CS585 Assignment 2: Scuba Hand Signal Recognition

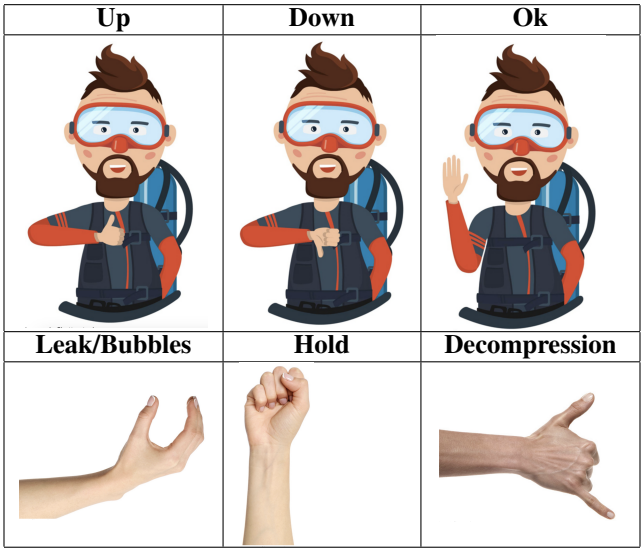
Seunghwan Hyun
Teammate: Hao Qi
February 14, 2024

1 Problem Definition

The goal of this project is to apply classic computer vision techniques learned in the image and video computing class to recognize essential scuba diving hand signals, such as "up," "down," "ok," "bubbles," "hold," and "decompression." These signals are crucial for conveying critical information and managing safety procedures in the non-verbal underwater environment. Due to their visual similarity, accurately distinguishing between these gestures poses a significant challenge for automated systems.

1.1 Hand Signals

After researching different hand signals, we found scuba hand signal from a scuda diving website . We chose 6 essential signals that displayed similar patterns. For instance, 'Up' and 'Down' can be reflection of on another and decompression also shows similar shape.



1.2 Goals and difficulties

We plan to implement several classic algorithms using some of the library functions of OpenCV, such as skin color detection, circularity measurement, template matching, etc. By combining these algorithms, we aim to improve recognition accuracy and enhance the resistance to background environmental interference. We will create our dataset to evaluate the performance of the model quantitatively and will also design a graphical interface to detect inputs from the camera dynamically.

The key to accurate recognition is to effectively extract and utilize the diverse morphological features of hand signals. We need to be familiar with the library functions we use and choose the appropriate hyperparameters (such as multiple thresholds) through tests, which I will explain one by one below. Other potential issues include converting image size and channels, selecting the testing environment, etc.

2 Method and Implementation

In this project, I was responsible for preparing the data, implementing template matching and integrating different methods to create ensemble model and fine-tuning the weights for optimal output. Hao took responsibility in creating the backbone of our code, implementing GUI and calculating circularity.

2.1 Data Collection

We took 15 photos for each of 6 hand signals (90 photos total) in same lighting and same background. We did change the size and angle of the hand and fingers to add little noise. Also, to make sure only the hand was being segmented, I wore a sweater and covered my wrist so only my hand would be visible.

2.2 Creating Templates

For each hand signals, I used 5 images to create a template. I changed the image to binary image using HSV and I used contours to find a bounding box of the hand sign. Since the hand sign was the biggest object in the image, I was able to detect the hand sign with max contour. After I had the bounding box, I saved them as template which resulted in 5 templates for each hand sign (total of 30 templates).

```
def extract_template(img_path):
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    high_HSV = np.array([15, 255, 255])
    low_HSV = np.array([0, 50, 50])
    img = cv2.inRange(img, low_HSV, high_HSV)

    _, binary_image = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY + cv2.
                                                    THRESH_OTSU)
    contours, _ = cv2.findContours(binary_image, cv2.RETR_EXTERNAL, cv2.
                                                    CHAIN_APPROX_SIMPLE)
    max_contour = max(contours, key=cv2.contourArea)
    x, y, w, h = cv2.boundingRect(max_contour)
    cropped_template = binary_image[y:y+h, x:x+w]
    return cropped_template
```

2.3 Template Matching

For an input image, I run the template matching with all 30 templates and save the maximum 'matching' value of each template in a list. Because each hand sign has 5 templates, I take average of 5 'matching values' for each hand sign. Predicted output would be the hand sign with the greatest average 'matching values'

```
predicted_handshape = [0]*6
ind = 0
for filename in os.listdir(template_path):
    file_path = os.path.join('templates', filename)
    template = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
    # print(self.img.shape)
    # print(template.shape)
    result = cv2.matchTemplate(self.img, template, cv2.TM_SQDIFF)

    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
    predicted_handshape[ind//5] += max_val
    ind += 1

answer = ['bubbles', 'decompress', 'down', 'hold', 'ok', 'up']
max_prob = max(predicted_handshape)
```

```
ind = int(predicted_handshape.index(max_prob))
gesture = answer[ind]
return gesture
```

2.4 Ensemble

I used the output of template matching and circularity to improve model's performance.

For the circularity, I labelled the output if a hand sign displayed a significantly different value from others.

```
# First Ensemble
if circularity >= 0.43:
    gesture = ['hold']
elif circularity < 0.43 and circularity >= 0.31:
    gesture = ['bubbles']
elif circularity < 0.31:
    gesture = ['down', 'decompress', 'ok', 'up']
```

In first ensemble, I used circularity prediction and template matching prediction together to create an ensemble model and compared it with template matching model. If the circularity output was 'hold' or 'bubbles', that was my final output. However, if that was not the case, template matching prediction was my final output.

In Second ensemble, if the circularity output was 'hold', that was my final output. However, if that was not the case, template matching prediction was my final output.

3 Experiments

3.1 Data

We collected a total of 90 images from a white background with a mobile phone in a fixed angle and lighting. 90 images can be broken down to 15 images per each hand sign. For each hand sign, we used the first 5 to calculate the circularity and produce the templates. We used all 15 images per hand sign for calculating Accuracy and F1-score for our model.

3.2 Hand Segmentation

Goal of our first experiment was to use different methods to have clear and exact inputs of hand images. It was hard to determine the evaluation metric at this stage so we checked the output of sample data and chose the parameters that we thought had the best segmentation of hand in a naked eye.

3.3 Circularity

We attempted to calculate average circularity for each hand signs to improve the performance of our prediction model. Because each hand sign has different shape, if circularity of each hand sign differs greatly from the others, using circularity to predict the hand sign would show great performance. We used 5 images for each hand sign to calculate the circularity and apply it to our model.

3.4 Template Matching

Template matching is a simple and direct method that calculates the similarity between given image and the template. There has been different ways of calculating the difference between the target image and the template which is what we experimented with, the 'Template Matching Mode'. We tried different template matching modes in the OpenCV website. We used Accuracy and F1-Score as a evaluation metric for how each hand sign template accurately predicts the hand sign of input image.

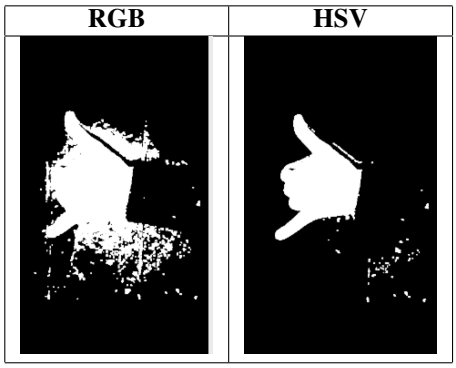
3.5 Ensemble

We compared the performance of 3 models. First model used Template matching. Second model used Circularity (when it predicts 'hold' and 'bubbles') and Template matching. Third model used Circularity (when it predicts 'hold') and Template matching.

4 Results

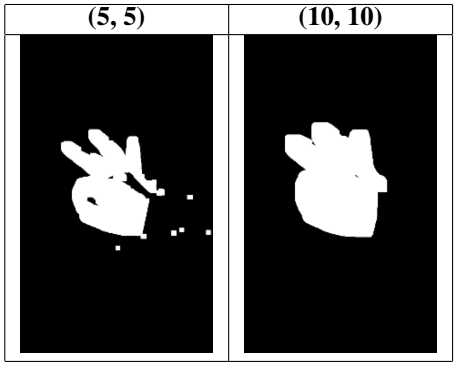
4.1 RGB vs. HSV

For the hyperparameters we used, converting the image into the HSV channel before binarization yielded better results. Therefore, we used it as the method for skin detection.



4.2 Kernel size

We need to use kernels when applying Gaussian blur, closing, and dilation operations. If the kernel is too large, it may lead to loss of image information, while if it is too small, it may result in excessive noise. In our experiments, we used relatively small kernels because we first cropped the image to a size of 480x480 before processing it, and the images were not large, with the hand being relatively small in the image.



4.3 Circularity

'Hold' hand sign had a circularity around 0.6 which was significantly greater than the other hand signs. However, other hand signs like 'Bubbles', 'Ok', 'Up' did not show significant difference when it comes to mean circularity of each hand signs. Although there was a subtle difference in mean of the circularity for each hand sign, large range made it difficult to predict the hand sign based solely on circularity.

	1	**2**	**3**	**4**	**5**
Bubbles	0.281	0.251	0.237	0.263	0.268
Decompress	0.317	0.258	0.331	0.302	0.259
Down	0.376	0.413	0.408	0.400	0.394
Hold	0.618	0.627	0.591	0.664	0.651
Ok	0.291	0.229	0.314	0.269	0.250
Up	0.290	0.322	0.399	0.225	0.287

Table 1: Circularity for Hand Signs

4.4 Template matching mode

This experiment was done before we synced image resizing method and increased the accuracy to 0.88. However, we expect the output to have similar performance as well.

We discovered that normalization modes outperformed their corresponding modes. Template matching's weakness is that they are not robust to variations in lighting and contrast to the surroundings. This could be the reason why normalization modes showed better performance because **CCOEFF_NORMED** showed the best performance in both Accuracy and F1-score.

	Accuracy	**F1-Score**
TM_CCOEFF	0.22	0.12
TM_CCOEFF_NORMED	0.40	0.30
TM_CCORR	0.17	0.05
TM_CCORR_NORMED	0.23	0.15
TM_SQDIFF	0.17	0.05
TM_SQDIFF_NORMED	0.17	0.05

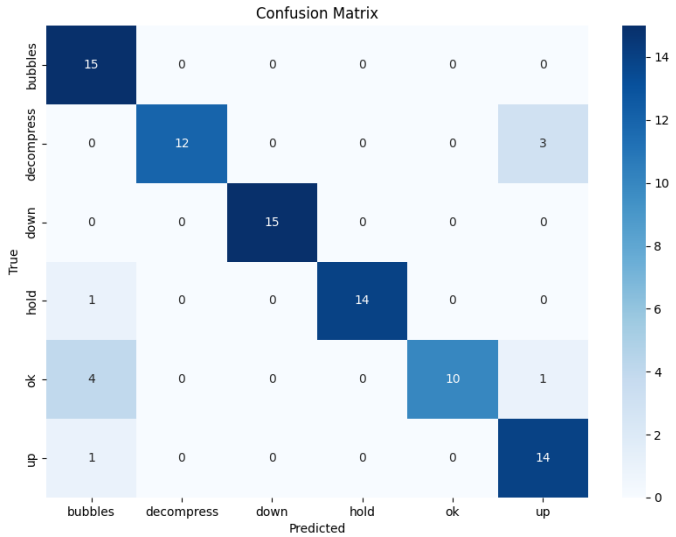
Table 2: Template Matching Operation Performance

4.5 Ensemble

Our Best model was Template Matching model which achieved an accuracy of 88.88% and F1-Score of 0.8894. Each hand sign had 15 images and following is our confusion matrix.

First Ensemble model was Circularity (when it predicts 'hold' and 'bubbles') and Template matching. It showed an accuracy of 68.88% and F1-score of 0.7039.

Second Ensemble model used Circularity (when it predicts 'hold') and Template matching. It showed an accuracy of 82.22% and F1-score of 0.8239.



Our model performed well with most of hand signs but okay and 'decompass'. Wrong cases of 'decompass' was predicted as 'up' and 'ok' was predicted as 'bubbles' which makes sense because they look alike one another in contour form.

This is the demo video of our model and GUI. The related codes are posted at Github repository.

5 Discussion

We used method matching and circularity to classify each image into one of the six scuba hand signs. Our model performed decently with all hand signs. However, for hand signs with similar contour, performance dropped.

We added 10 test images that was not used in calculating the circularity and creating templates to test our model for robustness. Use of circularity in ensemble model dropped the model's performance

6 Conclusions

To conclude this project, I wish I had time to implement other methods for this project. If I were to focus on high accuracy of my model, I could have chosen 4 data with critically different circularity such as rock, paper and scissor. However, Hao and I agreed to tackle a rather complicated task of 6 hand signs with similar features to apply what we learned from class. We thought combining two methods would be enough but it turns out that was not the case. Our best model is template matching algorithm.

This has been a great experience of understanding template matching method and its limitations. Our accuracy rose from 22% to 88% when we synced the resize code for image processing and template processing. We realized a vulnerability of template matching which was lack of robustness.

7 Credits and Bibliography

Cite any papers or other references you consulted while developing your solution. Credits any joint work or discussions with your classmates. Acknowledge any help by AI tools you may have received.