Stone Deduction from Image of a Jewellery

Suraj Soni
Sai Teja Talluri

Objective

To improve the previously done stone detection and make it more reliable and accurate. (To find the net weight of stones in a jewellery consisting of gold and stones using images of jewellery).

Algorithmic implementation is done in Python(openCV) and imageJ

Drawbacks and flaws with the previous approach

- The background was always assumed to be white previously, and when segmented using white background, the white stones couldn't be detected
- It was resolved by having colored backgrounds, which is different from all the stones of a given necklace

- Previously there was a constraint on the position of the black square (Reason for the index out of bound error)
- Now the square can be placed anywhere

- The number of clusters were assumed to be 3 initially(Gold,Stone,Background).
- This didn't give a satisfactory result with jewellery having different colored stones. E.g. Number of clusters could be 4(Gold, Red stone, White stone, Background)

- Previously while doing K-Means clustering, it was assumed that the stones will form the lowest cluster, hence including only that in the final output
- But this need not be the case always, we figured that even the gold region could form the lowest cluster or even the shadow region could.

Approach

WORKFLOW OF THE CODE

1. **Detect the gold**
   Traditional HSV based segmentation
2. **Detect background, find the scale**
   Traditional HSV based segmentation, keep two copies of the image, one with a white background and one with a similar background
3. **K-Means**
   Rolling ball algorithm is applied to one with white background , and then K-means is applied to it after converting the white background to the background color
4. **Identifying the clusters**
   The clusters which comes into the background shadow range are removed
5. **Area and weight detection**
   Finally the weight of the stone is calculated from the approximate linear fit b/w area and weight

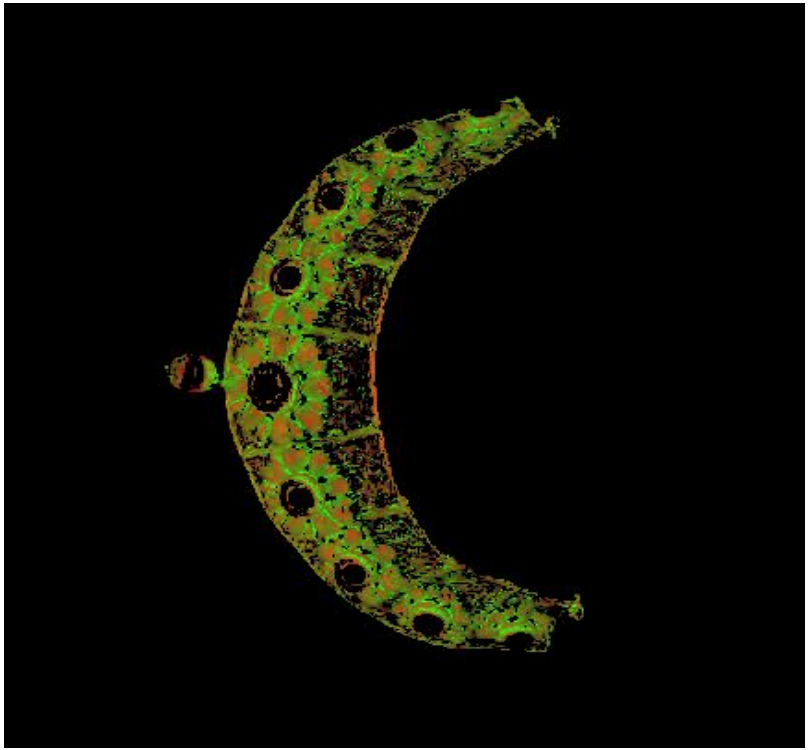
Gold Detection

Traditional segmentation using the ranges of pixel values .
Gold conveniently falls between these ranges of HSV -

[[15, 0, 0], [40, 255, 255]]

(S and V are ignored completely as evident from the above range)

INPUT → → GOLD DETECTED



Finding The Background

Traditional segmentation using the ranges of pixel values which were taken as input from the user.

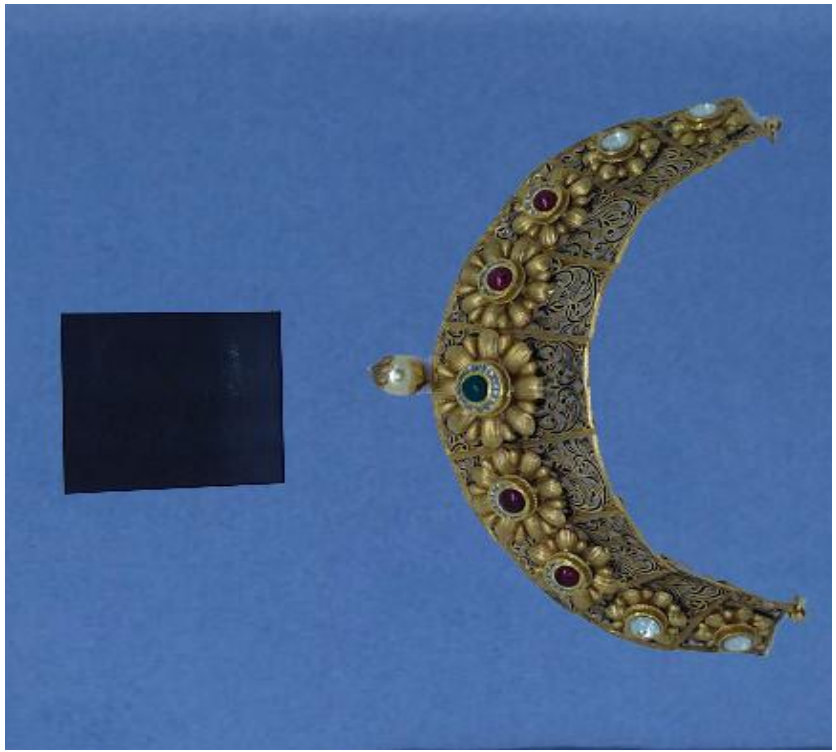
The ranges for the Background should be selected which is not so convenient and the ranges we used for some of the colors -

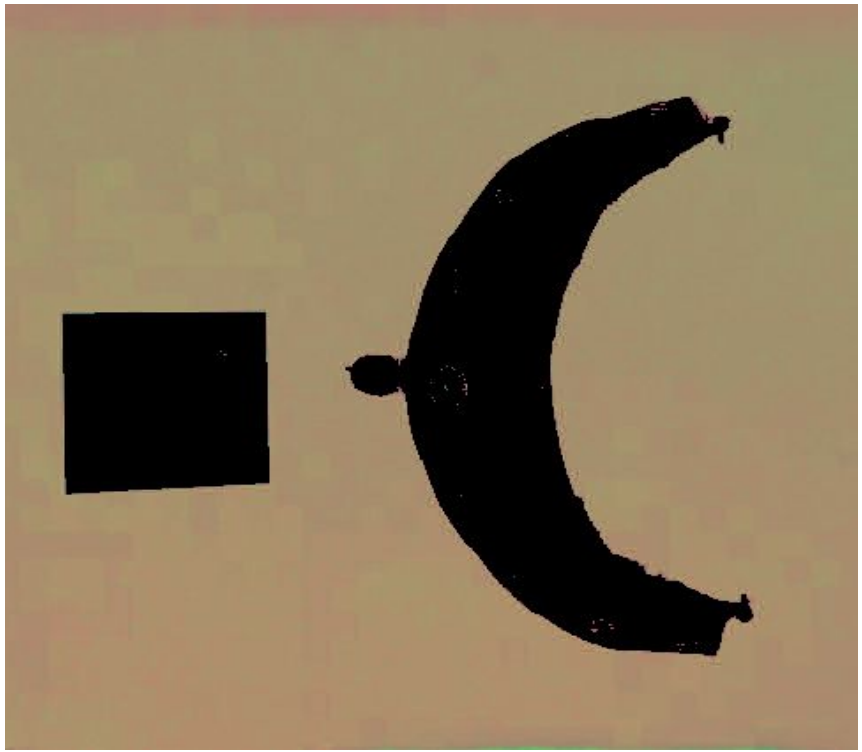
White →[0,0,165.75],[179,38.25,255]

Blue → [90,80,80],[130,255,255]

Purple → [120,70,110],[150,255,255]

INPUT →→ DETECTED BACKGROUND



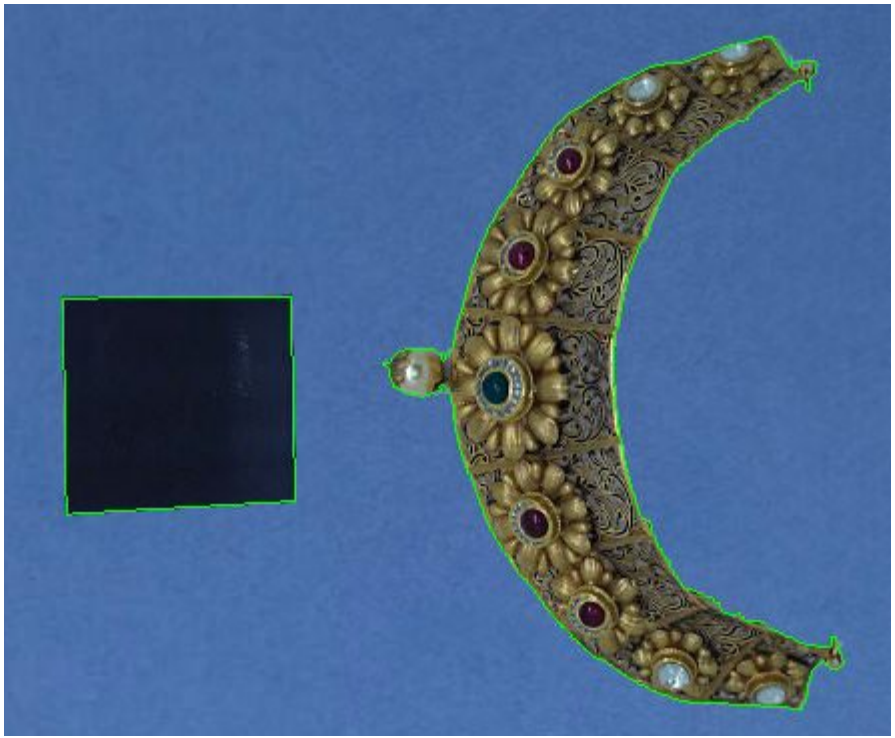


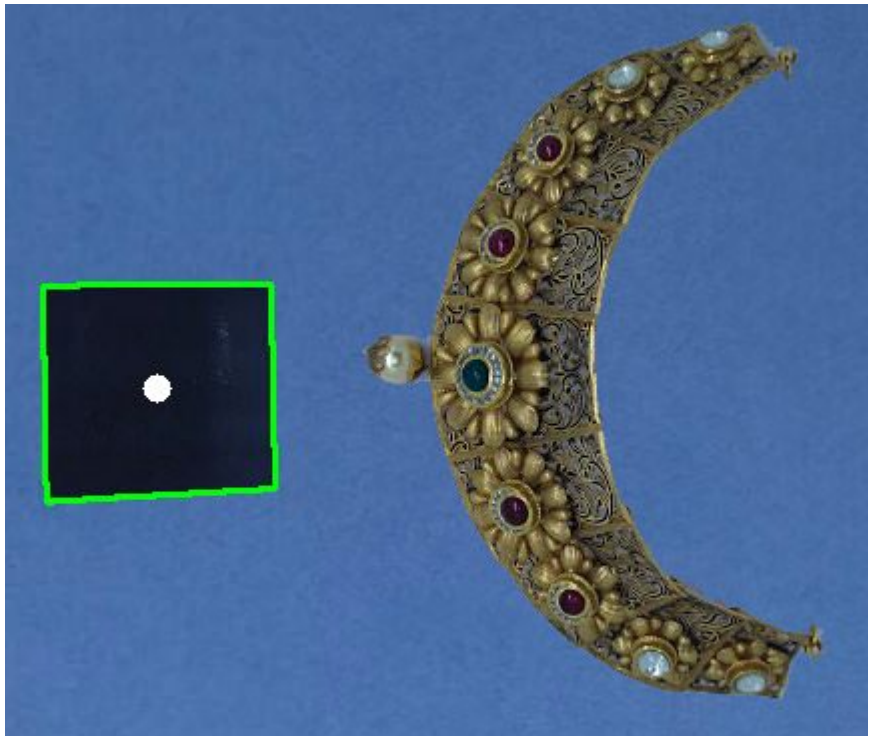
Removing The Black Square And Finding The Scale

Among all the contours, a contour which had 4 sides and whose area is greater than a defined minimum and less than a defined maximum is removed from the image. The scale is calculated from the area of this removed contour.

SCALE = AREA_SQUARE_REAL / AREA_SQUARE_IMG

ALL CONTOURS →→ SQUARE CONTOUR





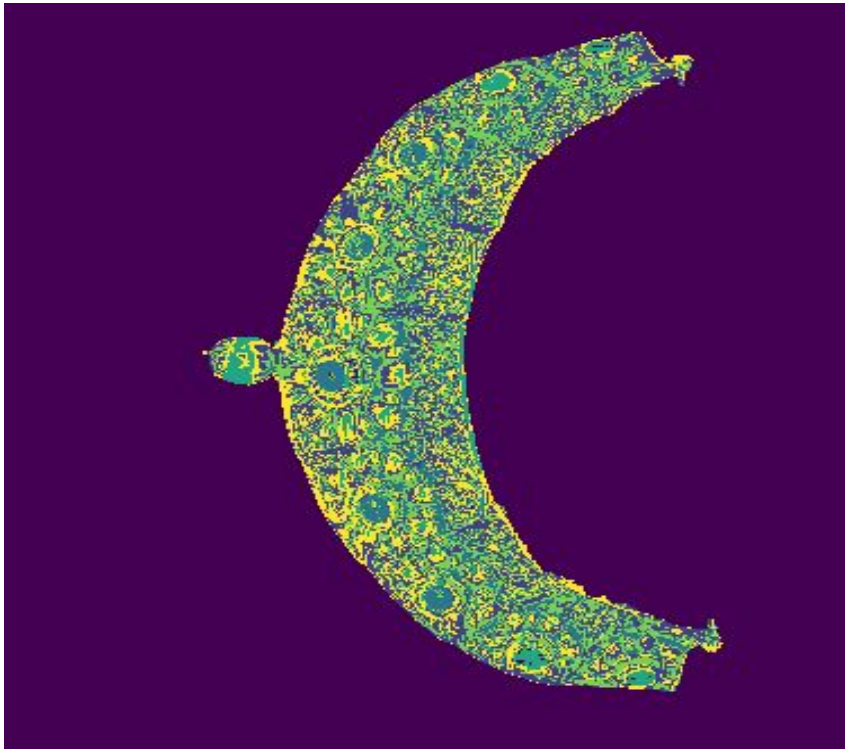
NOISE REMOVAL (K-MEAN CLUSTERING)

K-Means for gaining further information about the image, since it is unsupervised learning, had to make some assumptions about mapping back to image space.

K = 3 + number_of_stones

We found the mean squared error with the background color, and removed the last two of them. (Here the assumption is that the last two are the background and shadow respectively)

IMAGE AFTER FILLING SQUARE ⟶IMAGE AFTER CLUSTERING

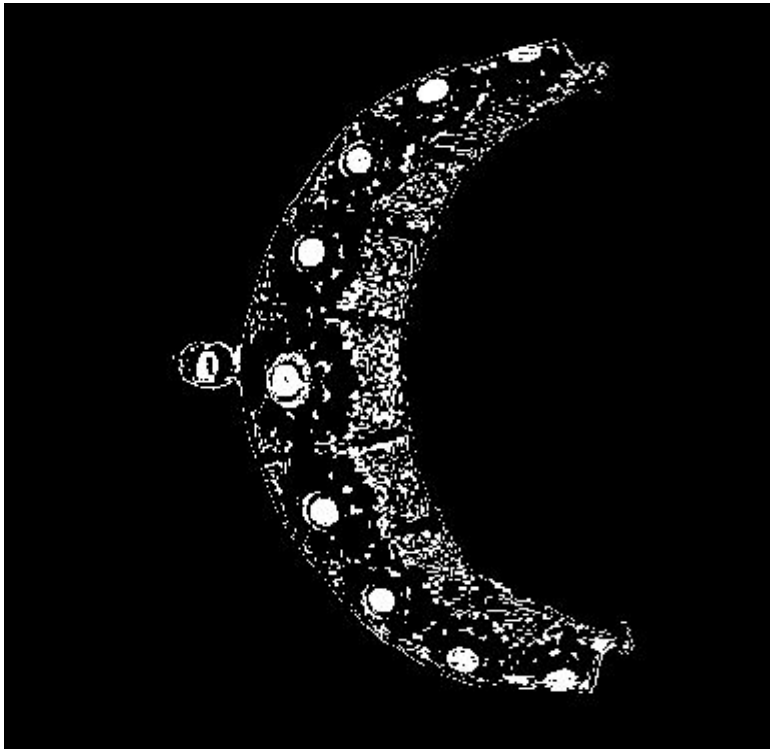


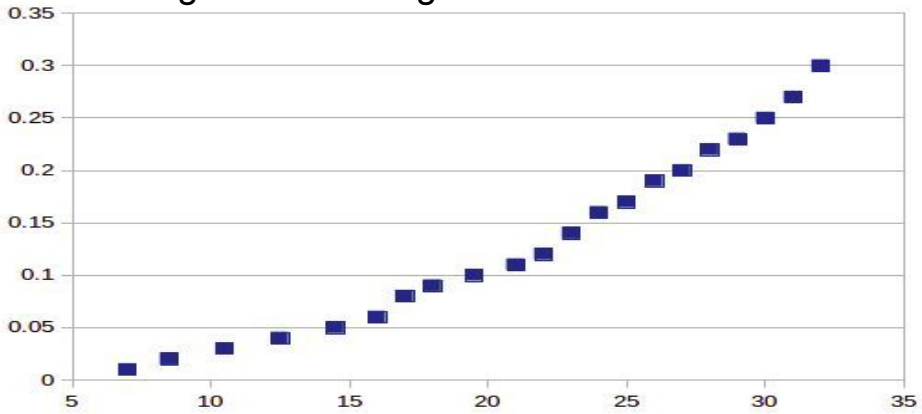
Finally Finding The Actual Area of The Stones

After all the previous steps (Gold removal, BG removal, Black Square removal), All that left are the stones.

This area is calculated by counting the number of pixels and then scaled appropriately using the scale value obtained with the help of black square.

INPUT →→ STONES DETECTED





GraphPad Software

Sign In

Scientific Software | Data Analysis Resource Center | Company | Support | How to Buy

QuickCalcs

1. Select category | 2. Choose calculator | 3. Enter data | **4. View results**

Linear Regression

Best-fit values	
Slope	0.01133 ± 0.0005774
Y-intercept	-0.1030 ± 0.01286
X-intercept	9.088
1/Slope	88.26
95% Confidence Intervals	
Slope	0.01012 to 0.01254
Y-intercept	-0.1299 to -0.07605
X-intercept	7.440 to 10.46
Goodness of Fit	
R square	0.9530
Sy.x	0.01946
Is slope significantly non-zero?	
F	385.0
DFn,DFd	1,19
P Value	< 0.0001
Deviation from horizontal?	Significant
Data	
Number of XY pairs	21
Equation	Y = 0.01133*X - 0.1030

Area To Weight

We know the relationship between Gauge no. and weight and that between Gauge no. and area and it was finally approximated that the relationship between weight and area is also linear.

This mapping b/w area and weight can be improved if we have lot of labelled data at hand.

Multiple Background Detection

Background range: `[120,70,110],[150,255,255]`

Background range: `[0,0,165.75],[179,38.25,255]`

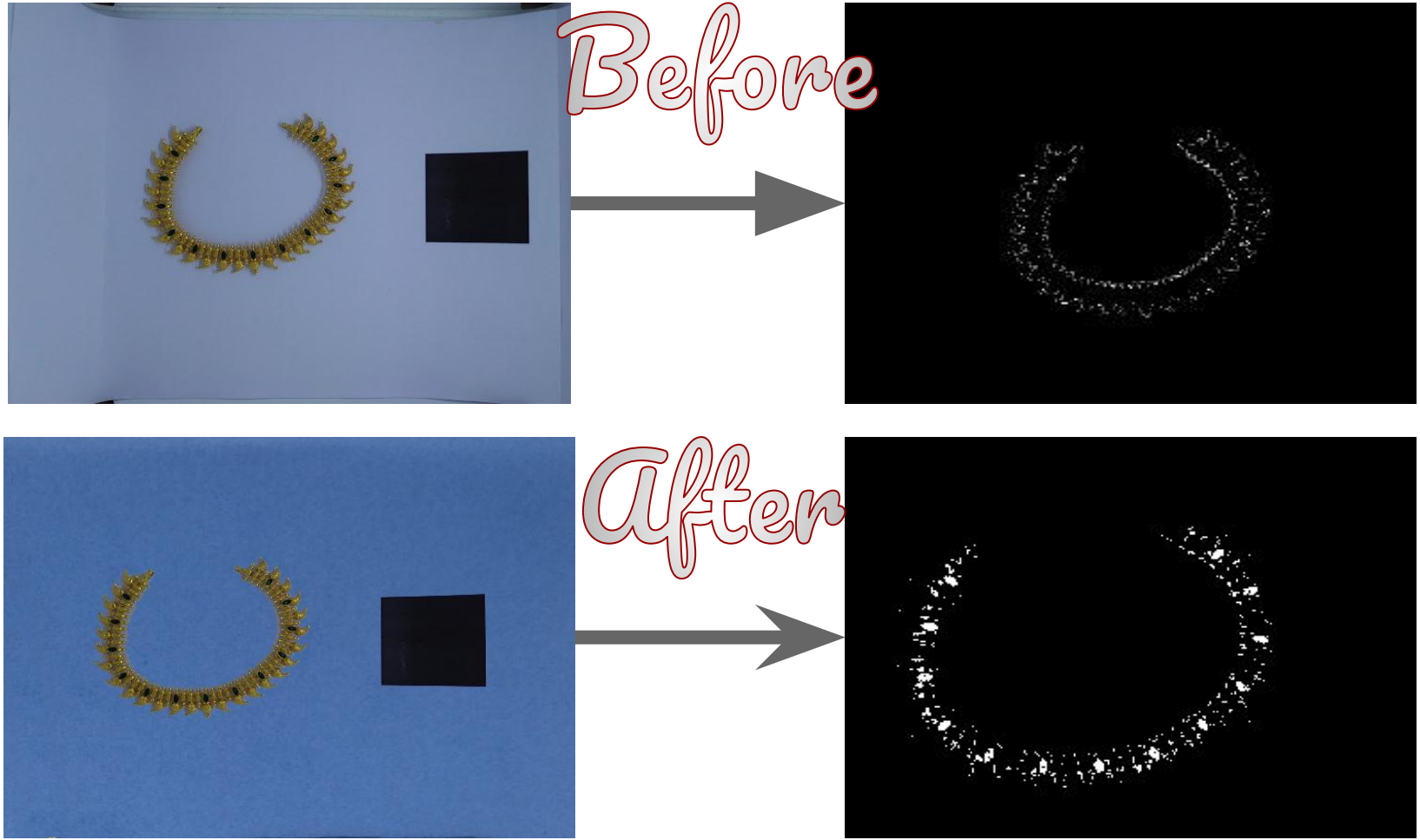
Possible Improvements And Drawbacks

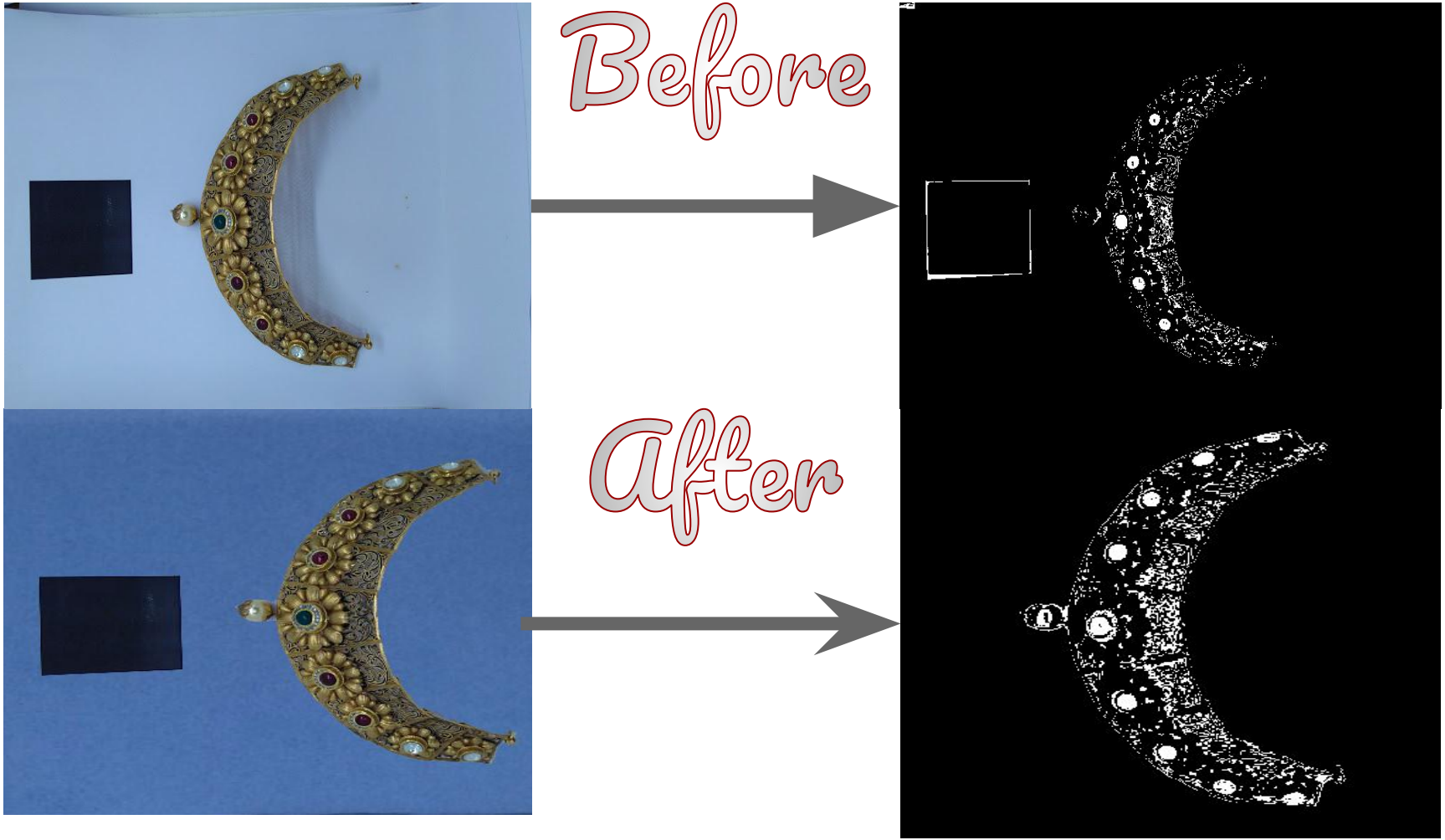
- The assumption about K-Means that the smallest mean squared error ones constitute of the background(seems legit) and the shadows (may fail in rare cases) need to be improved. We haven’t figured this out :(
- There is a lot of noise with stones having white(light color) as they reflect gold color as it is. This reduces the final weight of the stones. Can be improved by using different colored led.
- Doesn’t work with ornaments which are not flat (even necklaces) as the stones appear tilted, and reducing a significant area.

Deployment

- The functionality previously mentioned is exposed through an API
- The information to be passed should include the url of the image, the lower and upper bound of the background and the number of different stones (color).

Finally,
What did we Achieve ?


Before
After


Before
After

Thank You