matches closely with others, which leads to larger $\pi_m$ value, or set to 0. So the background probability can be approximated by the sum of B largest clusters:

$$p(\vec{x}^t|X_T, BG) = \sum_{m=1}^{B} \pi_m N(\vec{x}^t; \mu_m, \sigma_m^2 I) \quad (3)$$

Usually, the intruding foreground pixel has a small weight $\pi_m$, but if they remain stable or close to each other long enough, they will be updated as the background pixels as its corresponding $\pi_m$ will be large. So the foreground pixels will not immediately become background pixels. The application will use this property to find the foreground frame.

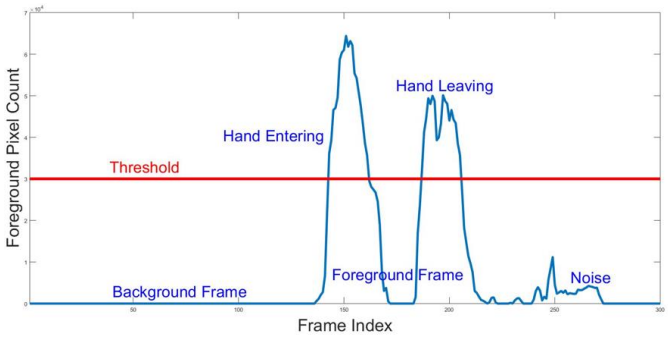
As shown in figure 2, the frames have no foreground pixels before the user's hand enters the scene, so these frames are the background frames. The user's hand entering the scene causes a change in some of the pixels, which become foreground pixels. When its counts are higher than a threshold, this frames is identified as the entering frame. When the user holds the item in front of the camera long enough, the foreground pixels will become background pixels. The user's hand leaving the scene also increases the foreground pixel count. So the frame that corresponds to the local minimum in foreground pixel count is identified as the foreground frame. This frame captures the scene that the user is holding the item. The amount of time the user needs to hold is determined by the number of frames $f$ it takes for a foreground pixel to be identified as a background pixel, as approximated in the following equation:

$$f = \frac{\log(1 - c_f)}{\log\left(1 - \frac{1}{T}\right)} \quad (4)$$

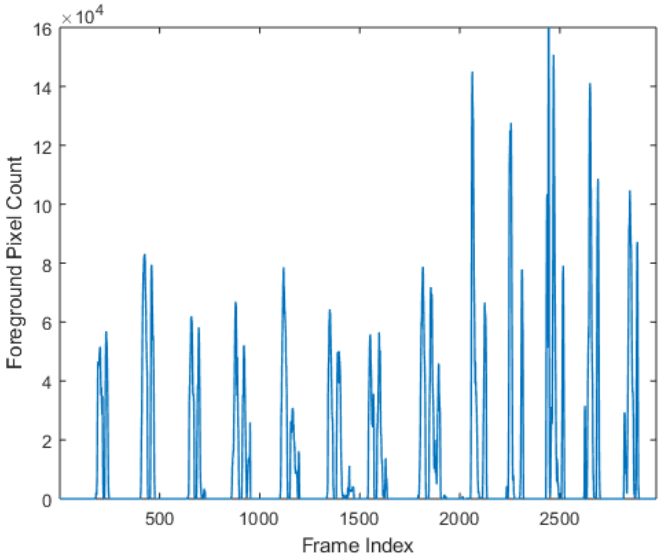
where $c_f$ is a threshold for the sum of the weights. The longer the $T$, the more frames are needed to change a foreground pixel to background pixel.

Closing morphological operator also removes the small foreground regions due to small changes in light caused by other people walking around or the user himself to suppress this noise. Based on the above equation (4), although evaluating pixels in a shorter duration allows the users to hold the item shortly, too short a time can also lead to false detection, such as the user is putting the item in too slowly. Therefore, this application requires the user to hold the item for about 0.5 seconds.

Therefore, only three frames need to be saved for further processing. The background and foreground frame can be found by tracking only the frame with minimum foreground pixel count within a window period. This significantly reduces the run-time memory requirement.




(a) Foreground pixel count when the user puts in or takes out an item




(b) Each double peak indicates the user is putting in or taking out an item

Figure 2. Foreground pixel count for the keyframes.

## IV. Image Subtraction

The foreground frame is then subtracted with the background frame to extract the pixels of the item. However, there are mainly two problems. First, when the user holds the item in front of the camera, it blocks some of the light, thus forming a shadow nearby. If there is an item right at the shadow region, the feature matching may match the item in the shadow region rather than the item user holds. Second, due to different illumination, different item color, it is difficult to find a single threshold that separates the item from the background.

### *A. Shadow Removal*

Shadow detection with the chromaticity-based method proposed by Cucchiara[4] has achieved about 80% accuracy in the indoor environment[5]. This method first converts the image from RGB color space to HSV color space, as the HSV color space separates chromaticity from luminosity. The shadow formed in the foreground frame usually has a much lower value than the background, but only small color variation, so it uses the ratio rather than the difference of the value. The

shadow also has less impact on the hue, and it lowers the saturation. Therefore, if a pixel satisfies the following conditions then it will be classified as shadow:

$$\left|H_{fg} - H_{bg}\right| \leq \tau_H \tag{5}$$

$$S_{fg} - S_{bg} \leq \tau_s \tag{6}$$

$$\beta_1 \leq \left(\frac{V_{fg}}{V_{bg}}\right) \leq \beta_2 \tag{7}$$

where $H_{fg}, H_{bg}$ denotes the hue of foreground and background pixel. $S_{fg}, S_{bg}$ denotes the saturation of the foreground and the background pixel. $V_{fg}, V_{bg}$ denotes the value of the foreground and the background pixel. Threshold $\tau_H, \tau_s, \beta_1, \beta_2$ are optimized for shopping mallthe environment.

As shown in figure 3, this algorithm detects most of the shadow nearby. Although it misclassifies some of the pixels on the item as shadow and does not identify all the shadow pixels, this application requires only the approximate region of the shadow rather than the exact pixel location. The small misclassified region can be removed or filled by the morphological operator such as opening and closing.


(a) Background frame


(b) Foreground frame: arm and item causes shadow nearby


(c) Shadow Detection


(d) Without shadow removal


(e) With Shadow Removal

Figure 3. Shadow Detection and removal

### *B. Double Threshold Image Subtraction*

However, the pixels of the text on the item still cannot be fully classified as foreground pixels, so the threshold needs to be lowered, but lowering the threshold will lead to other unwanted regions to be identified as foreground. Due to the different illumination and item color, it is difficult to find one threshold for all conditions. Therefore, this paper proposes a double threshold method to resolve this problem.

Since the top part of the foreground has clothes and the bottom part has the item, their color, and the illumination are usually different, so the picture is first divided into two parts and processed independently.

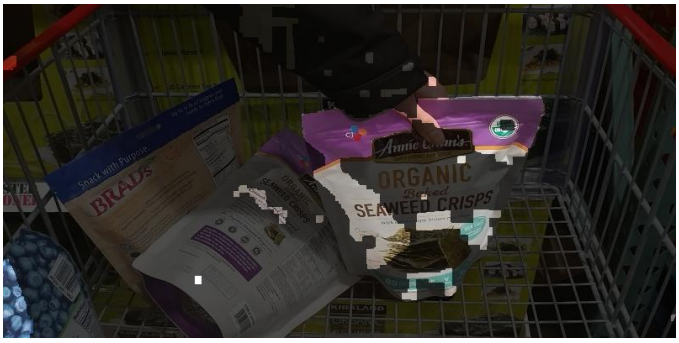
The threshold value is the fraction of the maximum square of the Euclidean distance of the foreground and background pixels in the HSV color space. The two thresholds are:

$$th_h = \max(\Delta H^2 + \Delta S^2 + \Delta V^2)\, k_h \quad (8)$$

$$th_l = \max(\Delta H^2 + \Delta S^2 + \Delta V^2)\, k_l \quad (9)$$

where $th_h, th_l$ denotes the high and low threshold. $k_h, k_l < 1$ $and$ $k_h > k_l$. As shown in figure 4, it first uses a high threshold to find the approximate location of the item, so only a part of the item is identified as foreground. Open and close morphological operator are also used to remove and fill the small region. Then, it applies a low threshold, so the entire item region is classified as foreground, and keep only the foreground region identified by the low threshold that has a strong connection with the region identified by the high threshold. By strong connection, it means that the regions identified by high threshold and low threshold share at least twenty percent of the pixels. So only the regions expanded from the foreground region identified by the high threshold can be identified as foreground. This allows the second threshold to be much lower and less strictly defined.

Even with a high threshold, there can still be some misidentified region. To resolve this problem, this method uses large open and close morphological operators to remove and fill the small region and connects the foreground regions that are close to each other. Then it keeps only the largest foreground region. This will remove other foreground regions identified by the high threshold that cannot find a connection with the largest foreground region, as shown in figure 4. This allows the first threshold to be set in a more relaxed way.




(a) High threshold with shadow removal: Even with a high threshold, there is still an unwanted foreground region at lower left corner.


(b) Low threshold with shadow removal: more unwanted region and the shadow region nearby has been removed by closing.


(c) Remove the region that does not expand from the high threshold foreground. Then select the largest region.

Figure 4. Double threshold image subtraction

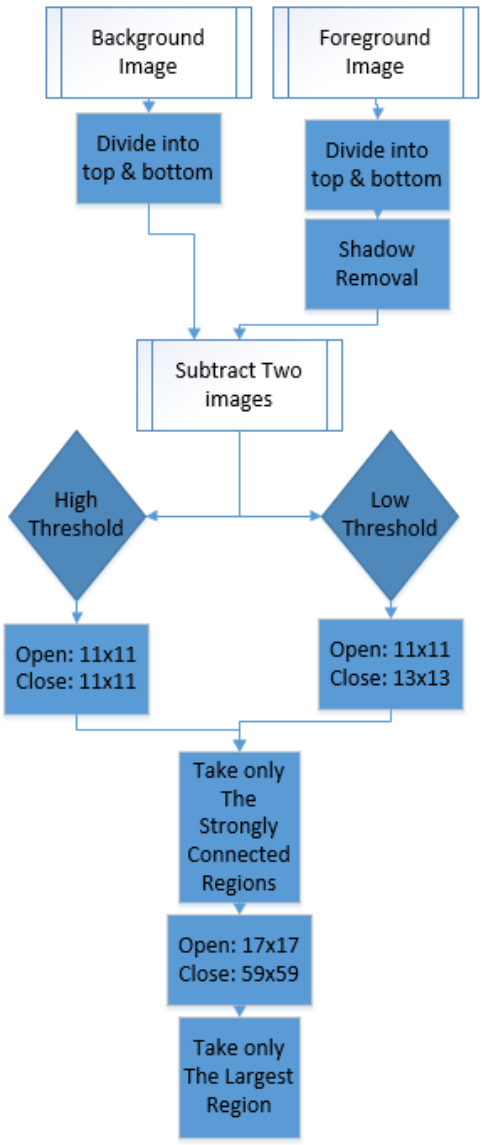



Figure 5. Flow chart for double threshold image subtraction

## V. Feature Matching

The keypoints and descriptors of the extracted foreground are then calculated with Scale-invariant Feature Transform (SIFT) proposed by Lowe in 1999[6]. The descriptors are matched to the pre-computed descriptors of 50 different items with k-nearest neighbors algorithm.

Ratio test proposed by Lowe [7] is used to determine if a match is correct, which takes the ratio between the closest and the next closest match. Since the background in the sample image has not been removed, it is likely to match the key points in the background of the sample image that has part of the shopping cart rather than the item itself, especially if some region of the shopping cart is classified as foreground during image subtraction. Therefore, a lower ratio of 0.7 rather than 0.8 in the paper is applied to reject more incorrect matches, although it also discards more correct matches as illustrated in figure 6 from Lowe's paper [2].

Minimum 10 good matched points are needed to be considered as a match. As shown in figure 7, many of the good-matched keypoints are around the text, so the items that do not have much text feature will be more difficult to match. If there is no match, the item will not be shown on the shopping cart list. In the future, it should allow the user to enter the information about the item when it cannot find a good match, so the item list can be updated.

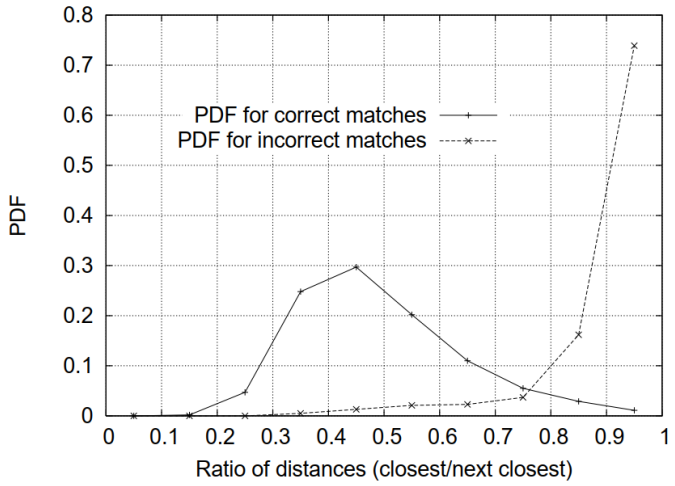



Figure 6. from [7]. PDF of the correct match and incorrect match vs ratio of distance. As ratio becomes larger, more correct matches and incorrect matches will be included.

.

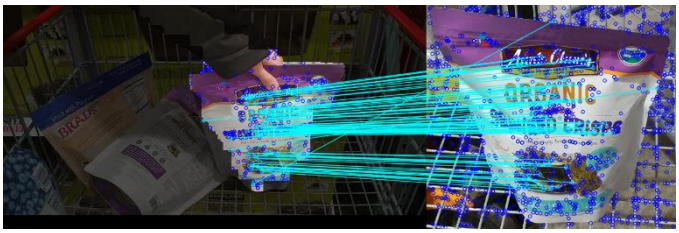


Figure 7. SIFT matching: only the good matches are shown

## VI. Action Recognition

It is important to recognize if the user is putting in or taking out an item. Traditional action recognition typically involves action representation and action classification. The dynamic human motion is encoded into a single image and then classified by support vector machine, k-nearest neighbors, a bag of the word, etc. [8]. However, in this application, since it only needs to recognize two types of actions, and we assume that the clothes typically have a different color from the item, it is possible to apply Linear Discriminant Analysis (LDA) to distinguish the color difference and identify the activity.

As shown in figure 8, at the foreground frame, the user's clothes are on top of the image, and the item is at the bottom of the image. Therefore, Linear Discriminant Analysis (LDA) can use the RGB values of the foreground pixels of the two parts to calculates the eigenvector. It first labels the RGB values of the foreground pixels at the top to be clothes and the bottom to be the item. The middle part is ignored as it contains the features of clothes, hand, item, and it is also difficult to extract the exact blob of these features. Next, the RGB values of the bottom part foreground pixels of the entering frame are projected onto the eigenvector for classification. When the user takes out the item from the shopping cart, the hand and clothes enter the scene first. When the user puts an item into the cart, the item enters the scene first, so the color of the foreground pixels is mainly the item color. Therefore, LDA can classify these two actions based on the color difference in the entering frame.

There are three advantages to this classification method. First, although the foreground pixels are blurred by motion in the entering frame, in LDA, to calculate eigenvector, the mean of each class needs to be calculated first, and then the within-class and between-class covariance matrix. So the impact of motion blur is not severe. Second, LDA tends to maximize class separation, so it can distinguish the item from clothes when they have a different color. Third, as the whole process takes place at the same location within a few seconds, it is not sensitive to illumination changes that may affect the RGB value of the item or the clothes.

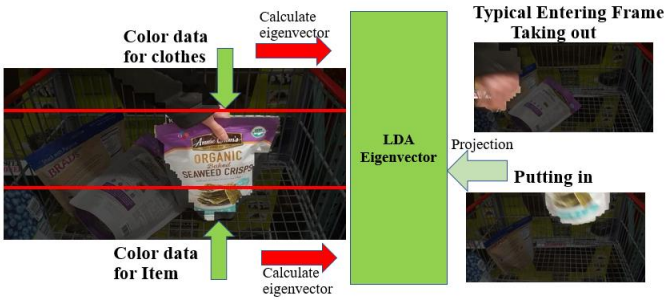



Figure 8. Color data are used to calculate LDA eigenvector. Then the foreground pixels of entering frame projects to the eigenvector for action recognition.

## VII. Test & Result

### A. Test Setup

The phone is placed on top of the cart. Due to angle, its camera can only capture half of the shopping cart, as shown in figure 9. So the user needs to hold the item in the front of the captured part then he can put the item anywhere he wants. The sample images and the video are taken at a different location with different illuminations.

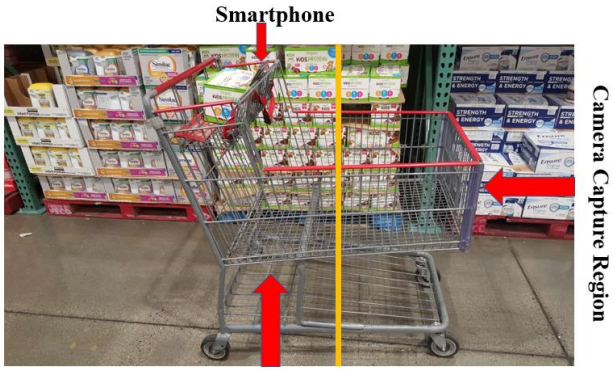



Figure 9. Test setup: the user needs to place the item in the camera capture region first before putting it in other location

### B. Result

9 different items are randomly selected and put into the shopping cart. Some items are placed in the camera captured region, and some are placed in the non-captured region. Then a few items are randomly selected and taken out. Although the RGB difference between the item and the background may require a different threshold during image subtraction, and other items already in the shopping cart also have a different response to the shadow, the algorithm is able to extract the foreground pixels and produce the correct match for all the items. It is also able to predict if the user is putting in or taking out the item correctly. This algorithm has also been verified with two videos with different items.

As illustrated in figure 10, when the user puts in an item, the application will add this item into the list shown on the top of the scene. If the user takes out an item, that item will be removed from the list.

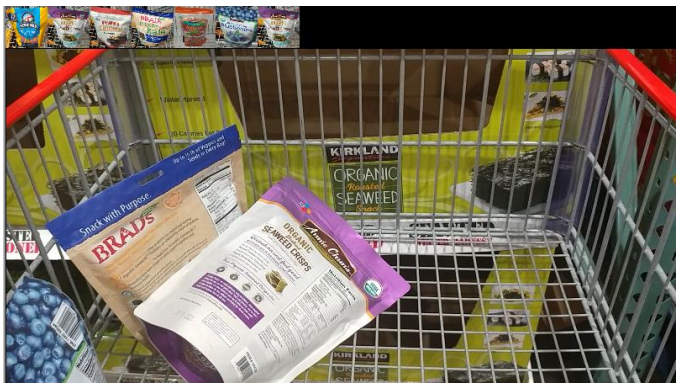



Figure 10. User interface: on the top of the frame shows the item list of the shopping cart.

## VIII. Conclusion

This paper presents a shopping cart item tracker that uses background subtraction with GMM to detect the keyframes. The proposed double-threshold image subtraction algorithm makes the extraction of foreground pixels for a single object image subtraction more robust and more accurately. The paper also presents a simple action recognition based on the difference in color between the item and the clothes. All of the above algorithms have shown a promising result when the user puts in or takes out 8 randomly selected items.

However, there is still much room for improvement, and more tests under different illumination, different color of clothes are needed to verify the proposed algorithm. The action recognition algorithm can also be improved with the data about hand significantly, so it does not rely on the difference in color between the item and the clothes.

Currently, the algorithm is implemented on a personal computer. In the future, a smartphone application can be developed.

The author of the paper Dian Huang did all the work independently.

## References


[1] Kastrenakes, Jacob (March 27, 2017). "Amazon's cashier-free store reportedly breaks if more than 20 people are in it". *The Verge*. Retrieved October 8, 2017.

[2] N. Friedman, S. Russell, "Image Segmentation in Video Sequences: A Probabilistic Approach", *Proc. 13th Conf. Uncertainty in Artificial Intelligence (UAI)*, 1997-Aug.

[3] Z. Zivkovic and F. van der Heijden. "Efficient adaptive density estimation per image pixel for the task of background subtraction". *Pattern Recognition Letters*, 27:773-780, 2006.

[4] Cucchiara, R., et al. "Detecting Moving Objects, Ghosts, and Shadows in Video Streams." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, 2003, pp. 1337–1342., doi:10.1109/tpami.2003.1233909.

[5] A. Sanin, C. Sanderson, B. C. Lovell, "Shadow detection: A survey and comparative evaluation of recent methods", *PR*, vol. 45, no. 4, pp. 1684-1695, 2012.

[6] Lowe, David G. "Object recognition from local scale-invariant features". *Proceedings of the International Conference on Computer Vision. no. 2. pp. 1150–1157, 1999.*

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91-110, 2004.

[8] Kong, Yu, and Yun Fu. "Action Recognition and Human Interaction." *Human Activity Recognition and Prediction*, 2015, pp. 23–48., doi:10.1007/978-3-319-27004-3_2