
# CALIFORNIA POLYTECHNIC STATE UNIVERSITY

---

CENG - DEPARTMENT OF ELECTRICAL ENGINEERING

## EE 428 - 01: Computer Vision

## Winter Quarter

## Final Project

**Report Date: 3/12/2026**

**Professor:**

Dr. Zhang

**Written By:**

Jordan Reichhardt

Bryce Watanabe

Christina Porras

Alonzo Arroyo

# Table of Contents

## Introduction:

The aim of this project is to identify hand shape-rock, paper, or scissors for hands in front of grassy and leafy distracting backgrounds. To accomplish this we implement a combination of color-based segmentation using both HSV and YCbCr techniques, followed by shaped-based feature detection. Using HSV and YCbCr color metrics we can create binary pixel maps of just the hand shape separated from their green leafy or grassy background. Then many thresholds including solidity, aspect ratio, and compactness are applied to this map to determine the most likely shape (rock, paper, or scissors) for the hand hand map.

Ultimately the project combines segmentation techniques explored in project five with feature detection from project three to improve feature detection in settings with cluttered backgrounds. The strawberry segmentation project demonstrated the usefulness of color-related spectrums in separating objects from their backgrounds. The initial rock-paper-scissors identification relied on template matching to determine the closest matches between template rock, paper, and scissors and the test images. The new feature matching script however relies on measured shape properties which create thresholds for solidity, extent, eccentricity, aspect ratio, and compactness. The new feature matching technique was implemented to combat difficulties in implementation with accuracy in detecting hand shape using the template matching technique for hand shape detection.

## Motivation:

Hand gesture recognition is an important problem in computer vision because it enables more natural human-computer interaction in applications such as robotics and assistive technology. Recognizing simple gestures like rock, paper, and scissors may seem straightforward in a controlled environment, but performance becomes difficult when the background contains many different colors, textures and shapes.

This project was motivated by the challenge of improving gesture recognition in more realistic and cluttered scenes. In project 3, our template-matching approach showed limitations when hand shapes varied slightly in appearance. To address these limitations, this project combines color-based segmentation techniques with shape-based feature analysis, allowing for the hand to be classified more reliably.

## Problem Statement:

The goal of this project is to develop a computer vision system that can correctly identify whether a hand in an image is showing rock, paper, or scissors, even when it appears in front of a complex background. The primary difficulty is that backgrounds may contain colors and textures that overlap with the hand region, making classification unreliable.

To address this problem, the system must separate hand pixels from the background using color-based segmentation in HSV and YCbCr color spaces. After isolating the hand as a binary mask, the system must

then analyze shape characteristics such as solidity, extent, eccentricity, compactness, aspect ratio, and radial signature peaks to determine the most likely gesture classification.

## Overall system flowchart:

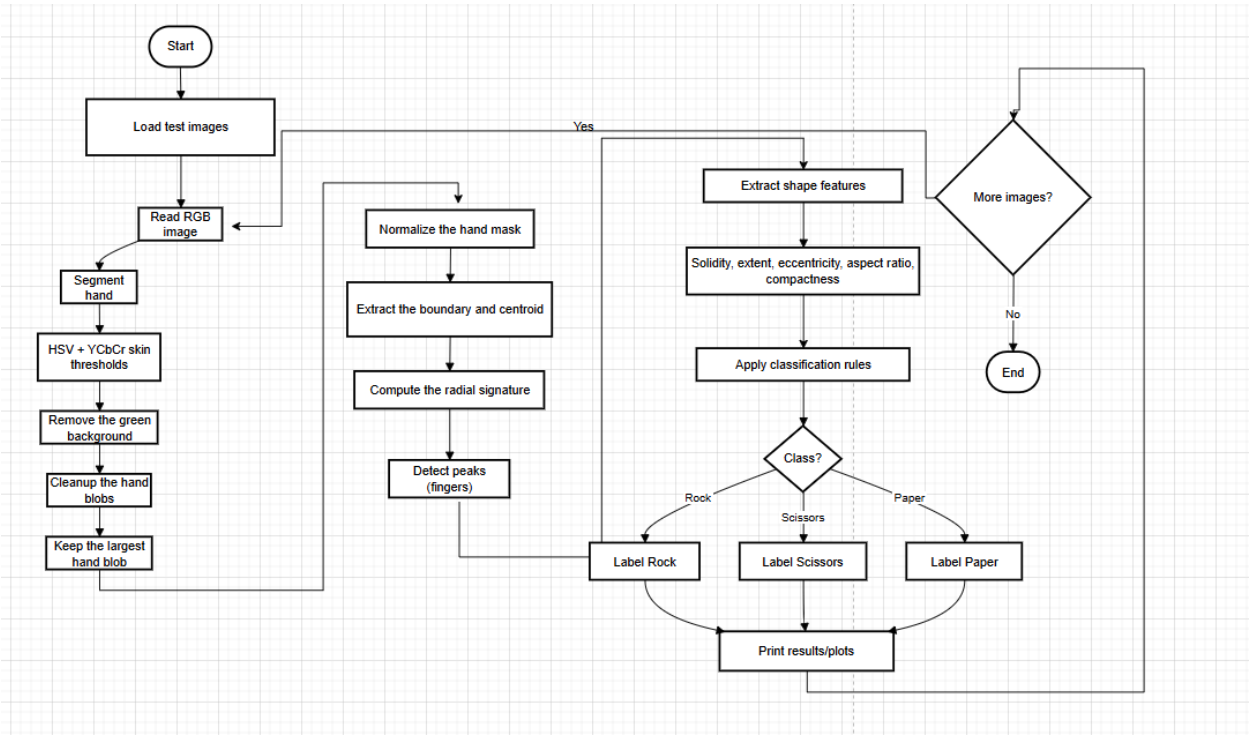



Figure 1.1. Flowchart of segmentation and feature extraction steps

Figure 1.1 illustrates the overall processing pipeline used in this project, beginning with image acquisition, followed by segmentation, feature extraction, and final classification.

# Part 1: HSV and YCbCr

## Introduction

To detect the hand in each image, different color spaces were explored to separate the hand from the background. The two main methods that were used were HSV and YCbCr. Both of these represent color differently than the standard RGB format and can make segmentation easier depending on the situation.

The HSV (Hue, Saturation, Value) color space separates color from brightness. Hue represents the type of color, saturation shows how strong the color is, and value represents how bright the image is. This method was used in previous lab strawberry detection, where it performed well because the object had a strong and consistent color. For this project, HSV was tested to see if similar thresholds could be used to detect skin.

The YCbCr color space separates brightness and color differently, with the Y channel representing brightness and the Cb and Cr channels representing color. This makes it less sensitive to lighting changes and more focused on color information. Skin tones tend to fall within a specific range in the Cb and Cr channels, therefore making YCbCr a good candidate for detecting a hand across different images.

## Procedure/Implementation

To evaluate which color space works best for hand detection, images of rock, paper, and scissors gestures were collected and resized to a common size. Each image was then converted from RGB into HSV and YCbCr color spaces. A polygon was manually drawn around the hand to create a mask, separating hand pixels from the background. Pixel values from both the hand and background were collected. For HSV, the hue, saturation, and value channels were stored, while for YCbCr, the Cb and Cr channels were used. These values were combined across all images to build a dataset representing both hand and background regions.

The data was then visualized using histograms and scatter plots to compare how hand and background pixels were distributed. HSV used H-S plots, while YCbCr used Cb-Cr plots. These visualizations helped identify which ranges best separated the hand from the background. From this, threshold values were selected for segmentation.

```
I = im2double(I0);

HSV = rgb2hsv(I);
YCbCr = rgb2ycbcr(I);

H = HSV(:,:,1);
S = HSV(:,:,2);
V = HSV(:,:,3);
Cb = YCbCr(:,:,2);
```

```
Cr = YCbCr(:,:,3);
```

Figure 1.2: Each image is converted from RGB into HSV and YCbCr color spaces

```
imshow(I0)
title("Draw polygon around HAND")

h = drawpolygon;
handMask = createMask(h);
bgMask = ~handMask;
```

Figure 1.3: A polygon is manually drawn around the hand to create a mask

```
allHandYCbCr = [allHandYCbCr; Cb(handMask), Cr(handMask)];
allBgYCbCr = [allBgYCbCr; Cb(bgMask), Cr(bgMask)];

histogram(allHandYCbCr(:,1),50,'FaceAlpha',0.5)
hold on
histogram(allBgYCbCr(:,1),50,'FaceAlpha',0.5)
```

Figure 1.4: Pixel values from the hand and background are collected and visualized using histograms

## Results / Conclusion

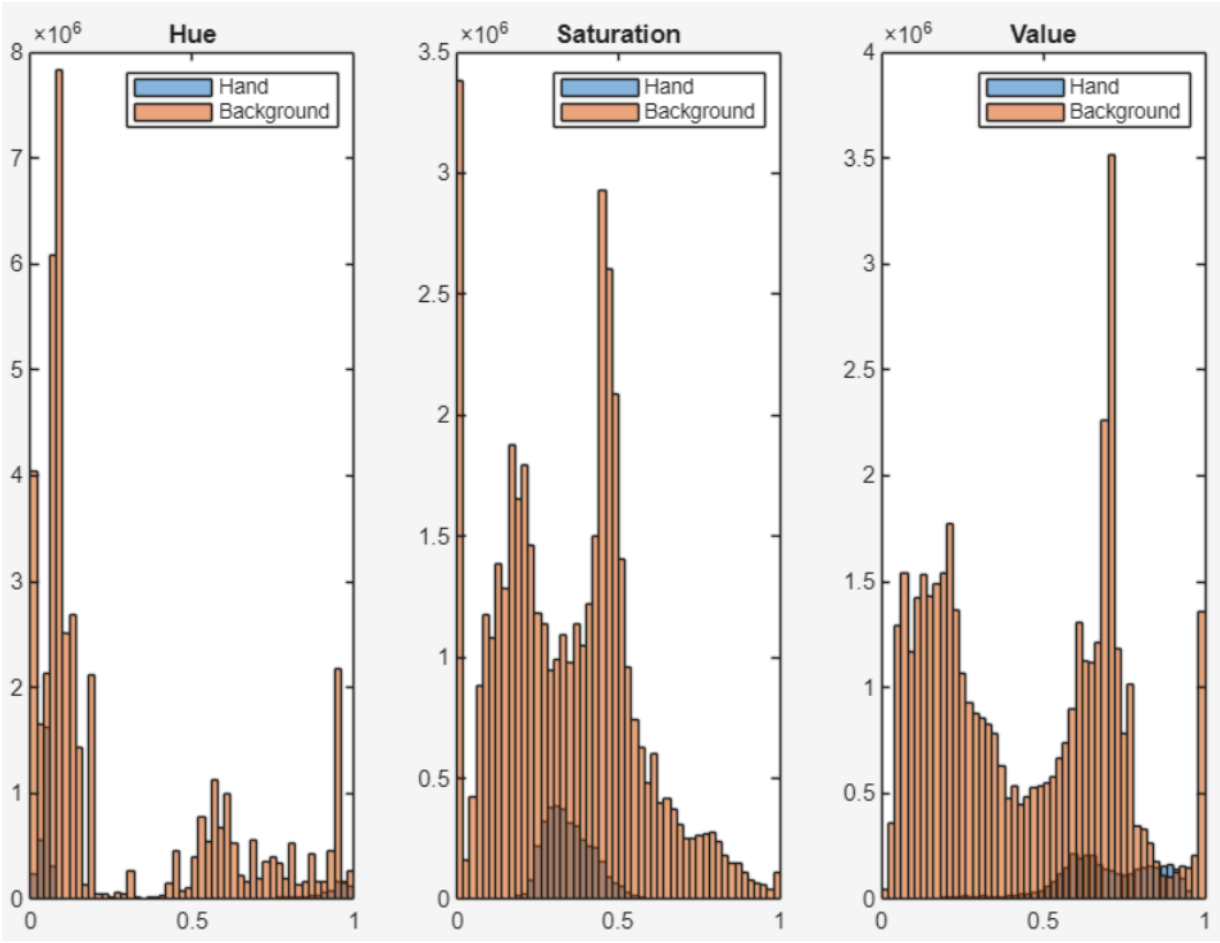



Figure 1.5: distribution of hand and background pixels across Hue, Saturation, and Value channels

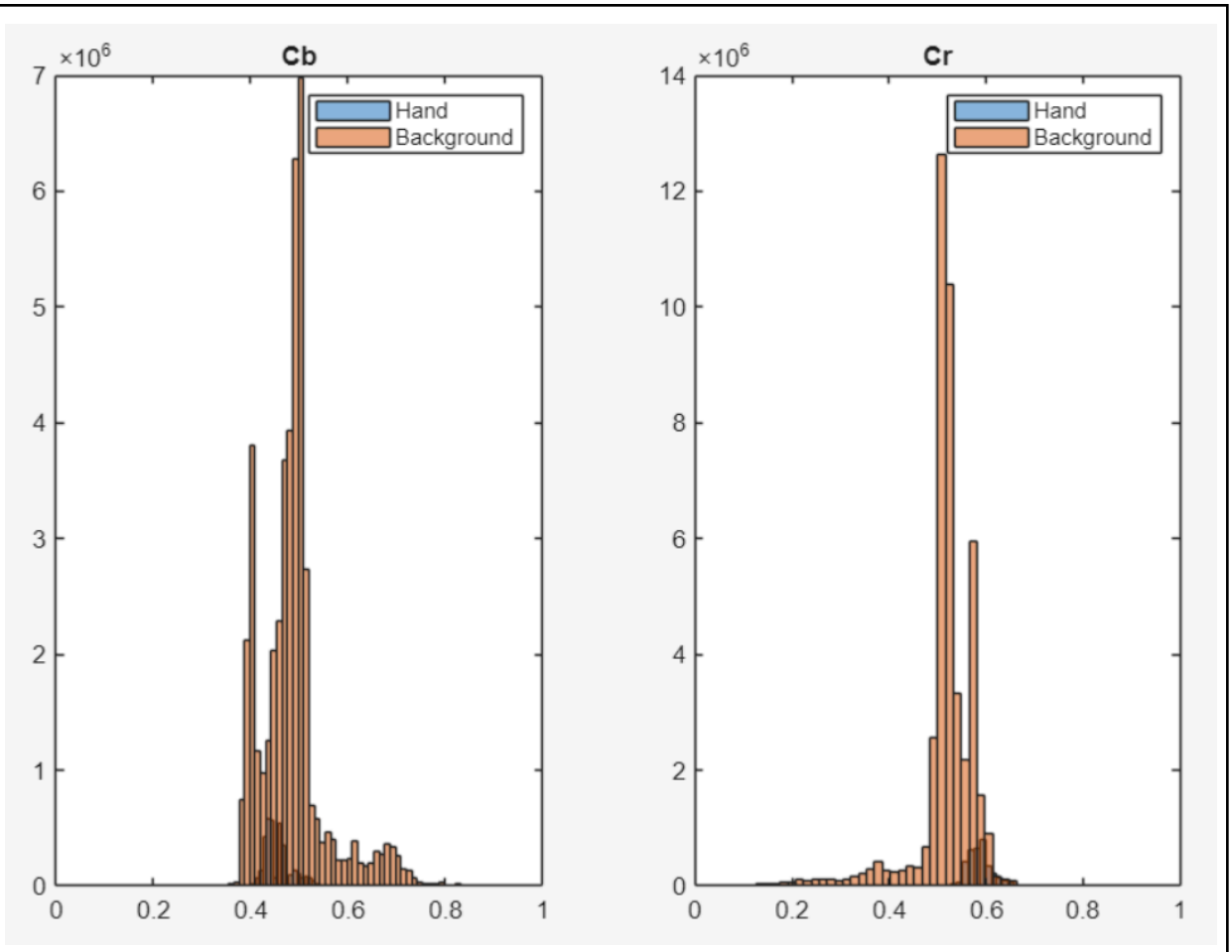



Figure 1.6: This figure shows the distribution of hand and background pixels in the Cb and Cr channels

From the graphs, the background tends to have higher brightness values than the hand, making a brightness threshold necessary to reduce false detections. In HSV, there is significant overlap between hand and background, making it harder to separate them cleanly. The HSV thresholds used were: H: 0.00–0.16, S: 0.10–0.80, V: 0.16–0.99. In YCbCr, the separation between hand and background is about the same as HSV with the thresholds being  Cb: 88/255–118/255, Cr: 140/255–166/255, and Y: 0.25–0.85. Putting these methods into actual use YCbCr produced more accurate results, so it was chosen as the main method for segmentation.

# Part 2: Segmentation

## Introduction

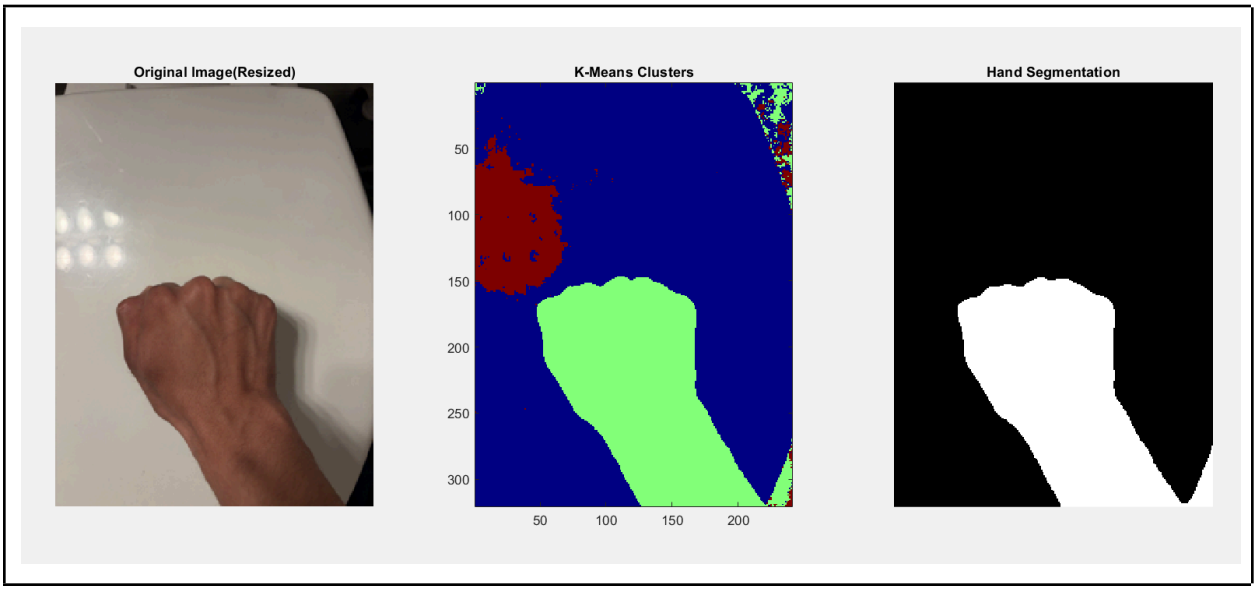
After determining what color space values best distinguish the hand from the background, a segmentation pipeline was developed to isolate the hand from the background using K-means, YCbCr thresholding, and other methods to clean up noise. The saturation and Cb Cr values were used independently for K-means. The clustering was first applied to group pixels into four color clusters. The cluster most similar to skin color was selected using RGB values, producing an initial mask. However, K-means alone often included background regions with similar colors.

A YCbCr mask was then created using the selected Cb, Cr, and Y thresholds. This method provided more consistent skin detection but could still include noise depending on lighting. To improve results, both methods were compared. If the YCbCr mask closely matched the K-means result, it was used alone since it was cleaner. Otherwise, a combined mask was used to retain missing parts of the hand

.

## Procedure/Implementation

The first iteration of segmentation attempted to use k-means with saturation as the feature vector. Post processing follows the clustering algorithm by applying morphological operations and then selecting the biggest blob as the hand. The k-means clustering in the saturation space yielded acceptable results for simple backgrounds, and distinct textured backgrounds. A k value of 3 was used to account for the hand, the hand's shadow, and the background.



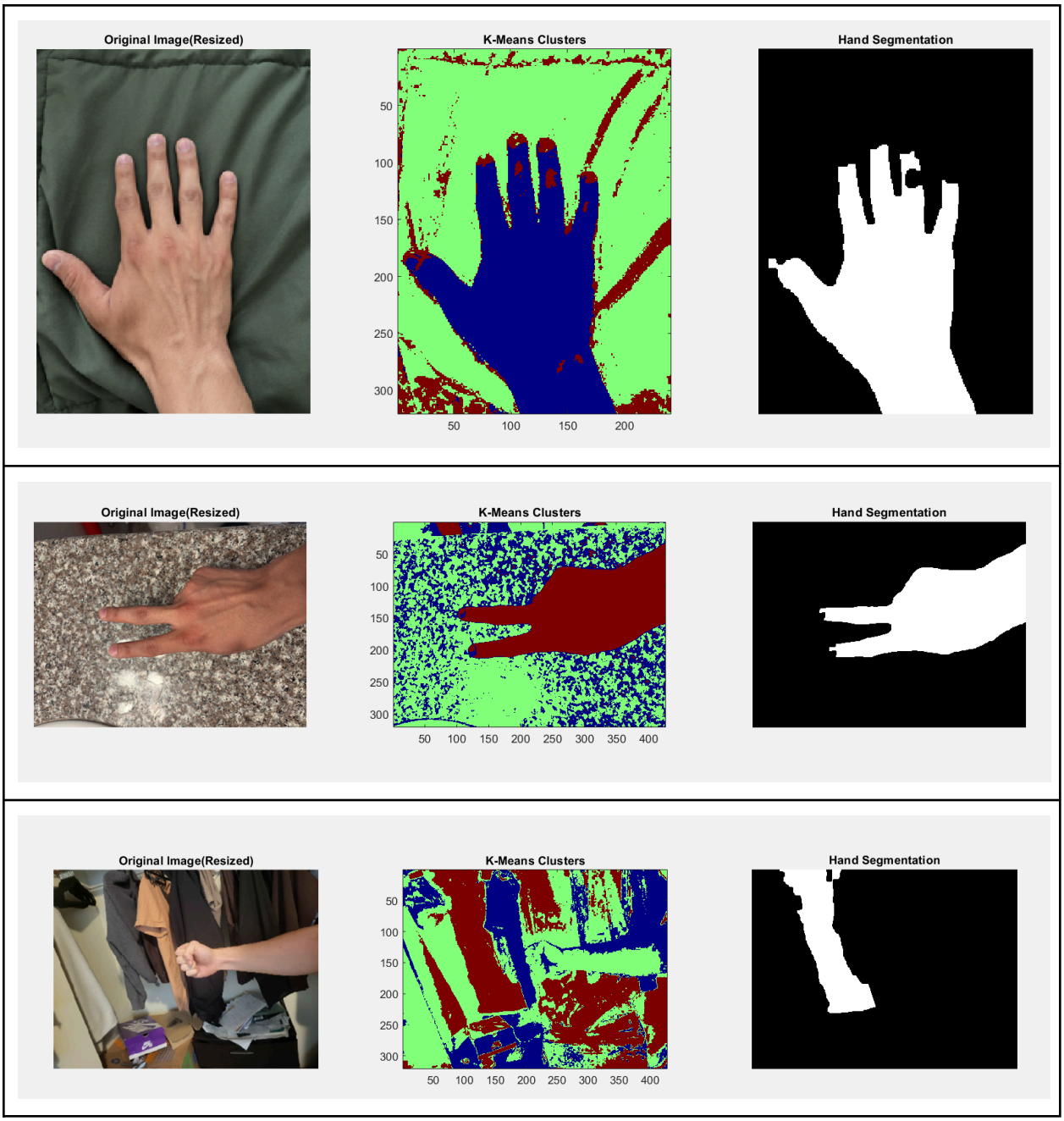



Figure 2. K means segmentation to obtain a hand mask of gestures with varying background colors and textures.

The K means segmentation in the saturation color space in conjunction with post processing morphological operations provides adequate segmentation for feature detection when there is a consistent background, regardless of color and texture. However, it is also shown that with a clustered background, the saturation based K means clustering alone does not segment the hand properly. A different segmentation technique will need to be used.

The segmentation was designed to reliably isolate the hand from the background using a combination of clustering, color thresholding, and post-processing. First, K-means clustering was applied to the image by reshaping it into a list of RGB pixels and grouping them into four clusters. Since K-means does not label clusters, a scoring method based on RGB values was used to select the cluster most likely to represent skin. This produced an initial mask of the hand, but it often included unwanted background regions.

Next, a YCbCr-based mask was created using the threshold values determined from earlier analysis. This mask focused on the Cb and Cr channels, along with a brightness constraint using the Y channel, to better isolate skin-colored regions.

The two masks were then compared. If the YCbCr mask closely matched the K-means result (based on overlap), the YCbCr mask alone was used since it was typically cleaner. If not, a combined mask was created by keeping only the K-means regions that overlapped with the YCbCr mask.

After this, k-means operations were applied to refine the result. Opening removed small noise, closing helped connect nearby regions, and hole filling ensured the hand region was solid. A final size and shape-based filtering(aspect ratio) step was then applied to remove remaining unwanted blobs and keep the most likely hand region.

```
[idx, C] = kmeans(X, 4, 'Replicates', 3);
seg = reshape(idx, rows, cols);

bestCluster = 1;
bestScore = -inf;

for k = 1:4
   R = C(k,1);
   G = C(k,2);
   B = C(k,3);

   score = (R - B) + 0.5*(R - G);

   if score > bestScore
      bestScore = score;
      bestCluster = k;
   end
end

maskKmeans = seg == bestCluster;
```

Figure 3.1: K-means clustering was used to divide the image into color-based regions. The cluster with the most skin-like average RGB values was selected as the initial hand mask.

```
YCbCr = rgb2ycbcr(I);
Y = YCbCr(:,:,1);
Cb = YCbCr(:,:,2);
```

```
Cr = YCbCr(:,:,3);

maskYCbCr = (Cb >= 88/255 & Cb <= 118/255) & ...
(Cr >= 140/255 & Cr <= 166/255) & ...
(Y >= 0.25 & Y <= 0.85);
```

Figure 3.2: A second mask was created using threshold values in the YCbCr color space. This helped isolate skin-colored regions while also limiting rightness to reduce background noise

```
YCbCr = rgb2ycbcr(I);
Y = YCbCr(:,:,1);
Cb = YCbCr(:,:,2);
Cr = YCbCr(:,:,3);

maskYCbCr = (Cb >= 88/255 & Cb <= 118/255) & ...
(Cr >= 140/255 & Cr <= 166/255) & ...
(Y >= 0.25 & Y <= 0.85);
```

Figure 3.3: The final mask was chosen by comparing how closely the YCbCr mask matched the combined mask. If the overlap was high, the cleaner YCbCr mask was used; otherwise, the combined mask was kept to preserve more of the hand region.

```
maskKmeans = imopen(maskKmeans, strel('disk',3));
maskKmeans = imclose(maskKmeans, strel('disk',7));
maskKmeans = imfill(maskKmeans,'holes');
maskKmeans = bwareafilt(maskKmeans, 1);
```

Figure 3.4: The K-means mask is refined by removing noise, connecting regions, filling holes, and keeping only the largest detected objects to better isolate the hand.

## Results / Conclusion

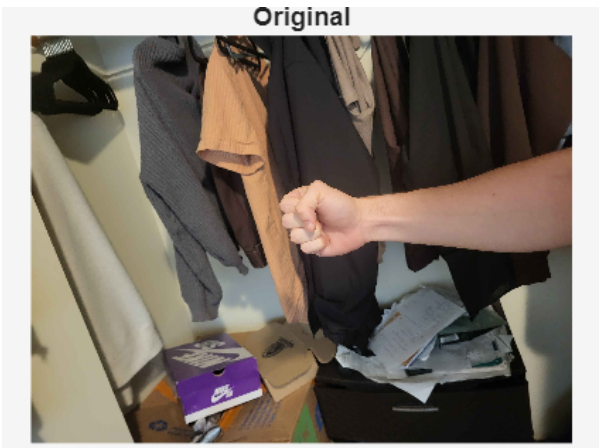
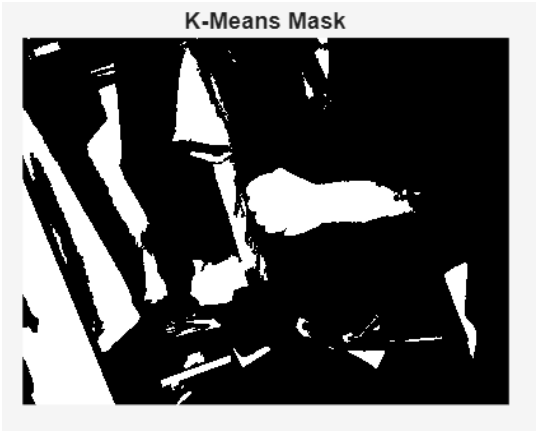




Figure 3.5: As shown in the HSV mask, there is too much background noise, making it unreliable for accurate hand detection



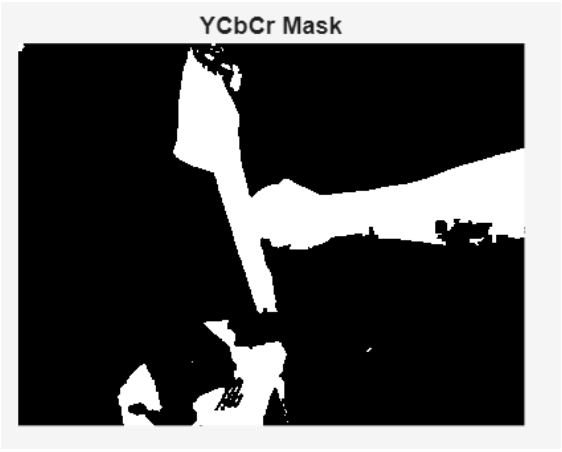




Figure 3.6: As shown, k means or YCbCr are not enough to create a solid mask

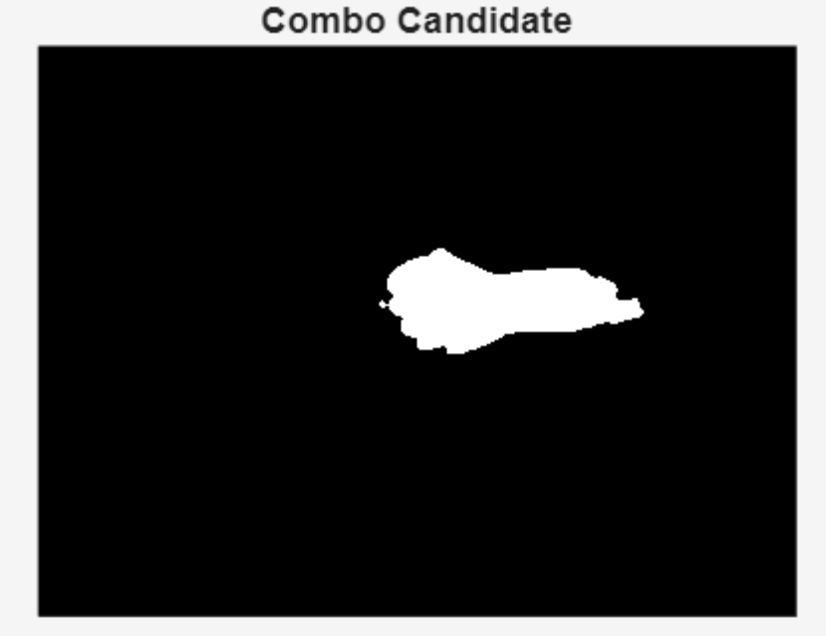



Figure 3.7: After combining all methods, the result is a clean and reliable mask of the hand, even with background noise.

HSV and YCbCr color spaces were explored to separate the hand from the background. HSV showed significant overlap and was not reliable on its own, while YCbCr provided more consistent separation due to better handling of color and brightness. K-means clustering helped group pixels but often included background noise, and YCbCr alone could miss parts of the hand. By combining both methods with image cleanup steps and blob filtering, a more accurate mask was achieved. Overall, using YCbCr as the main method with K-means as support produced the most reliable hand segmentation across different images.

# Part 3: Feature detection/ classification

## Introduction

Feature detection is a key component of shape detection in computer Vision projects such as the rock-paper-scissors detection in this project. The threshold based system used in this project relies on geometric structural features of the shapes rather than using raw pixel data. While previous techniques such as template matching created many false positives and false negatives and required manual cropping, using geometrics markers is automated and tends to be more robust to scale, small segmentation noise, and rotation.

In this project two main structural components are relied upon, these include radial signature and region-based shape descriptors. The radial signature indicates where a given number of evenly spaced points on the hand boundary are relative to the center of the hand shape. The shape features including solidity, eccentricity, compactness, extent, and aspect ratio characterize the structure of the hand allowing thresholds to be created for each category.

## Procedure/Implementation

After segmentation, the binary hand mask is normalized to a fixed size so that all shapes can be compared consistently.

```
normSize = [240 240];
handMask = normalize_mask(handMask, normSize);
```

The boundary of the hand and its centroid are then extracted. The largest boundary is selected and resampled to ensure a consistent number of points for all images.

```
B = bwboundaries(mask, 'noholes');
stats = regionprops(mask, 'Centroid');
cx = stats(1).Centroid(1);
cy = stats(1).Centroid(2);
```

A radial signature is computed by measuring the distance from the centroid to each boundary point. This converts the hand shape into a one-dimensional signal.

```
dx = boundaryXY(:,1) - cx;
dy = boundaryXY(:,2) - cy;
rSig = sqrt(dx.^2 + dy.^2);
```

The radial signature is smoothed to reduce noise, and peaks are detected to estimate the number of fingers.

```
rSmooth = movmean(rSig, 9);
rSmooth = movmean(rSmooth, 9);

[peakVals, peakIdx] = findpeaks(rSmooth, ...
  'MinPeakProminence', prom, ...
  'MinPeakDistance', minDist);
```

In addition to peak detection, region-based shape features are extracted using MATLAB functions.

```
stats = regionprops(mask,'Solidity','Extent','Eccentricity', ...
  'Area','BoundingBox','Perimeter');
```

From these, aspect ratio and compactness are computed to further describe the hand shape.

```
bb = stats(1).BoundingBox;
aspectVal = bb(3) / bb(4);

P = stats(1).Perimeter;
compactVal = (P^2) / (4*pi*areaVal);
```

The gesture is then classified using rule-based thresholds. Rock is identified by low peak count, high solidity, and low compactness.

```
isRock = (numPeaks <= 1) && (solidityVal > 0.78) && (compactVal < 3.1);
```

Scissors is identified by 2–3 peaks and high eccentricity, with additional rules to handle merged fingers.

```
isScissors = (numPeaks >= 2 && numPeaks <= 3) && (eccentricityVal > 0.88);
```

Paper is assigned when the hand does not meet rock or scissors conditions and shows multiple peaks or high variability.

```
isPaper = (~isRock) && (~isScissors) && (numPeaks >= 3);
```

Finally, a label is assigned based on the detected features.

```
if isRock
   label = 'Rock';
elseif isScissors || isScissorsMerged
   label = 'Scissors';
else
   label = 'Paper';
end
```

This process converts the segmented hand into radial and geometric features, allowing for reliable classification even when segmentation noise or finger merging occurs.

| Feature | Rock (Typical) | Rock (abnormal) | Scissors (normal) | Scissors (merged finger) | Paper (normal) | Paper (merged fingers) |
|---|---|---|---|---|---|---|
| numPeaks | ≤ 1 | ≤ 1 | 2 – 3 | 1 | ≥ 3 | 2 or 1 (missed |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | fingers) |
| Solidity | > 0.78 | > 0.80 | < 0.82 | < 0.70 | < 0.86 | 0.65 – 0.85 |
| Extent | 0.40 – 0.70 | > 0.40 | < 0.42 | 0.24 – 0.36 | 0.35 – 0.55 | > 0.35 |
| Eccentricity | 0.60 – 0.85 | < 0.88 | > 0.88 | > 0.88 | 0.70 – 0.90 | < 0.90 |
| Compactness | < 3.1 | < 2.8 | 3 – 6 | > 6.0 | > 3.2 | > 3.0 |
| Aspect Ratio | 0.8 – 1.5 | 0.8 – 1.6 | 0.7 – 1.8 | 0.90 – 1.65 | 0.8 – 1.7 | Wide range |
| Radial Std Dev | < 20 | < 20 | 15 – 30 | > 20 | > 26 | > 22 |

Table 3.1. Radial signature (numPeaks), and geometric shape thresholds

## Results:

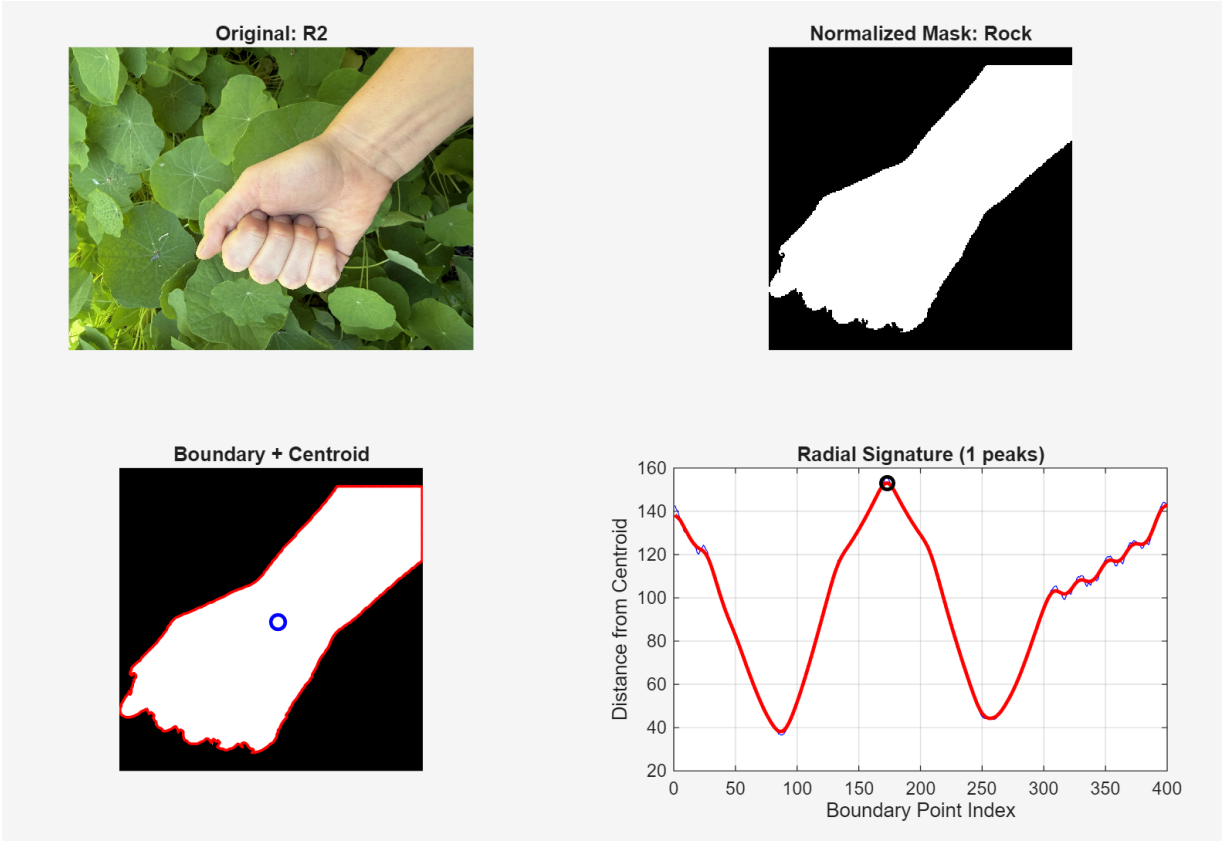



Figure 4.1. Radial Signature of rock mask

Figure 3.1 shows the pipeline for a rock gesture, including the original image, normalized mask, extracted boundary with centroid, and the resulting radial signature. The radial signature exhibits minimal variation in distance from the centroid to the boundary. Since the hand is closed, there are no clearly defined finger extensions resulting in zero to one detected peak.

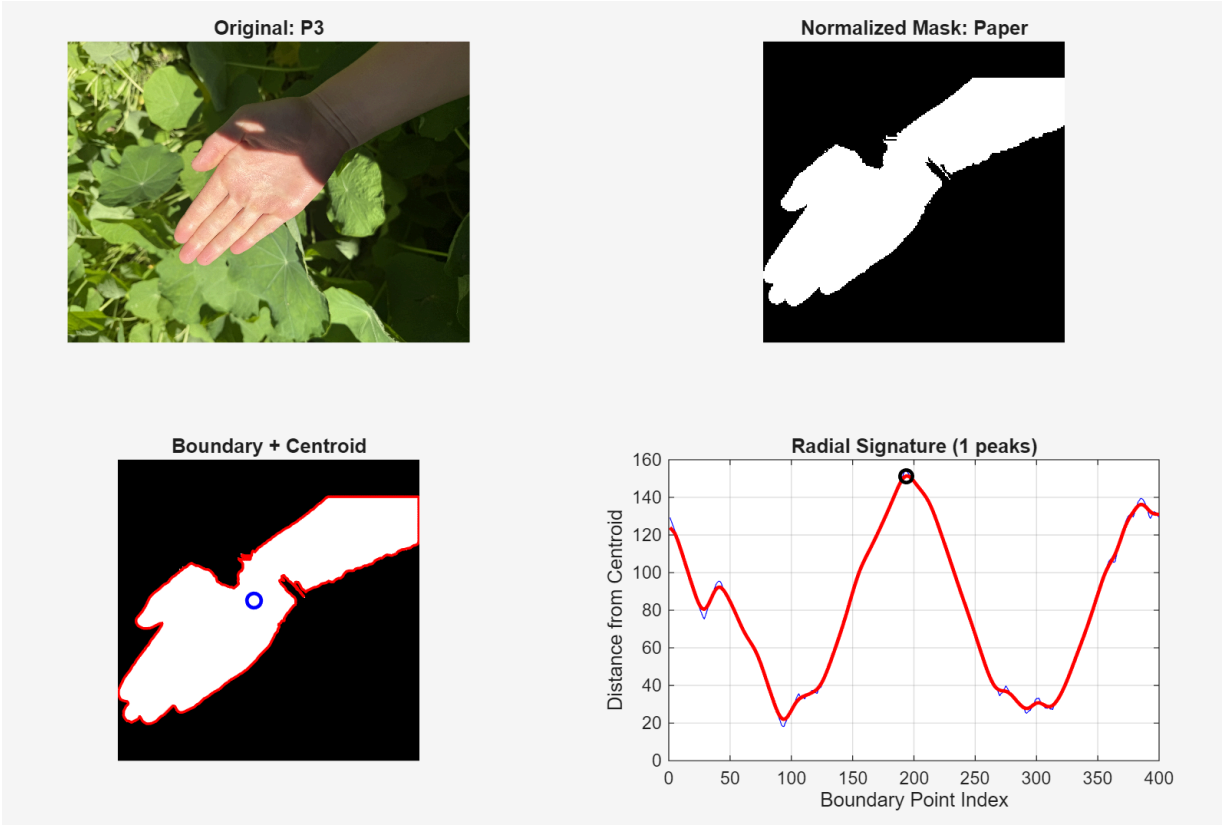



Figure 4.2. Radial Signature of paper mask

Figure 4.2 illustrates the processing steps for a paper gesture. Unlike rock, the radial signature shows larger fluctuations in distance, corresponding to multiple extended fingers. Ideally, paper should produce several distinct peaks, however in this example only one dominant peak is detected. This could occur because of some fingers being merged in the segmentation mask. Despite this reduced peak count, the signal still shows high variability, as  seen in the wider range and higher standard deviation.

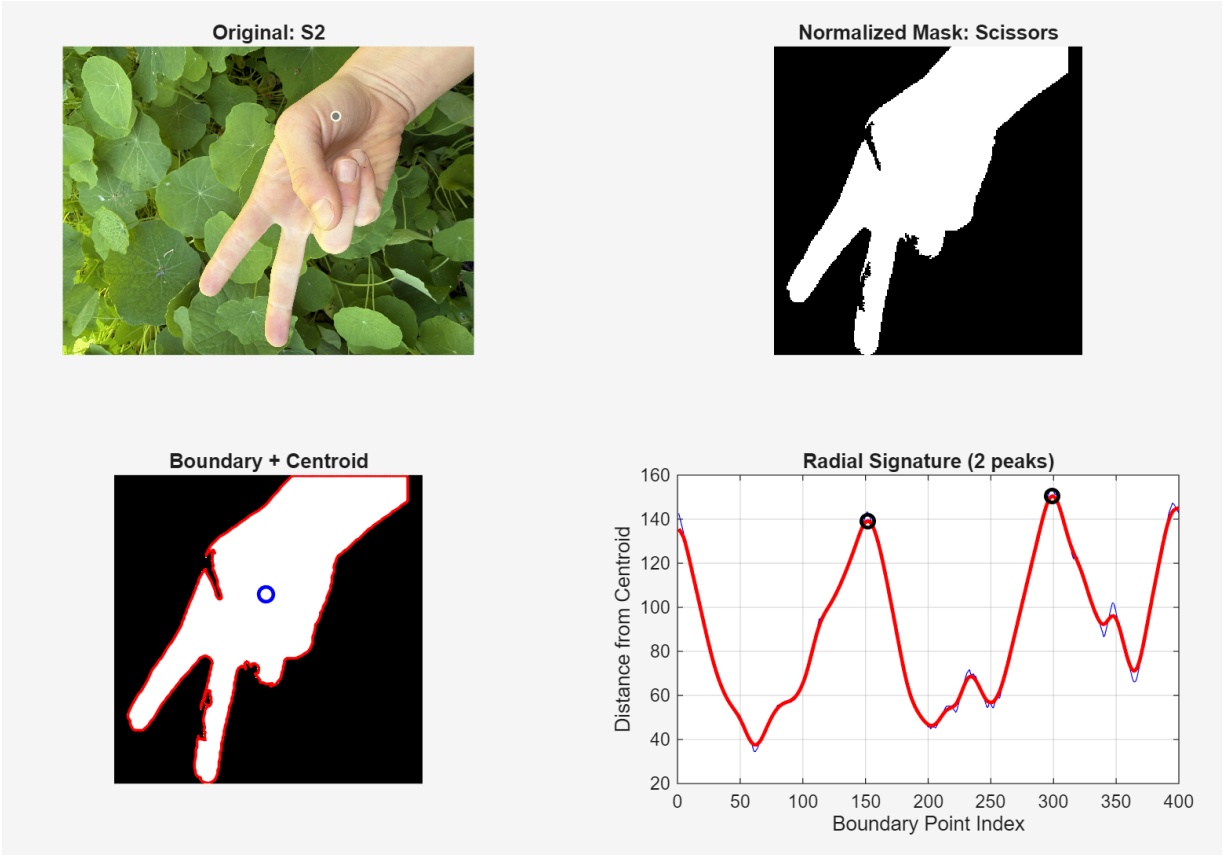
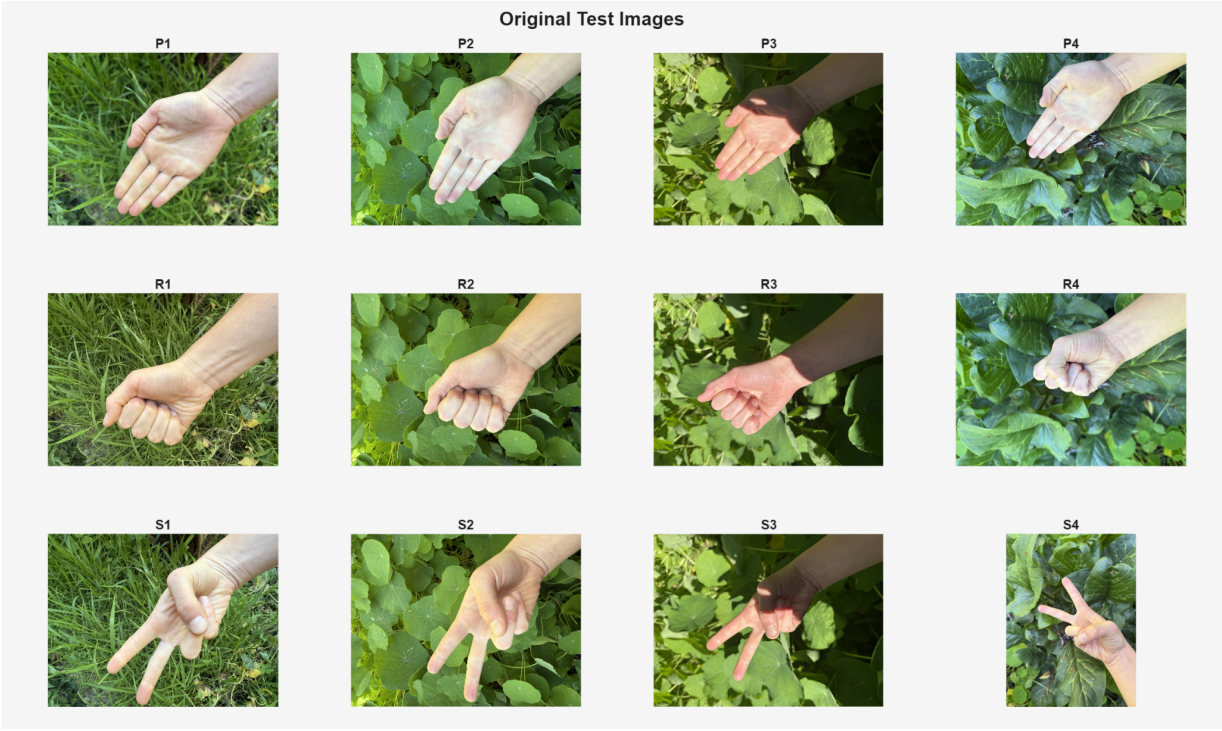



Figure 3.3. Radial Signature of scissors mask

Figure 3.3 shows the feature extraction results for a scissors gesture. The radial signature shows two prominent peaks, corresponding to the two extended fingers. These peaks are clearly separated and more pronounced than in the paper case, making them a strong indicator for the scissors classification. In addition to the peak count, the elongated shape of the hand is evident in the boundary and centroid visualization, and is supported by a high eccentricity value.




Hand Identifications

---

P1.jpeg -> Paper
Radial Peaks=1 Mean=89.92 Std=28.66 Range=118.50
Shape: Sol=0.795 Ext=0.418 Ecc=0.883 Area=18877 Aspect=1.277 Compact=4.505
P2.jpeg -> Paper
Radial Peaks=3 Mean=73.32 Std=25.72 Range=115.31
Shape: Sol=0.605 Ext=0.376 Ecc=0.816 Area=16770 Aspect=0.775 Compact=13.225
P3.jpeg -> Paper
Radial Peaks=1 Mean=80.44 Std=39.27 Range=129.35
Shape: Sol=0.805 Ext=0.398 Ecc=0.959 Area=17395 Aspect=1.319 Compact=3.396
P4.jpeg -> Paper
Radial Peaks=1 Mean=68.38 Std=38.30 Range=128.46
Shape: Sol=0.451 Ext=0.238 Ecc=0.974 Area=6613 Aspect=2.069 Compact=13.242
R1.jpeg -> Rock
Radial Peaks=1 Mean=96.26 Std=35.92 Range=121.02
Shape: Sol=0.832 Ext=0.442 Ecc=0.945 Area=22056 Aspect=1.154 Compact=1.969
R2.jpeg -> Rock
Radial Peaks=1 Mean=98.68 Std=33.05 Range=114.87
Shape: Sol=0.866 Ext=0.471 Ecc=0.931 Area=23963 Aspect=1.132 Compact=2.002
R3.jpeg -> Rock
Radial Peaks=1 Mean=86.92 Std=35.23 Range=113.53
Shape: Sol=0.820 Ext=0.378 Ecc=0.963 Area=16519 Aspect=1.319 Compact=2.297
R4.jpeg -> Rock
Radial Peaks=2 Mean=97.19 Std=14.16 Range=59.64

```
  Shape: Sol=0.931  Ext=0.798  Ecc=0.690  Area=28368  Aspect=1.462  Compact=1.526
S1.jpeg -> Scissors
  Radial Peaks=1  Mean=73.77  Std=27.07  Range=108.02
  Shape: Sol=0.612  Ext=0.308  Ecc=0.916  Area=12204  Aspect=1.455  Compact=9.601
S2.jpeg -> Scissors
  Radial Peaks=2  Mean=88.87  Std=32.22  Range=112.91
  Shape: Sol=0.735  Ext=0.356  Ecc=0.926  Area=18707  Aspect=0.912  Compact=4.608
S3.jpeg -> Scissors
  Radial Peaks=2  Mean=87.70  Std=34.94  Range=121.14
  Shape: Sol=0.724  Ext=0.306  Ecc=0.962  Area=14558  Aspect=1.212  Compact=3.402
S4.JPG -> Scissors
  Radial Peaks=2  Mean=68.70  Std=37.26  Range=125.54
  Shape: Sol=0.384  Ext=0.111  Ecc=0.993  Area=3143  Aspect=0.500  Compact=9.865
-----------------------------------------------------------------------------------------------------------------
```

Figure 4.4. Original hand images and final identification

Figure 4.4 summarizes the final classification results across all test images. Each image is processed through segmentation, normalization, radial signature computation, and feature-based detection. The radial peaks, mean, standard deviation, and geometric features demonstrate how each decision was made. The results show that the classification of all gestures was successful, even in challenging cases such as paper with merged fingers, scissors with one detected peak, and rock with minor irregularities.

### Conclusion:

The feature detection used in this rock, paper, scissors detection effectively captures the parameters of hand shapes in order to correctly identify them. In combining radial signature and geometric shape descriptors the algorithm can detect key features such as number of fingers and compactness of a shape which are unique to each hand configuration.

## Overall Conclusion:

Within the main code we are performing image segmentation based on various thresholds of Hue Value and Saturation. This is used first to differentiate the hand pixels from their noisy leafy green backgrounds so that before classifying the type of hand we can determine whether the object even qualifies as a hand. This HSV classification is similar to the processes used in project 5 where we determined whether pixel clusters were strawberries based on their red intensities. Additionally in this rock, paper scissors project we also create binary hand masks similar to the binary strawberry masks in project 5. In this rock paper

scissors classification, however, we then use additional feature thresholding to classify each mask as a rock, paper, or scissors shape.

Within the segmentation section we explored other methods of segmentation beyond HSV and YCbCr methods. Here we used a combination of both K-means and HSV segmentation to generate masks for background that were similar to the hand color. While we were able to generate some clean hand masks with this method, many of them still had noise which became problematic in the feature detection stage.

The feature detection stage was composed of two methods. First is finding the radial signature from the center of the hand to determine the amount of "peaks"of the hand. This works well when the paper and scissors gestures have their fingers spread out, as counting the peaks of the radial signature is crucial for classification. The next part of detection was analyzing the shape properties of the hand gesture, with rock, paper, and scissors having certain structures that make them distinct from each other regardless of the finger spread. Classification uses these detected features as criteria for each rock paper and scissors.

## References:


[1] MathWorks, "bwboundaries — Trace object boundaries in binary image," MathWorks Documentation.

[2] MathWorks, "regionprops — Measure properties of image regions," MathWorks Documentation.

[3] MathWorks, "findpeaks — Find local maxima," MathWorks Documentation.

[4] MathWorks, "interp1 — 1-D data interpolation," MathWorks Documentation.

[5] MathWorks, "movmean — Moving average," MathWorks Documentation.

[6] MathWorks, "rgb2hsv — Convert RGB image or colormap to HSV," MathWorks Documentation.

[7] MathWorks, "rgb2ycbcr — Convert RGB image or colormap to YCbCr," MathWorks Documentation.

[8] P. Trigueiros, F. Ribeiro, L. P. Reis, and A. P. Moreira, "A Comparative Study of Different Image Features for Hand Gesture Recognition," in *Proc. Int. Conf. on Agents and Artificial Intelligence*, 2013.

[9] S. Kolkur, D. Kalbande, P. Shimpi, C. Bapat, and J. Jatakia, "Human Skin Detection Using RGB, HSV and YCbCr Color Models," *arXiv preprint* arXiv:1708.02694, 2017.

[10] Y. Morgenstern, E. K. H. Salge, and V. F. Pauwels, "One-shot categorization of novel object classes in humans," *Vision Research*, vol. 165, pp. 146–162, 2019.


## Contributions:

| | | | |
|---|---|---|---|
| Bryce Watanabe | Jordan Reichardt | Christian Porras | Alonzo Arroyo |

| | | | |
|---|---|---|---|
| Motivation, problem statement, part 3 procedure/implementation, part 3 results writeup. I experimented with using SIFT and RANSAC to locate the hand, and then applying segmentation and shape analysis to classify the gesture. This approach was not robust and showed keypoint-based methods were not better for this application than using what we explored in this project. | I worked on the main code with, which mostly added a new feature detection/ classification technique. While the segmentation with K-means and HSV was not able to be effectively combined with the feature detection, the segmentation using HSV and YCbCr thresholding proved to be effective on a noisy but similar color spectrum background. I also wrote the feature matching portion of the report which describes the techniques used for both radial detection and geometric shape classification of hand types. | I worked on the color analysis and segmentation part of the code. The method worked really well when the background had strong contrast with the hand, and it still did okay in some lower-contrast situations. However, it started to struggle when the background had similar colors to the hand. Unfortunately we had to stick with a simpler masking due to not being able to get it to work with the main code. I also worked on both Part 1 and Part 2 of the project documentation. | Experimented with different color spaces and feature vectors to determine which provides the best K-means clustering and overall segmentation. The saturation was found to be the best feature space to work in for solely k means, but fails when backgrounds were more complex. Contributed to overall formatting and adding figure captions. Wrote part of the overall conclusion. |

# Appendix

## Main

```
% EE428 Final Project main script
%References used to make the code and listed in the comments:
% [1] MathWorks, "bwboundaries - Trace object boundaries in binary image."
% [2] MathWorks, "regionprops - Measure properties of image regions."
% [3] MathWorks, "findpeaks - Find local maxima."
% [4] MathWorks, "interp1 - 1-D data interpolation."
% [5] MathWorks, "movmean - Moving average."
% [6] MathWorks, "rgb2hsv - Convert RGB image or colormap to HSV."
% [7] MathWorks, "rgb2ycbcr - Convert RGB image or colormap to YCbCr."
% [8] P. Trigueiros, F. Ribeiro, L. Paulo Reis, and A. P. Moreira,
%     "A Comparative Study of Different Image Features for Hand Gesture
```

```
%     Recognition," Proc. Int. Conf. on Agents and Artificial Intelligence,
%     2013.  (centroid distance / radial signature geometric basis)
% [9] S. Kolkur, D. Kalbande, P. Shimpi, C. Bapat, and J. Jatakia,
%     "Human Skin Detection Using RGB, HSV and YCbCr Color Models,"
%     arXiv:1708.02694, 2017.  (HSV / YCbCr skin-segmentation basis)
%[10] Y. Morgenstern, E. K. H. Salge, and V. F. Pauwels,
% "One-shot categorization of novel object classes in humans,"
% Vision Research, vol. 165, pp. 146–162, 2019.
clc; clear; close all;
folder = 'C:\Users\jorda\Documents\Masters program\Winter_Masters\EE428\Final Project';
%files to classify hand type
testFiles = {
   pick_file(folder,'P1')
   pick_file(folder,'P2')
   pick_file(folder,'P3')
   pick_file(folder,'P4')
   pick_file(folder,'R1')
   pick_file(folder,'R2')
   pick_file(folder,'R3')
   pick_file(folder,'R4')
   pick_file(folder,'S1')
   pick_file(folder,'S2')
   pick_file(folder,'S3')
   pick_file(folder,'S4')
};
%make each mask the same size
normSize = [240 240];
%store original images for final display window
numFiles = length(testFiles);
resultImgs = cell(numFiles,1);
resultNames = cell(numFiles,1);
fprintf('\nHand Identifications\n');
fprintf('-----------------------------------------------------------------------------------------------------------\n');
for k = 1:length(testFiles)
   filePath = testFiles{k};
   [~, name, ext] = fileparts(filePath);
   I = imread(filePath);
   resultImgs{k} = I;
   resultNames{k} = name;
  %make hist. plots
   showDebugHists = strcmpi(name, 'P1') || strcmpi(name, 'S1');
   if showDebugHists
      Idbg = im2double(I);
      hsv_dbg = rgb2hsv(Idbg); % [6]
      h_dbg = hsv_dbg(:,:,1);
      s_dbg = hsv_dbg(:,:,2);
      v_dbg = hsv_dbg(:,:,3);
      YCbCrdbg = rgb2ycbcr(Idbg); % [7]
      Cb_dbg = YCbCrdbg(:,:,2);
```

```
    Cr_dbg = YCbCrdbg(:,:,3);
    % HSV / YCbCr skin-threshold idea follows common skin-color
    % this segmentation practice is explained in [9]
    skinHSVdbg = (h_dbg > 0.00 & h_dbg < 0.16) & ...
                 (s_dbg > 0.10 & s_dbg < 0.80) & ...
                 (v_dbg > 0.16 & v_dbg < 0.99);
    skinYCbCrdbg = (Cb_dbg >= 77/255 & Cb_dbg <= 127/255) & ...
                   (Cr_dbg >= 133/255 & Cr_dbg <= 173/255);
    skinMaskCombineddbg = skinHSVdbg & skinYCbCrdbg;
    greenMaskdbg = (h_dbg > 0.18 & h_dbg < 0.46) & ...
                   (s_dbg > 0.15) & ...
                   (v_dbg > 0.08);
    skinMaskDebugFinal = skinMaskCombineddbg & ~greenMaskdbg;
end
try
    handMask = get_hand_mask_simple(filePath);
catch ME
    fprintf('%s%s segmentation failed: %s\n', name, ext, ME.message);
    continue;
end
if showDebugHists
    figure('Name', ['HSV Visuals: ' name], 'NumberTitle', 'off');
    subplot(2,2,1);
    imshow(I);
    title(['Original RGB: ' name]);
    subplot(2,2,2);
    imshow(h_dbg, []);
    title('Hue');
    subplot(2,2,3);
    imshow(s_dbg, []);
    title('Saturation');
    subplot(2,2,4);
    imshow(v_dbg, []);
    title('Value');
    figure('Name', ['HSV Histograms + Masks: ' name], 'NumberTitle', 'off');
    subplot(2,3,1);
    histogram(h_dbg(:), 40);
    title('Hue Histogram');
    xlim([0 1]);
    subplot(2,3,2);
    histogram(s_dbg(:), 40);
    title('Saturation Histogram');
    xlim([0 1]);
    subplot(2,3,3);
    histogram(v_dbg(:), 40);
    title('Value Histogram');
    xlim([0 1]);
    subplot(2,3,4);
    imshow(skinHSVdbg);
```

```
    title('HSV Skin Mask');
    subplot(2,3,5);
    imshow(skinMaskCombineddbg);
    title('HSV + YCbCr Mask');
    subplot(2,3,6);
    imshow(handMask);
    title('Final Cropped Hand Mask');
    figure('Name', ['Chroma Histograms + Masks: ' name], 'NumberTitle', 'off');
    subplot(2,3,1);
    histogram(Cb_dbg(:), 40);
    title('Cb Histogram');
    xlim([0 1]);
    subplot(2,3,2);
    histogram(Cr_dbg(:), 40);
    title('Cr Histogram');
    xlim([0 1]);
    subplot(2,3,3);
    scatter(Cb_dbg(:), Cr_dbg(:), 4, '.');
    xlim([0 1]);
    ylim([0 1]);
    xlabel('Cb');
    ylabel('Cr');
    title('Cb-Cr Chroma Scatter');
    subplot(2,3,4);
    imshow(skinYCbCrdbg);
    title('YCbCr Skin Mask');
    subplot(2,3,5);
    imshow(skinMaskDebugFinal);
    title('Final Skin Mask Before Cleanup');
    subplot(2,3,6);
    imshow(handMask);
    title('Final Cropped Hand Mask');
end
handMask = normalize_mask(handMask, normSize);
try
    [boundaryXY, cx, cy, rSig, rSmooth, peakIdx, peakVals] = ...
        get_radial_signature(handMask);
catch ME
    fprintf('%s%s boundary extraction failed: %s\n', name, ext, ME.message);
    continue;
end
[solidityVal, extentVal, eccentricityVal, areaVal, aspectVal, compactVal] = ...
    extract_shape_cues(handMask);
numPeaks = length(peakIdx);
radialRange = max(rSmooth) - min(rSmooth);
radialMean  = mean(rSmooth);
radialStd   = std(rSmooth);
  % IMPROVED CLASSIFICATION
% Strong rock:
```

```
%with no extended fingers there are 0 radial peaks
%rock has solidity close to 1 (convex) solidity= A/convexhullarea
%rock has small compacteness com=(perim)^2/A
%shape-based hand gesture classification using geometric descriptors is
%supported by [10], while compactness / solidity style region metrics as
%useful discriminative shape features are also discussed in [11]
isRock = (numPeaks <= 1) && (solidityVal > 0.78) && (compactVal < 3.1);
% Strong scissors:
%has 2-3 rad peaks, high eccentricity(highly elongated) and low extent
%meaning does not fill much of the bounding box
%eccentricity, extent, and other region-based geometric descriptors are
%consistent with the types of hand-shape features used in [10], [11]
isScissors = ...
    (numPeaks >= 2 && numPeaks <= 3) &&(eccentricityVal > 0.88) && ...
    (extentVal < 0.42) && (solidityVal < 0.82);
% abnormal scissors:
% when the two fiungers mearge in mask
%here only one peak, strong concavity, still low extent (unlike rock),
%and still elongated like typical scissors
%the use of multiple geometric constraints to separate similar hand
%postures is in line with the shape-feature approaches in [10], [11]
isScissorsMerged = ...
    (numPeaks == 1) && ...
    (solidityVal < 0.70) && ...
    (extentVal > 0.24) && ...
    (extentVal < 0.36) && ...
    (eccentricityVal > 0.88) && ...
    (compactVal > 6.0) && ...
    (aspectVal > 0.90) && ...
    (aspectVal < 1.65);
% Strong paper:
%when not already classified as rock or scissors check for more tahn 3
%peaks (fingers), medium compactness, medium solidity, and large radial
%standard deviation
%combining radial-signature evidence with global shape metrics follows
%this geometric hand classification is used in [8]
% with region-shape descriptors used in [10]
isPaper = ...
    (~isRock) && (~isScissors) && (~isScissorsMerged) &&...
    ((numPeaks >= 3) ||(compactVal > 3.2 && solidityVal < 0.86) ||...
     (radialStd > 26 && compactVal > 3.0));
if isRock
    label = 'Rock';
elseif isScissors || isScissorsMerged
    label = 'Scissors';
elseif isPaper
    label = 'Paper';
else
    %backup rules
```

```
    if numPeaks <= 1
        if compactVal < 2.8 && solidityVal > 0.80
            label = 'Rock';
            %if 1 or less finger but other metrics indicate not a rock,
            %check other metrics to classify as scissors or paper in
            %the case of merged fingers
        elseif (solidityVal < 0.70) && ...
                (extentVal > 0.24) && (extentVal < 0.36) && ...
                (eccentricityVal > 0.88) && ...
                (compactVal > 6.0) && ...
                (aspectVal > 0.90) && (aspectVal < 1.65)
            label = 'Scissors';
        else
            label = 'Paper';
        end
    %two finger peaks is most often scissors
    elseif numPeaks == 2
        if eccentricityVal > 0.88 && extentVal < 0.45
            label = 'Scissors';
        elseif compactVal > 3.3
            label = 'Paper';
        else
            label = 'Rock';
        end
    %more than 3 finger peaks is most often paper
    elseif numPeaks >= 3
        if eccentricityVal > 0.90 && extentVal < 0.40 && solidityVal < 0.80
            label = 'Scissors';
        else
            label = 'Paper';
        end
    else
        label = 'Paper';
    end
end
%debug prints
fprintf('%s%s -> %s\n', name, ext, label);
fprintf('   Radial Peaks=%d  Mean=%.2f  Std=%.2f  Range=%.2f\n', ...
    numPeaks, radialMean, radialStd, radialRange);
fprintf('   Shape: Sol=%.3f  Ext=%.3f  Ecc=%.3f  Area=%d  Aspect=%.3f  Compact=%.3f\n', ...
    solidityVal, extentVal, eccentricityVal, areaVal, aspectVal, compactVal);
%print plots
figure('Name', [name ext]);
subplot(2,2,1);
imshow(I);
title(['Original: ' name]);

subplot(2,2,2);
imshow(handMask);
```

```
  title(['Normalized Mask: ' label]);

  subplot(2,2,3);
  imshow(handMask); hold on;
  plot(boundaryXY(:,1), boundaryXY(:,2), 'r', 'LineWidth', 1.5);
  plot(cx, cy, 'bo', 'MarkerSize', 8, 'LineWidth', 2);
  hold off;
  title('Boundary + Centroid');

  subplot(2,2,4);
  plot(rSig, 'b'); hold on;
  plot(rSmooth, 'r', 'LineWidth', 2);
  if ~isempty(peakIdx)
     plot(peakIdx, peakVals, 'ko', 'MarkerSize', 7, 'LineWidth', 2);
  end
  hold off;
  grid on;
  title(sprintf('Radial Signature (%d peaks)', numPeaks));
  xlabel('Boundary Point Index');
  ylabel('Distance from Centroid');

end
fprintf('-----------------------------------------------------------------------------------------------------------------------\n');
% summary display window with original images only
figure('Name','All Original Images','NumberTitle','off', ...
    'Units','normalized','Position',[0.03 0.07 0.94 0.84]);
for k = 1:numFiles
  subplot(3,4,k);
  imshow(resultImgs{k});
  title(resultNames{k}, 'Interpreter', 'none');
end
sgtitle('Original Test Images','FontSize',16,'FontWeight','bold');
% hand segmentation (these script concepts are drawn from the strawberry
% segmentation in project 5)
function handmsk = get_hand_mask_simple(imgPath)
  I = imread(imgPath);
%check if rgb image
  if size(I,3) ~= 3
     error('Image is not RGB.');
  end
  I = im2double(I);
  HSV = rgb2hsv(I); % Mathworks formating used here [6]
  h = HSV(:,:,1);
  s = HSV(:,:,2);
  v = HSV(:,:,3);
  YCbCr = rgb2ycbcr(I); %  Mathworks formating used here [7]
  cb = YCbCr(:,:,2);
  cr = YCbCr(:,:,3);
  %define skin tone range for hue saturationa and value
```

```
%threshold-style HSV / YCbCr skin detection follows the common
%color-space approach discussed in [9]
HSV_skin = (h > 0.00 & h < 0.16) & ...
        (s > 0.10 & s < 0.80) & ...
        (v > 0.16 & v < 0.99);
%define chroma range
skinYcbcr = (cb >= 77/255 & cb <= 127/255) & ...
         (cr >= 133/255 & cr <= 173/255);
%each pixel must be in both ranges to be classified
skinmsk = HSV_skin & skinYcbcr;
%identifying green background pixels
greenmsk = (h > 0.18 & h < 0.46) & (s > 0.15) & (v > 0.08);
skinmsk = skinmsk & ~greenmsk;
%calling cleanup funcs.
%reduce salt and peper noise
skinmsk = medfilt2(skinmsk, [3 3]);
%remove little blobs
skinmsk = imopen(skinmsk, strel('disk', 2));
%fill gaps
skinmsk = imclose(skinmsk, strel('disk', 3));
%fill holes in the hand
skinmsk = imfill(skinmsk, 'holes');
%%remove conctions
skinmsk = bwareaopen(skinmsk, 250);
%the largest blob is hand blob
CC = bwconncomp(skinmsk);
if CC.NumObjects == 0
   error('No hand region found.');
end
%count number of connect hand pixels
numpix = cellfun(@numel, CC.PixelIdxList);
%find which connected blob is the largest
[~, idxLargest] = max(numpix);
 %mask should only have the hand in it
%createa  blank bin image
handmsk = false(size(skinmsk));
%make largest blob pixels be 1 to create hand mask
handmsk(CC.PixelIdxList{idxLargest}) = true;
%fill in any holes
handmsk = imfill(handmsk, 'holes');
%remove noise
handmsk = imopen(handmsk, strel('disk', 1));
%more gap filling
handmsk = imclose(handmsk, strel('disk', 2));
handmsk = bwareaopen(handmsk, 150);
stats = regionprops(handmsk, 'BoundingBox'); %region props function and
% formating from mathworks [2]
if isempty(stats)
   %code stops if no mask
```

```
        error('Bound box not found.');
    end
    %hand rect., crop mask to hand, and ensure bin mask
    bbox = stats(1).BoundingBox;
    handmsk = imcrop(handmsk, bbox);
    handmsk = logical(handmsk);
end
%Make all of the masks the same size so that results are compared fairly
function outmsk = normalize_mask(inMask, outSize)
    inMask = logical(inMask);
    stats = regionprops(inMask, 'BoundingBox'); % [2]
    if isempty(stats)
        outmsk = false(outSize);
        return;
    end
    bbox = stats(1).BoundingBox;
    cropMask = imcrop(inMask, bbox);
    cropMask = logical(cropMask);
    [h, w] = size(cropMask);
    scale = min(outSize(1)/h, outSize(2)/w);
    newH = max(1, round(h * scale));
    newW = max(1, round(w * scale));
    resized = imresize(cropMask, [newH newW], 'nearest');
    outmsk = false(outSize);
    rStart = floor((outSize(1)-newH)/2) + 1;
    cStart = floor((outSize(2)-newW)/2) + 1;
    outmsk(rStart:rStart+newH-1, cStart:cStart+newW-1) = resized;
    outmsk = imfill(outmsk, 'holes');
    outmsk = bwareaopen(outmsk, 80);
end
%radial signature
function [boundaryXY, cx, cy, rSig, rSmooth, peakIdx, peakVals] = get_radial_signature(mask)
    %Find boundary of mask and ignore holes
    %boundary tracing and centroid-to-boundary radial signature follow
    %the geometric shape-signature idea is discussed in [8]
    % we then implement it with the use of [1]–[5]
    B = bwboundaries(mask, 'noholes'); % [1]
    if isempty(B)
        error('No boundary found.');
    end
    %choose largest boundary which should be the main hand
    maxLen = 0;
    bestIdx = 1;
    for i = 1:length(B)
        if size(B{i},1) > maxLen
            maxLen = size(B{i},1);
            bestIdx = i;
        end
    end
```

```
  %convert row column format of the boundaries into coordinates for ease
  %of understanding
  boundaryRC = B{bestIdx};
  boundaryXY = [boundaryRC(:,2), boundaryRC(:,1)];
  %make sure the boundary is big enough to be a hand
  nPts = size(boundaryXY,1);
  if nPts < 40
     error('Boundary too short.');
  end
  %make each boundary and equal 400 points
  oldt = linspace(0,1,nPts);
  newt = linspace(0,1,400);
  res_x = interp1(oldt, boundaryXY(:,1), newt, 'linear'); % [4]
  res_y = interp1(oldt, boundaryXY(:,2), newt, 'linear'); % [4]
  boundaryXY = [res_x(:), res_y(:)];
  %find the centroid of each hand boundary
  stats = regionprops(mask, 'Centroid'); % [2]
  if isempty(stats)
     error('Centroid not found.');
  end
  cx = stats(1).Centroid(1);
  cy = stats(1).Centroid(2);
  %commpute a vector of disctances between the center and the boundary
  dx = boundaryXY(:,1) - cx;
  dy = boundaryXY(:,2) - cy;
  rSig = sqrt(dx.^2 + dy.^2);
  %smooth radial center to clear out spikes
  rSmooth = movmean(rSig, 9); % [5]
  rSmooth = movmean(rSmooth, 9); % [5]
  %define radial peaks within the vector
  prom = max(4, 0.10 * (max(rSmooth) - min(rSmooth)));
  minDist = 22;
  %find the peaks which could be fingers
  [peakVals, peakIdx] = findpeaks(rSmooth, ... % [3]
     'MinPeakProminence', prom, ...
     'MinPeakDistance', minDist);
  %take out bad peaks
  if ~isempty(peakIdx)
     keep = peakVals > (mean(rSmooth) + 0.08 * (max(rSmooth)-min(rSmooth)));
     peakIdx = peakIdx(keep);
     peakVals = peakVals(keep);
  end
end
%helper functions for feature thresholding (explained in tables)
function [solidityVal, extentVal, eccentricityVal, areaVal, aspectVal, compactVal] = extract_shape_cues(mask)
  stats = regionprops(mask,'Solidity', 'Extent', 'Eccentricity', 'Area', 'BoundingBox', 'Perimeter'); % [2]
  %if the mask is empter there are no features in it
  if isempty(stats)
```

```
    solidityVal = 0;
    extentVal = 0;
    eccentricityVal = 0;
    areaVal = 0;
    aspectVal = 0;
    compactVal = 0;
    return;
  end
  %for each detected hand get these values (equations in report tables)
  solidityVal = stats(1).Solidity;
  extentVal = stats(1).Extent;
  eccentricityVal = stats(1).Eccentricity;
  areaVal = stats(1).Area;
  %Each bounding box has the objects coordinates, height, and width
  bb = stats(1).BoundingBox;
  w = bb(3);
  h = bb(4);

  %make sure not dividing by zero
  if h == 0
    aspectVal = 0;
  else
    aspectVal = w / h;
  end
  %compute the compactness because this is not a built in matlab func.
  P = stats(1).Perimeter;
  if areaVal == 0
    compactVal = 0;
  else
    compactVal = (P^2) / (4*pi*areaVal);
  end
end
%func. to troubleshoot file issues
function filePath = pick_file(folder, base_name)
  d = dir(fullfile(folder, [base_name '.*']));
  if isempty(d)
    error('Could not find %s.* in folder:\n%s', base_name, folder);
  end
  filePath = fullfile(folder, d(1).name);
End
```

## HSV and YCbCr:

```
clc; clear; close all;
names = ["rock","paper","scissors"];
N = 2;
allHandHSV = [];
```

```
allBgHSV = [];
allHandYCbCr = [];
allBgYCbCr = [];
%% ---- Find smallest image size ----
minH = inf;
minW = inf;
for n = 1:length(names)
  for i = 1:N
     fname = names(n) + i + ".JPG";
     I = imread(fname);
     [h,w,~] = size(I);
     minH = min(minH,h);
     minW = min(minW,w);
  end
end
targetSize = [minH minW];
%% ---- Collect hand and background pixels ----
for n = 1:length(names)
  for i = 1:N
     fname = names(n) + i + ".JPG";
     I0 = imread(fname);
     I0 = imresize(I0,targetSize);
     I = im2double(I0);
     HSV = rgb2hsv(I);
     YCbCr = rgb2ycbcr(I);
     H = HSV(:,:,1);
     S = HSV(:,:,2);
     V = HSV(:,:,3);
     Cb = YCbCr(:,:,2);
     Cr = YCbCr(:,:,3);
     %% ---- Draw hand polygon ----
     figure
     imshow(I0)
     title("Draw polygon around HAND for " + fname)
     h = drawpolygon;
     handMask = createMask(h);
     bgMask = ~handMask;
     %% ---- Show clipped hand ----
     handOnly = I0;
     handOnly(repmat(~handMask,[1 1 3])) = 0;
     figure
     subplot(1,2,1)
     imshow(I0)
     title("Original - " + fname)
     subplot(1,2,2)
     imshow(handOnly)
     title("Clipped Hand - " + fname)
     %% ---- Store pixels ----
     allHandHSV = [allHandHSV; H(handMask), S(handMask), V(handMask)];
```

```
    allBgHSV   = [allBgHSV;   H(bgMask),   S(bgMask),   V(bgMask)];
    allHandYCbCr = [allHandYCbCr; Cb(handMask), Cr(handMask)];
    allBgYCbCr   = [allBgYCbCr;   Cb(bgMask),   Cr(bgMask)];
  end
end
%% ---- HSV Histograms ----
figure
subplot(1,3,1)
histogram(allHandHSV(:,1),50,'FaceAlpha',0.5)
hold on
histogram(allBgHSV(:,1),50,'FaceAlpha',0.5)
title("Hue")
legend("Hand","Background")
xlim([0 1])
subplot(1,3,2)
histogram(allHandHSV(:,2),50,'FaceAlpha',0.5)
hold on
histogram(allBgHSV(:,2),50,'FaceAlpha',0.5)
title("Saturation")
legend("Hand","Background")
xlim([0 1])
subplot(1,3,3)
histogram(allHandHSV(:,3),50,'FaceAlpha',0.5)
hold on
histogram(allBgHSV(:,3),50,'FaceAlpha',0.5)
title("Value")
legend("Hand","Background")
xlim([0 1])
%% ---- YCbCr Histograms ----
figure
subplot(1,2,1)
histogram(allHandYCbCr(:,1),50,'FaceAlpha',0.5)
hold on
histogram(allBgYCbCr(:,1),50,'FaceAlpha',0.5)
title("Cb")
legend("Hand","Background")
xlim([0 1])
subplot(1,2,2)
histogram(allHandYCbCr(:,2),50,'FaceAlpha',0.5)
hold on
histogram(allBgYCbCr(:,2),50,'FaceAlpha',0.5)
title("Cr")
legend("Hand","Background")
xlim([0 1])
%% ---- H-S Scatter ----
figure
scatter(allHandHSV(:,1), allHandHSV(:,2), 5, 'r', 'filled')
hold on
scatter(allBgHSV(:,1), allBgHSV(:,2), 5, 'b', 'filled')
```

```
xlabel("Hue")
ylabel("Saturation")
title("Hand vs Background in H-S")
legend("Hand","Background")
grid on
xlim([0 1])
ylim([0 1])
%% ---- Cb-Cr Scatter ----
figure
scatter(allHandYCbCr(:,1), allHandYCbCr(:,2), 5, 'r', 'filled')
hold on
scatter(allBgYCbCr(:,1), allBgYCbCr(:,2), 5, 'b', 'filled')
xlabel("Cb")
ylabel("Cr")
title("Hand vs Background in Cb-Cr")
legend("Hand","Background")
grid on
xlim([0 1])
ylim([0 1])
```

## Masking:

```
clc; clear; close all;
%% ---------------- LOAD IMAGE ----------------
I0 = imread("rock1.jpg");
I = im2double(I0);
[rows, cols, ~] = size(I);
minBlobFinal = 40000;
%% ---------------- K-MEANS HAND MASK ----------------
X = reshape(I, [], 3);
[idx, C] = kmeans(X, 4, 'Replicates', 3);
seg = reshape(idx, rows, cols);
bestCluster = 1;
bestScore = -inf;
for k = 1:4
  R = C(k,1);
  G = C(k,2);
  B = C(k,3);
  score = (R - B) + 0.5*(R - G);
  if score > bestScore
     bestScore = score;
     bestCluster = k;
  end
end
maskKmeans = seg == bestCluster;
maskKmeans = imopen(maskKmeans, strel('disk',3));
```

```
maskKmeans = imclose(maskKmeans, strel('disk',7));
maskKmeans = imfill(maskKmeans,'holes');
%% ---------------- YCbCr + Y SKIN MASK ----------------
YCbCr = rgb2ycbcr(I);
Y  = YCbCr(:,:,1);
Cb = YCbCr(:,:,2);
Cr = YCbCr(:,:,3);
maskYCbCr = (Cb >= 88/255 & Cb <= 118/255) & ...
        (Cr >= 140/255 & Cr <= 166/255) & ...
        (Y  >= 0.25    & Y  <= 0.85);
maskYCbCr = imopen(maskYCbCr, strel('disk',3));
maskYCbCr = imclose(maskYCbCr, strel('disk',11));
maskYCbCr = imfill(maskYCbCr,'holes');
%% ---------------- HSV SKIN MASK (VISUAL ONLY) ----------------
HSV = rgb2hsv(I);
h = HSV(:,:,1);
s = HSV(:,:,2);
v = HSV(:,:,3);
HSV_skin = (h > 0.00 & h < 0.16) & ...
        (s > 0.10 & s < 0.80) & ...
        (v > 0.16 & v < 0.99);
HSV_skin = imopen(HSV_skin, strel('disk',3));
HSV_skin = imclose(HSV_skin, strel('disk',7));
HSV_skin = imfill(HSV_skin,'holes');
%% ---------------- BUILD COMBO MASK ----------------
CC = bwconncomp(maskKmeans);
maskCombo = false(rows, cols);
for k = 1:CC.NumObjects
  blobPixels = CC.PixelIdxList{k};
  if any(maskYCbCr(blobPixels))
     maskCombo(blobPixels) = true;
  end
end
%% ---------------- CLEAN BOTH CANDIDATES ----------------
% Candidate 1: YCbCr only
candYCbCr = maskYCbCr;
candYCbCr = imclose(candYCbCr, strel('disk',8));
candYCbCr = imfill(candYCbCr,'holes');
candYCbCr = bwareafilt(candYCbCr, 1);
% Candidate 2: K-means + YCbCr helper
candCombo = maskCombo;
candCombo = imopen(candCombo, strel('disk',14));
candCombo = imclose(candCombo, strel('disk',12));
candCombo = imfill(candCombo,'holes');
candCombo = bwareafilt(candCombo, 1);
%% ---------------- OVERLAP CHECK ----------------
overlapPixels = sum(candYCbCr(:) & candCombo(:));
ycbcrPixels = sum(candYCbCr(:));
```

```
if ycbcrPixels > 0
  overlapRatio = overlapPixels / ycbcrPixels;
else
  overlapRatio = 0;
end
fprintf('\nOverlap Ratio (YCbCr vs Combo) = %.3f\n', overlapRatio);
%% ---------- CHOOSE METHOD ----------
if overlapRatio >= 0.8
  finalMask = candYCbCr;
  chosenMethod = 'YCbCr only (high agreement)';
else
  finalMask = candCombo;
  chosenMethod = 'K-means + YCbCr helper (low agreement)';
end
fprintf('Chosen method: %s\n', chosenMethod);
%% ---------- FINAL SIZE FILTER ----------
finalMask = bwareaopen(finalMask, minBlobFinal);
%% ---------- BLOB FILTERING RULE ----------
CCfinal = bwconncomp(finalMask);
stats = regionprops(CCfinal, 'BoundingBox','Area','PixelIdxList');
minAspect = 0.65;
maxAreaFrac = 0.30;
imageArea = rows * cols;
if CCfinal.NumObjects > 1
  candidateAreas = [];
  candidateIdx = [];
  for k = 1:length(stats)
    bbox = stats(k).BoundingBox;
    width = bbox(3);
    height = bbox(4);
    aspectRatio = width / height;
    areaVal = stats(k).Area;
    if aspectRatio > minAspect && areaVal < maxAreaFrac * imageArea
      candidateAreas(end+1) = areaVal; %#ok<SAGROW>
      candidateIdx(end+1) = k; %#ok<SAGROW>
    end
  end
  if ~isempty(candidateIdx)
    [~, ind] = max(candidateAreas);
    bestBlob = candidateIdx(ind);
    finalMask = false(rows, cols);
    finalMask(stats(bestBlob).PixelIdxList) = true;
  end
end
%% ---------- BLOB ANALYSIS ----------
CCfinal = bwconncomp(finalMask);
stats = regionprops(CCfinal, 'BoundingBox','Area');
fprintf('\n------ Blob Analysis ------\n')
```

```
for k = 1:length(stats)
  bbox = stats(k).BoundingBox;
  width  = bbox(3);
  height = bbox(4);
  aspectRatio = width / height;
  fprintf('Blob %d\n', k)
  fprintf('Area: %.0f pixels\n', stats(k).Area)
  fprintf('Width: %.2f\n', width)
  fprintf('Height: %.2f\n', height)
  fprintf('Aspect Ratio (W/H): %.3f\n\n', aspectRatio)
end
%% ---------- DISPLAY ----------
figure
subplot(2,3,1)
imshow(I0)
title("Original")
subplot(2,3,2)
imshow(maskKmeans)
title("K-Means Mask")
subplot(2,3,3)
imshow(maskYCbCr)
title("YCbCr Mask")
subplot(2,3,5)
imshow(candYCbCr)
title("YCbCr Candidate")
subplot(2,3,6)
imshow(candCombo)
title("Combo Candidate")
figure
imshow(finalMask)
title("Final Mask - " + chosenMethod)
%% ---------- HSV VISUALIZATION ----------
figure
subplot(1,3,1)
imshow(HSV_skin)
title("HSV Mask")
subplot(1,3,2)
imshow(I0)
title("Original")
subplot(1,3,3)
overlay = I0;
overlay(repmat(~HSV_skin,[1 1 3])) = 0;
imshow(overlay)
title("HSV Applied")
```

## K-means Segmentation:

```
function [handMask] = kmeans_seg(img)
hsv_img = rgb2hsv(img);
S = hsv_img(:,:,2);
features = [S(:)];
k =3; % hand, hand shadow, and background
[idx, centroids] = kmeans(features, k);
% Reshape cluster labels
cluster_img = reshape(idx, size(S));
imgSize = size(cluster_img);
pixelLabels = reshape(idx, imgSize);
bestScore = -inf;
handMask = false(imgSize);
se = strel('disk',3);
for k = 1:3
  % Create a binary mask for the current cluster
  currentMask = (pixelLabels == k);
  currentMask = imopen(currentMask, se);
  % Get Area and Centroid
  stats = regionprops(currentMask, 'Area', 'Centroid');

  if ~isempty(stats)
    % Find the largest blob in this cluster
    [maxArea, maxIdx] = max([stats.Area]);
    centroid = stats(maxIdx).Centroid;

    % Calculate Centrality (distance from image center)
    imgCenter = [imgSize(2)/2, imgSize(1)/2];
    distFromCenter = norm(centroid - imgCenter);
    % Want a big blob that is close to the center
    score = maxArea / (distFromCenter + eps);

    if score > bestScore
      bestScore = score;
      handMask = currentMask;
    end
  end
end
% keep biggest blob
handMask = imclose(handMask, se);
handMask = bwareafilt(handMask, 1);
% Close gaps and smooth edges
se = strel('disk', 1);
handMask = imerode(handMask, se);
handMask = imfill(handMask, 'holes');
figure(1);
subplot(1,3,1)
imshow(img)
title('Original Image(Resized)')
% Display clustering
subplot(1,3,2)
imagesc(cluster_img)
axis image
title('K-Means Clusters')
```

```
colormap(jet)
subplot(1,3,3)
imshow(handMask)
title('Hand Segmentation')
end
```