CS585 - Assignment 2

Name: Nazia Tasnim

Teammate: Shubhrangshu Bit

Date: 02/14/2024

Problem Definition

The problem is to implement a simple image processing pipeline that takes input image stream from the webcam, performs a series of image processing operations to identify the object of interest, and then displays the processed image with the object of interest highlighted on a GUI.

In our case the object of interest is human hand, and we plan to identify 4 different hand gestures corresponding to 1, 5, 6 and 9 in Bangla Sign Language System. - Index and middle finger for 1 -> 161 sample - Open palm for 5 -> 152 sample - Thumbs up for 6 -> 154 sample - Thumbs down for 9 -> 149 sample

Challenges

- Identifying skin-color bobs across various lighting conditions and orientation is difficult
- Gesture detection may be influenced by the background
- Finding an appropriate template for each gesture is not a linear task. The same can be said for the thresholding values
- The hand gesture for 1 and 6 are very similar and can be easily confused.

Method and Implementation

While similar pipelines can be made very sophisticated using deep learning and other advanced techniques, **we strictly stuck with the techniques demosntrated to us in the classes and labs**. To simplify some of the modules, we have used `opencv-python` library. Few modules were also adapted directly from lab exercises. The demo pipeline can be run from `exp_nb2.ipynb` file. The utils and modules are in `utils.py` and `modules.py` respectively. The video demo is also available here.

The pipeline consists of the following steps:

1. **Capture Image**: We capture the image from the webcam using `cv2.VideoCapture(0)`. A clean frame is captured to calculate the background through frame differencing.We show 4 windows at a time, the original frame, the background, skin detection, and the hand gesture. Related functions `find_frame_difference(current_frame, background)` and `get_motion_energy(frame_difference)`.
2. **Skin Detection**: We use the HSV color space to detect skin within a certain range, and generate a binary mask. The mask is then used to

filter out the non-skin pixels from the original frame.We calculate the largest contour in the mask and draw a bounding box around it.Used the following ranges for skin detection:

- `Lower Range: [0, 100, 100]`
- `Upper Range: [20, 230, 230]`

Related functions `find_skin_color_blobs(frame)`, `find_contours(mask)`, & `find_projection_bounding_box(frame)`.

3. **Shape Features**: We fit an ellipse around the largest contour, then calculate it's circularity, orientation, size and position. The initial plan was to use this information to reduce the search-space during template matching and modify the templates accordingly. Related functions `get_shape_features(shape_frame, contour, x, y)`, `get_orientation_features(moment)`, & `degree_to_plaintext(theta)`.

4. **Template Generation**: We separate out a subset of the images for each class and average them out to get a single template for each class. We also threshold the templates to get a binary image. Template can be generated from either the skin blob mask or the largest contour ROI. Related functions : `generate_template(image)`, `generate_template_from_mask(mask)`, & `get_thresholded_template(template_directory)`.

5. **Pyramid Generation**: Pyramid is generated for both the templates and the skin blob mask. This is done to match the templates with the skin blob mask at different scales. By default we use 6 levels of image pyramid for both. Related functions `get_pyramid(image)`.

6. **Template Matching**: This stage combines all the submodules described earlier for a custom template matching pipeline. Image and templates are matched against each other through `cv2.matchTemplate` across corresponding pyramid levels. After iterating through each level, we calculate an average of the match scores to get the final match score. If the match score is above a certain threshold, we draw a bounding box around the matched region and show the gesture. Related functions `custom_template_matching(skin_mask,template_pyramids, threshold=0.6)`.

7. **Display**: We display the original frame, the background difference, the skin blob mask, and the hand gesture in separate windows. We also display the shape features and the match score on the original frame in the gesture window. When the gesture is detected, we display the corresponding number on the gesture window.

Experiments

Following the pipeline, we have experimented with different threshold values for gesture detection, different number of pyramid levels, largest contour offsets, and different number of images for template generation. We also had to manually tweak the skin color range through trial and error to get the best results. We also experimented with rotating and flipping the templates to see if it improves the accuracy.

Results

After an initial phase of qualitative testing, we proceeded to calculate confusion matrix and accuracy for the 4 classes. The results are as follows Confusion Matrix:

	1	5	6	9
1	135	18	0	4
5	94	36	20	2
6	96	37	18	3
9	32	51	33	33

- The pipeline performs best for identifying the gesture for 1, and worst for 6.

  - Generating templates from skin masks is more accurate than generating templates from the largest contour ROI.

Discussion

The pipeline is not perfect and has a lot of room for improvement. The skin detection is not robust and can be easily influenced by the background. The template matching is also not very accurate and can be influenced by the orientation and size of the hand. The pipeline is also not very fast and can be optimized further.

However, it is to be noted that the methodology we employed is heavily restricted by only using the techniques we have learned in the class. Even without using deep learning, we may be able to improve performance by using techniques such as k-means clustering for skin detection, convex hull for finger detection and using more advanced template matching techniques such as SIFT or SURF.

Conclusion

In conclusion, we have implemented a simple image processing pipeline to detect hand gestures. When the background is dark and there is a single light source, resulting in fewer background noises, and when the hands are clearly visible in the video, the accuracy rate is satisfactory.