EECS 4422 Introduction to Computer Vision

Gesture Triggered Alarming System

Project Report

Yucheng Zhou 213169636

1. Objective

This project aims at developing a gesture triggered alarming system (GTAS) that allow user to send out emergency help message by presenting a gesture to the camera. In U.S., convenience store or gas station crimes result in billions of losses every year. Worse, innocent clerks often get involved and harmed. Although there's emergency alarm installed, under the threatening from the criminals, sometimes the clerk cannot seek for help directly and openly (e.g. can't press a button or use phone). Therefore, there's interest to develop an automated system that doesn't require physical interactions to trigger the alarm, for example, presenting a gesture to the camera as the trigger of alarm, and a computer vision algorithm to detect the gesture. To realize this function, this algorithm must be able to analyse frames coming from input video stream then identify the gesture in a reliable manner.

2. Method

1) Assumptions

Considering the purpose of the system and its main use cases, there are few environmental assumptions need to be made:

Assumption 1: the camera has a fixed position and angle. This assumption is made to reduce the complexity of computation.

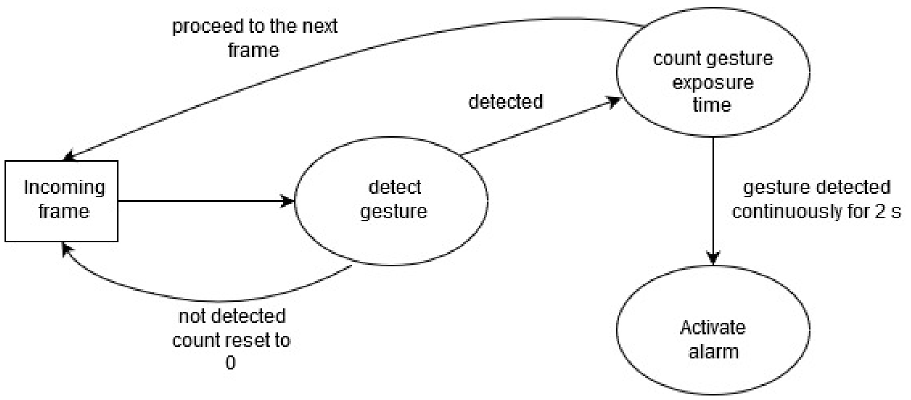
Assumption 2: the background camera is facing is not very complex. This assumption is also made for the same purpose as the previous assumption.

Assumption 3: the hand is close to the camera, in a range around [0.5, 1.5] meter. Similarly, overcomplicated noise removal is expected to be avoided.

Assumption 4: the system doesn't strictly have to be real-time. Although in general an always-on system is expected to be real-time, to concentrate on the field of computer vision, run-time concerns will not be addressed.

2) System Level Implementation

The design of GTAS whole system is composed of 4 states. The chart below describes the state switching mechanism. The main detection algorithm resides in monitoring state.


proceed to the next frame
count gesture exposure time
detected
Incoming frame
detect gesture
gesture detected continuously for 2 s
not detected count reset to 0
Activate alarm

The system shall run as long as the input video stream is on and not empty. After the gesture is detected in a frame, use a counter to record the number of consecutive detected. The counter represents how long has the gesture been posed. If the gesture has been detected consecutively for 2 seconds, it should activate the alarm. If the gesture is not detected again within 2 seconds, then reset the counter to 0 and wait for the next detection. The conversion of digital time and actual time is defined by:

$$time_{actual} = time_{digital} \times {}^{1}/_{FPS}$$

FPS is frame rate. If frame rate of the video stream is 30 fps, then 60 frames indicated a 2 seconds duration.

3) Detection Algorithm Implementation

The algorithm is equipped with 2 computer vision methods: Key Points Matching and Similarity Comparing[1]. The package I used to implement these methods is *opencv*. *opencv* is actually powerful package. The function I mainly used is *cv2.matchTemplate()* and *cv2.BFMatcher()*. These 2 methods can be used independently or jointly to determine if the frame contains the target gesture.

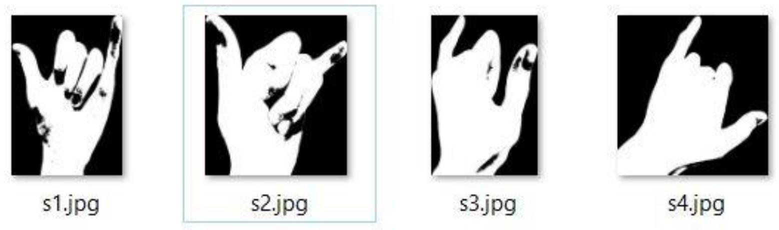
Setup

Before implementing these methods, some setups need to be prepared:

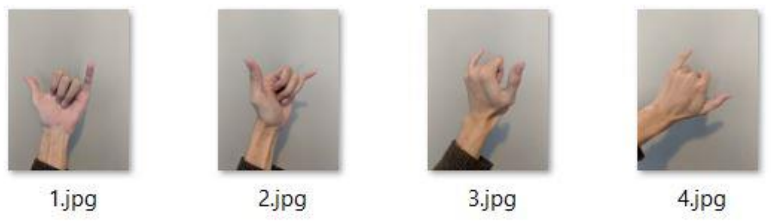
- For Similarity method, a template image of the gesture needs to slide over the frame and be compared. I created a set of 4 standard images that project the same gesture from 4 different angles. Then I filtered out the background as much as possible by

[1] The Similarity method mentioned in this report is NOT *compare_ ssim()* from *skimage* package (the one from the presentation is). *cv2.matchTemplate()* is armed with some functions to compare the similarity, e.g. CCOEFF_NORMED.

pixel thresholding[2], extract a region of interest. The binary ROI images are the final standard images (their sizes are relatively small, less than 160,000 pixels).

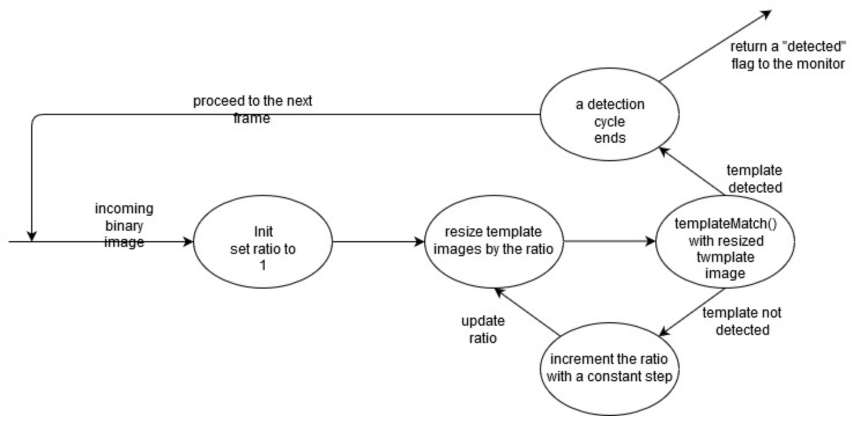


- For Key Points Matching method, descriptors are needed. I also created 4 sets of descriptors, from the same 4 original images I used for *SSIM* method.



Similarity Comparing Implementation

The flow chart below explains how to implement similarity comparison with $cv2.matchTemplate()$ function:

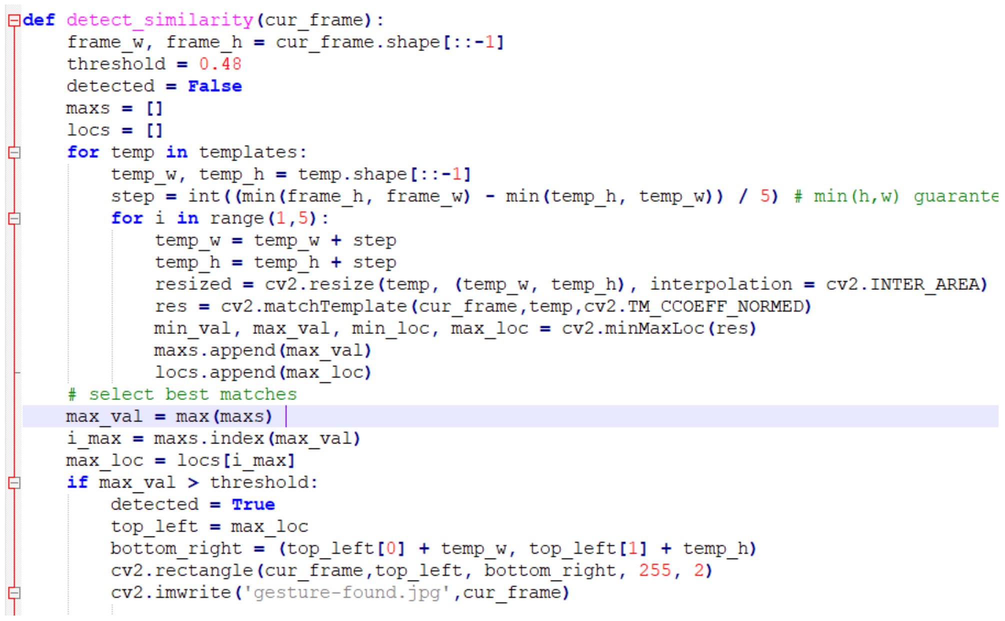

return a "detected" flag to the monitor
proceed to the next frame
a detection cycle ends
template detected
incoming binary image
Init set ratio to 1
resize template images by the ratio
templateMatch() with resized twmplate image
template not detected
update ratio
increment the ratio with a constant step

Check footnote [3] for more information on "ratio".

[2] To create a binary image, saliency map could also be employed instead of masking pixels by defining the expected value of pixels. However, in an indoor environment (complexity of objects) with unnatural light source, masking with saliency map does not perform well.

[3] Ratio in the flow chart refers to resize ratio used to resize template standard images. It is initialized at 1, indicate original size. The maximum number of increments attempted is 5. The step of each increment is calculated as :

```
    step = (min(frame_h, frame_w) - min(temp_h, temp_w)) / 5 % start small
```



```
def detect_similarity(cur_frame):
    frame_w, frame_h = cur_frame.shape[::-1]
    threshold = 0.48
    detected = False
    maxs = []
    locs = []
    for temp in templates:
        temp_w, temp_h = temp.shape[::-1]
        step = int((min(frame_h, frame_w) - min(temp_h, temp_w)) / 5) # min(h,w) guarante
        for i in range(1,5):
            temp_w = temp_w + step
            temp_h = temp_h + step
            resized = cv2.resize(temp, (temp_w, temp_h), interpolation = cv2.INTER_AREA)
            res = cv2.matchTemplate(cur_frame,temp,cv2.TM_CCOEFF_NORMED)
            min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
            maxs.append(max_val)
            locs.append(max_loc)
    # select best matches
    max_val = max(maxs)
    i_max = maxs.index(max_val)
    max_loc = locs[i_max]
    if max_val > threshold:
        detected = True
        top_left = max_loc
        bottom_right = (top_left[0] + temp_w, top_left[1] + temp_h)
        cv2.rectangle(cur_frame,top_left, bottom_right, 255, 2)
        cv2.imwrite('gesture-found.jpg',cur_frame)
```

Above is the code of implementing similarity check. Threshold is introduced to the precision of prediction. Its value is initially random assigned, then tune it so that it yields the best result.

Key Points Matching Implementation:

A key point is a point the contains unique features. We can build a feature vector that contains all possible features of the gesture, then identify if these features exist in the candidate frame. The structure of this method is relatively simple due to the introduction of feature detection algorithms offered by opencv. This method can be implemented in 3 steps.

step 1: accept incoming frame, extract key points and corresponding descriptors.

step 2: compare the key points I just obtained with the key points in the library.

step 3: set a threshold on distances between matches of key points. Only the pairs that have Euclidean distance smaller than threshold will be marked as effective matching. If there are at least 10 pairs of effective matching, then it should flag that object detected.

4) Model Validation

To verify the correctness of my implementation, I recorded 10 videos, each one has approximately 2000 frames. Then I passed them to the system and obtained the predictions generated by the program. In this section, the

The iteration of ratio is needed because the mechanism $matchTemplate()$ applied as it doesn't consider the size of objects. Therefore, I have to manually loop and adjust the size see if there is a match.

implementation code is modified a bit so that it's able to display frame by frame, so that at the same time, I can manually mark the flag whether this detection is a TP, FP or FN.

3. Result

Tests must be conducted to evaluate the performance of a method. A test can be performed by creating a set of test inputs and let the program executes the inputs, finally compare the actual output and expected output. The 10 test input videos are distinct to each other, in terms of duration, distance of camera to object, brightness, hand that present the gesture, and other properties.

The performance is evaluated by calculating precisions and recalls. In order to obtain the precision and recall of an algorithm, I need to count TP, FP, FN, TN. Unfortunately, the verifying process cannot be efficiently automated, so I inspected manually to obtain the statistic. Here is a table of results:

Test #	Similarity Comparing		Key Points Matching	
	Precision	Recall	Precision	Recall
1	0.872	0.732	1	0.803
2	0.673	0.778	1	0.612
3	0.91	0.69	1	0.761
4	0.347	0.761	0.976	0.860
5	0.721	0.525	0.889	0.892
6	0.765	0.613	0.780	0.722
7	0.591	0.801	0.823	0.757
8	0.333	0.899	1	0.760
9	0.482	0.828	0.910	0.720
10	0	0	0.596	0.730

In general, a precision-recall curve will be plot for further discussion, but in this case, due to a small dataset (only 10 tests) the curve derived from it will not be very meaningful. i.e. to obtain a relation that is statistically meaningful, a big enough sample set is required. Hence, I decided not to plot the curve.

The results generated by 2 methods is different as well. Here is a comparison, applied 2 methods to the same image:

Similarity	Key Points

4. Analysis

The analytical reviewing will be focus on a) accuracy of prediction, b) constraints and, c) potential future improvement.

a) Accuracy:

Key points matching method is able to achieve a high score on precision, which indicates they perform pretty well in terms of avoiding faulty matches (if the prediction suggests a detect, then the object very likely exists). In contrast, the FP rate in similarity method is not very optimistic. The reason I supposed is in similarity comparison I used binary image instead of full image, which can result in a reduce of target object's "uniqueness". It's not very hard to understand that the more unique the target is, the easier to identify.

In terms of recall rate, which is also known as sensitivity, similarity method is able to have similar performance as the other one. Sensitivity indicates the ability of identifying all targets.

Both methods have consistent performance on identifying a frame that contains the gesture, but not very consistent not picking a wrong target. However, they have completely different performance for the last test, Test 10. The video for that test was taken at a completely different background (video 1 to 9 were recorded in my room, video 10 was recorded in the waiting room at Appletree).

b) Constraints:

Every system performs differently under different conditions. Some conditions constraint the program's effectiveness. Similarity method is more constrained compared to key points method, for example, ratio and angle must be considered when performing 2D comparison, while with key points matching, it's no longer necessary to preprocess the frame.

However, in key points method, tuning the threshold will cost extra efforts. How to determine the value of maximum Euclidean distance? What's the minimum number of key point pairs that guarantees a "detected"? Numerous testing is needed to ensure the universalness in the algorithm, so that it can always be deployed.

c) **Potential Improvement:**

As we can see the performance of such algorithm is sensitive to the environment, thereby the effect of environment must be minimized. To achieve this, a salient process can be added to the incoming frame before it gets fed into the algorithm. By evaluating the saliency map, background objects can be masked off, left only the interested object.

Besides, it is also possible to have both algorithms evaluating the same source at the same time, and each's result can be used to verify each other's correctness mutually. i.e. If both methods returns "detected", then the final prediction is "detected"; similarly, if both return "not detected", then returns "not detected"; if one outputs "detected" another outputs no, then do something to determine which result should be adopted.

5. Conclusion

Gesture triggered alarming system can be realized just by utilizing computer vision methods without a deep learning model. Although it is totally feasible, but it is also hard to guarantee high reliability and robustness in practice. Key points matching is still very powerful handling matching objects from 2 different sources, especially in the case of objects with complex shape, the accuracy and efficiency often holds.