Code Explanation

Get background information

```
[m, n, c] = image.shape
```

```
if(color_space == “HSV"):
    image = cv2.cvtColor(image, cv2.COLOR_RGB2HSV)#change image 
color space to HSV
```

```
image, trimap = image / 255.0, trimap / 255.0
```

```
foreground = (trimap == 1.0).astype(int)
```

```
background = (trimap == 0.0).astype(int)
```

```
all_constraints = foreground + background#Combine confirmed 
foreground and background into a image
```

Prepare feature vector for difference colour space

```
print("Finding KNN")
nbrs_num  = 10
```

```
a, b = np.unravel_index(np.arange(m*n), (m,n))
```

```
if(color_space == "HSV"):#Using difference feature for each color 
space
    h,s,v =cv2.split(image)
    feature = np.append( (np.reshape(np.cos(h),
(m*n)),np.reshape(np.sin(h), (m*n)), np.reshape(s,
(m*n)),np.reshape(v, (m*n))), [a, b] /np.sqrt(m*m + n*n),
axis=0).T#X(i) = (coś(h), sin(h), s, v, x, y)
```

```
    else:
        feature = np.append(np.reshape(image, (m*n, c)).T, [a,
b] /np.sqrt(m*m + n*n), axis = 0 ).T#X(i)
```

Find KNN

```
nbrs = sklearn.neighbors.NearestNeighbors(n_neighbors=nbrs_num,
n_jobs=-1).fit(feature)
```

```
knn_indices = nbrs.kneighbors(feature)[1]#Find X(j) with KNN
```

Calculate affinity matrix A

```
print("Calculating affinity matrix A")
row_index = np.repeat(np.arange(m*n), nbrs_num)#X(i)'s indices
```

```
col_index = np.reshape(knn_indices, (m*n*nbrs_num))#X(j)'s indices
```

```
kernel = 1 - np.linalg.norm(feature[row_index] -
feature[col_index], axis=1)/ (c+2)#||X(i) - X(j)|| / C
```

```
A = scipy.sparse.coo_matrix((kernel, (row_index, col_index)),
shape=(m*n, m*n))
```

Prepare matrix H

```
D = scipy.sparse.diags(np.ravel(A.sum(axis=1)))
L = D-A
```

```
D = scipy.sparse.diags(np.ravel(all_constraints))
```

```
v = np.ravel(foreground)#a binary vector of pixel indices
corresponding to user markups for a given layer
```

```
c = 2*my_lambda*np.transpose(v)
```

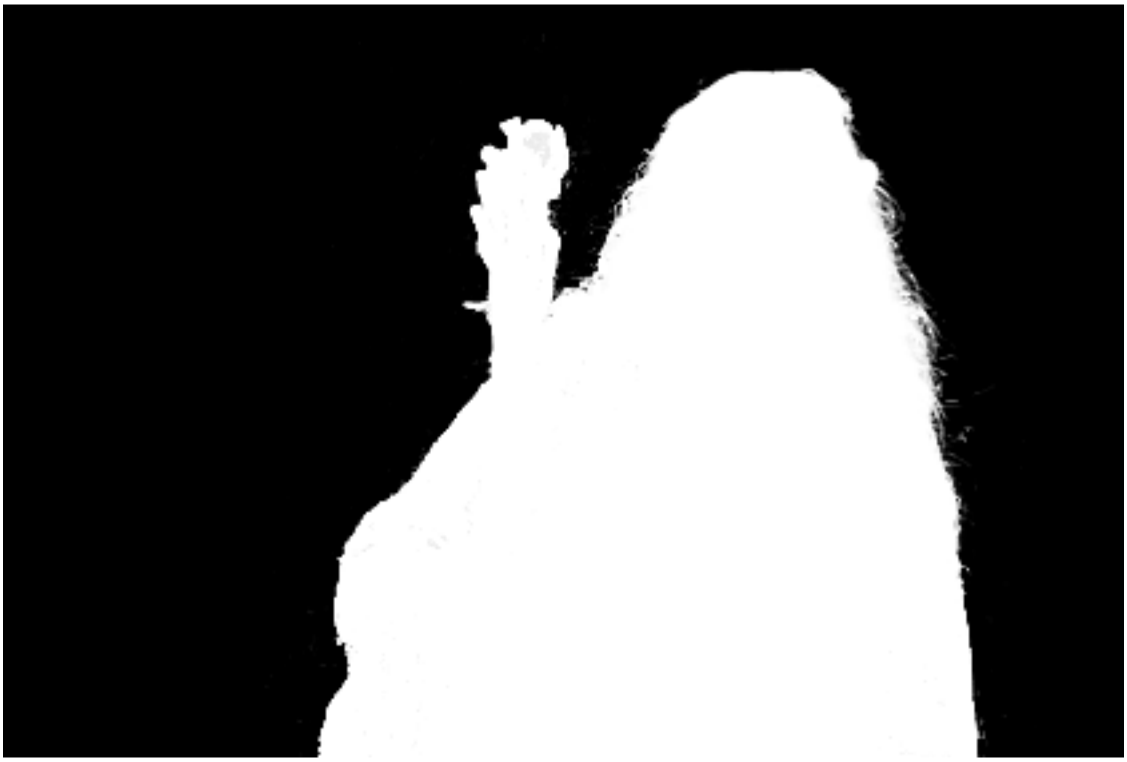
```
H = 2*(L + my_lambda*D)#H = 2* (L + lambda*D)
```

Solve linear system

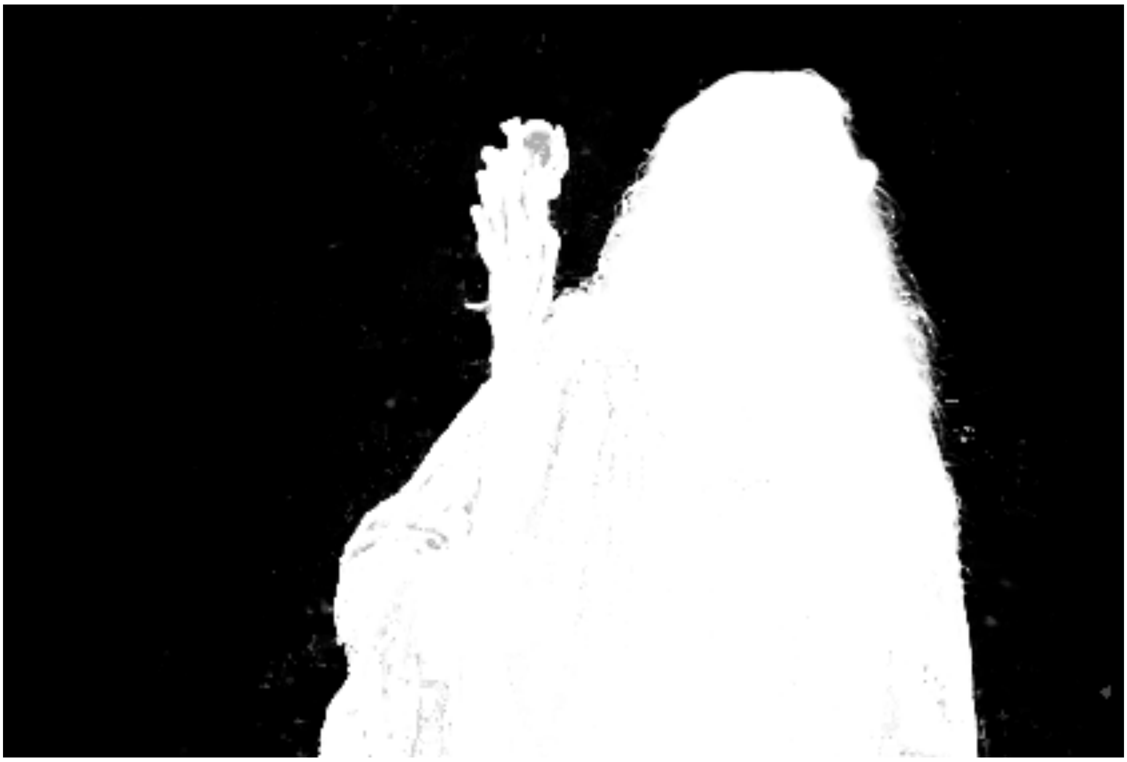
```
print('Solving linear system')
warnings.filterwarnings('error')
alpha = []
try:
    alpha = np.minimum(np.maximum(scipy.sparse.linalg.spsolve(H,
c), 0), 1).reshape(m, n)

except Warning:#Cannot find solution
    x = scipy.sparse.linalg.lsqr(H, c)
    alpha = np.minimum(np.maximum(x[0], 0), 1).reshape(m, n)
return alpha
```

Try to play with the number of K and see how it affects the result.

Smaller K means a shorter KNN search time as well as a shorter time for solving a sparser/faster linear system.
Very large K will produce undesired artifacts in the alpha result

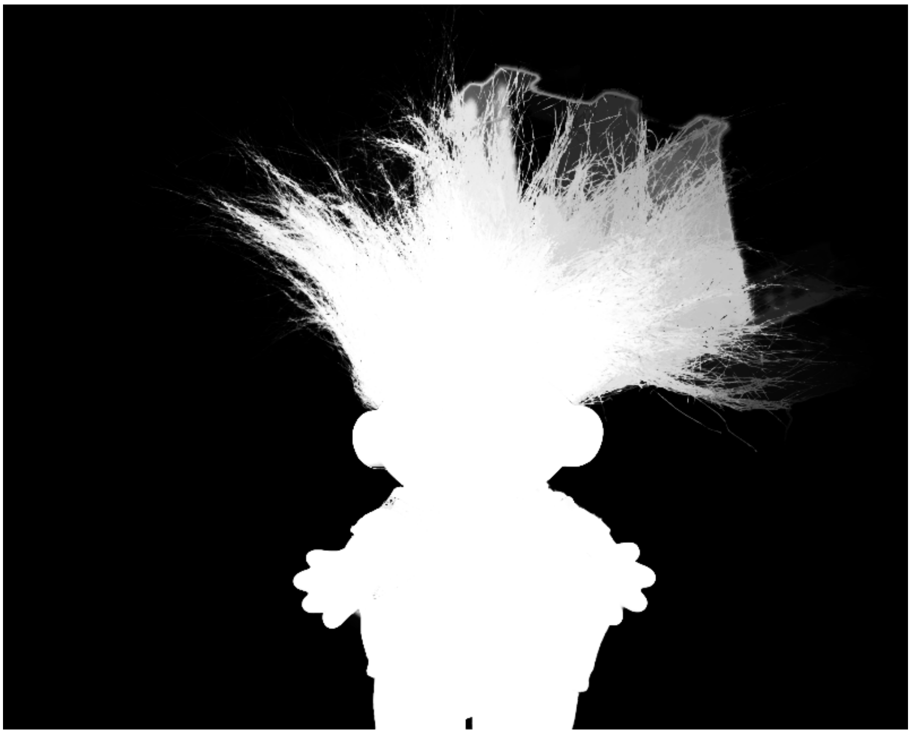


HSV K=10

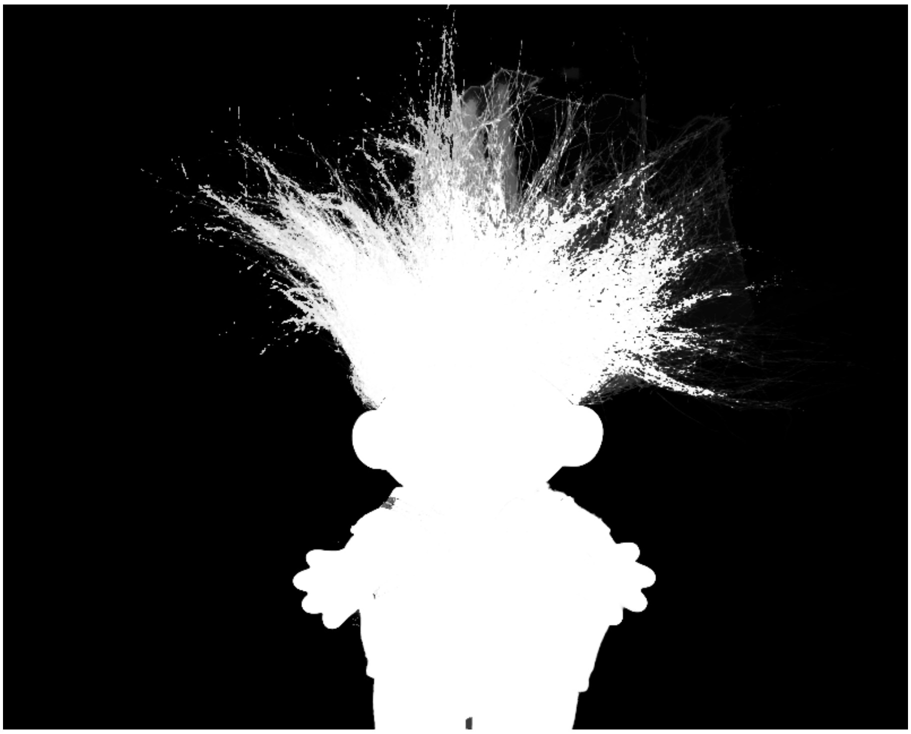


HSV K=50

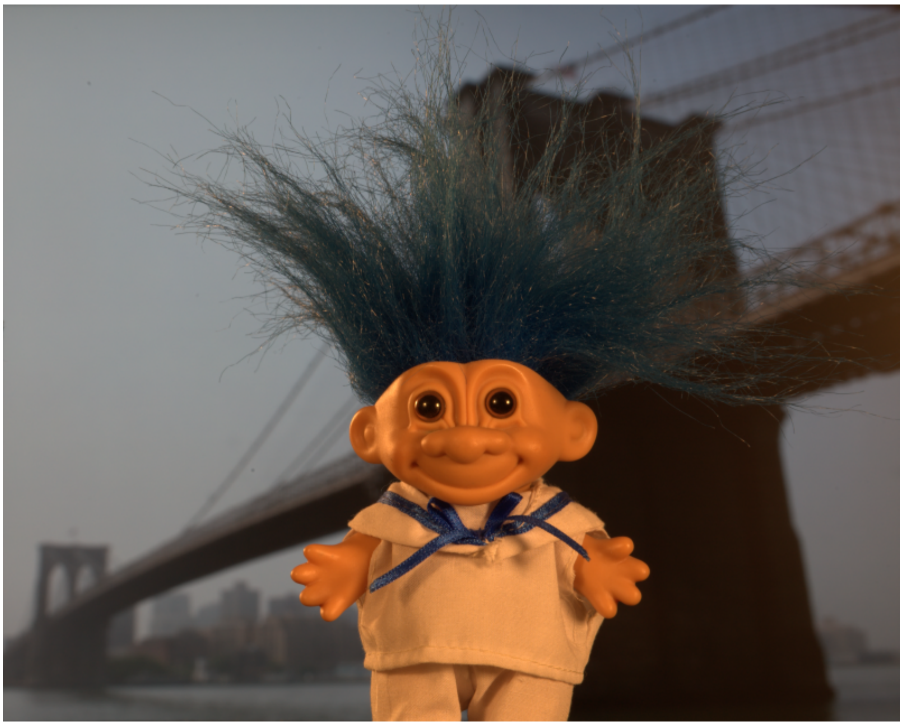
Try different ways of representing feature vectors(RGB/HSV)



RGB result



HSV result



Origin image

We can see HSV have better result than RGB because the troll’s hair colour is similar with background.