Aim: To study the Depth Estimation

Objective: To Capturing Frames form a depth camera creating a mask from a disparity map Masking a copy operation Depth estimation with normal camera

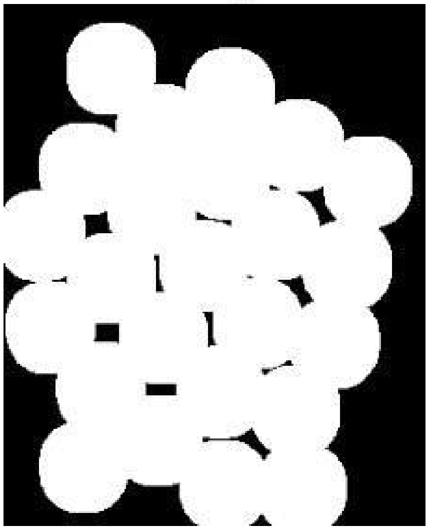
Theory:

The watershed algorithm uses topographic information to divide an image into multiple segments or regions.
The algorithm views an image as a topographic surface, each pixel representing a different height.
The watershed algorithm uses this information to identify catchment basins, similar to how water would collect in valleys in a real topographicmap.
The watershed algorithm identifies the local minima, or the lowest points, in the image.
These points are then marked as markers.
The algorithm then floods the image with different colors, starting from these marked markers.
As the color spreads, it fills up the catchment basins until it reaches the boundaries of the objects or regions in the image.

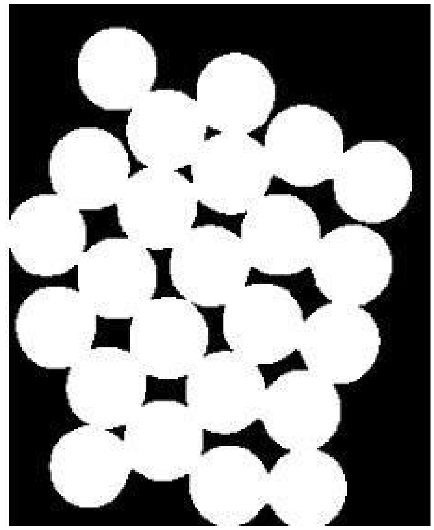
The catchment basin in the watershed algorithm refers to a region in the image that is filled by the spreading color starting from a marker . The catchment basin is defined by the boundaries of the object or region in the image and the local minima in the intensity values of the pixels. The algorithm uses the catchment basins to divide the image into separate regions and then identifies the boundaries between the basins to create a segmentation of the image for object recognition, image analysis, and feature extraction tasks.

The whole process of the watershed algorithm can be summarized in the following steps:
Marker placement: The first step is to place markers on the local minima, or the lowest points, in the image. These markers serve as the starting points for the flooding process.
Flooding: The algorithm then floods the image with different colors, starting from the markers. As the color spreads, it fills up the catchment basins until it reaches the boundaries of the objects or regions in the image.
Catchment basin formation: As the color spreads, the catchment basins are gradually filled, creating a segmentation of the image. The resulting segments or regions are assigned unique colors, which can then be used to identify different objects or features in the image.
Boundary identification: The watershed algorithm uses the boundaries between the different colored regions to identify the objects or regions in the image. The resulting segmentation can be used for object recognition, image analysis, and feature extraction tasks.

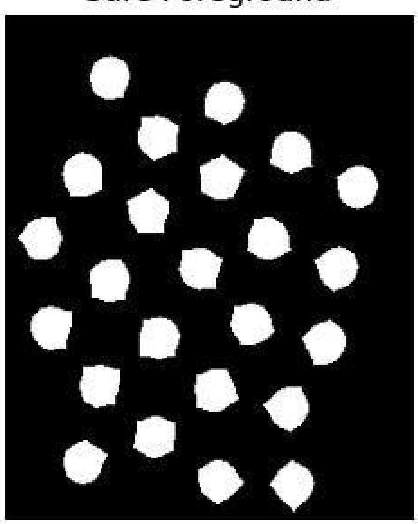
Sure Background



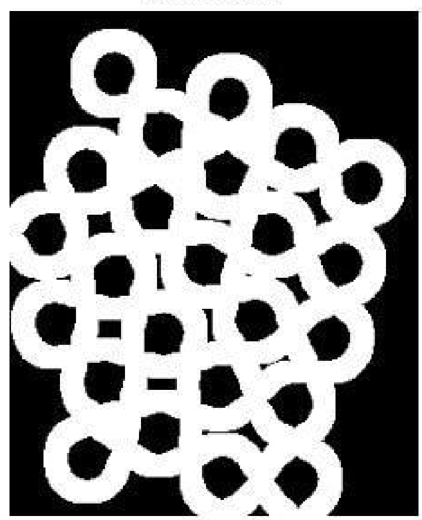
Distance Transform



Sure Foreground



Unknown




Input
GrabCut Mask
Output

Code:

```
import cv2
import numpy as np
from IPython.display import Image, display
from matplotlib import pyplot as plt

# Plot the image
def imshow(img, ax=None):
  if ax is None:
      ret, encoded = cv2.imencode(".jpg", img)
      display(Image(encoded))
  else:
      ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
      ax.axis('off')
```

```
#Image loading
img = cv2.imread("/content/imge1.jpg")
```

```
#image grayscale conversion
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # Show image
imshow(img)
```

```
#Threshold Processing
ret, bin_img = cv2.threshold(gray, 0, 255,
cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
print("threshold image")
imshow(bin_img)
# noise removal
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
bin_img = cv2.morphologyEx(bin_img,
cv2.MORPH_OPEN,
kernel, iterations=2)
print("noise removal")
imshow(bin_img)
# Create subplots with 1 row and 2 columns
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(8, 8)) # sure background area
sure_bg = cv2.dilate(bin_img, kernel, iterations=3)
```

```
imshow(sure_bg, axes[0,0])
axes[0, 0].set_title('Sure Background')
```

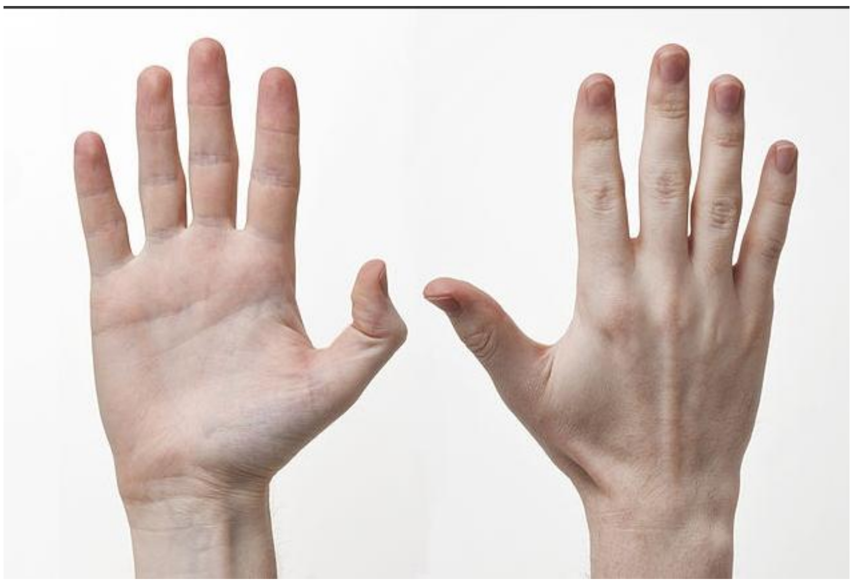
```
# Distance transform
dist = cv2.distanceTransform(bin_img, cv2.DIST_L2, 5)
imshow(dist, axes[0,1])
axes[0, 1].set_title('Distance Transform')
```

```
#foreground area
ret, sure_fg = cv2.threshold(dist, 0.5 * dist.max(), 255, cv2.THRESH_BINARY)
sure_fg = sure_fg.astype(np.uint8)
imshow(sure_fg, axes[1,0])
axes[1, 0].set_title('Sure Foreground')
```

```
# unknown area
```

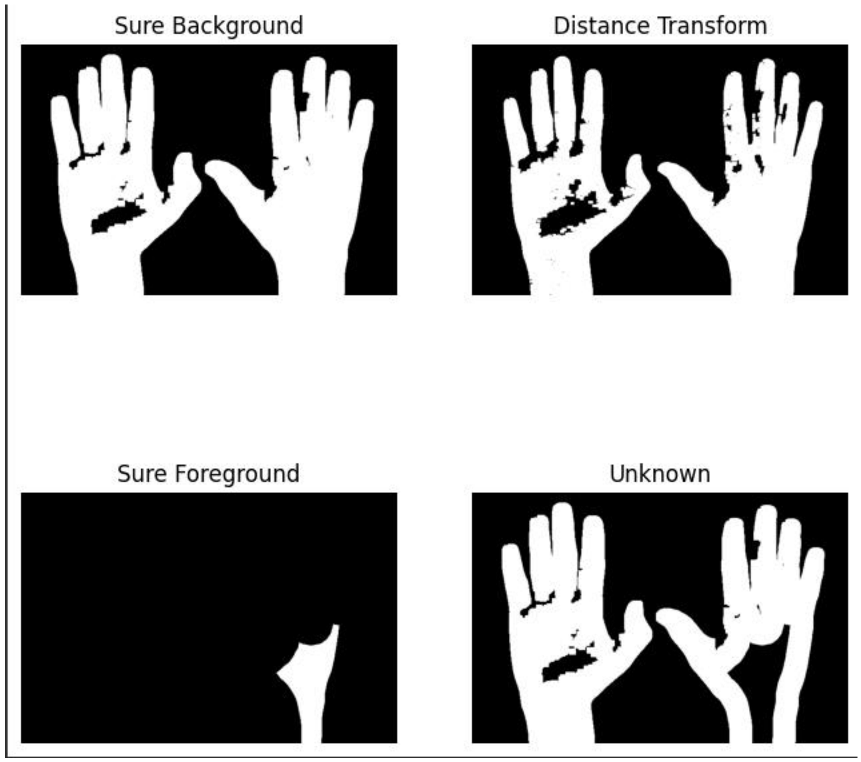
```
unknown = cv2.subtract(sure_bg, sure_fg)
imshow(unknown, axes[1,1])
axes[1, 1].set_title('Unknown')

plt.show()
```

Output :




threshold image


noise removal
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB


Sure Background
Distance Transform
Sure Foreground
Unknown

Conclusion:

In conclusion, our study delved into the watershed algorithm's utility for image segmentation, a potent method for separating objects from complex backgrounds. Despite its effectiveness, this approach necessitates meticulous parameter tuning and preprocessing. Notably, the algorithm's strengths lie in its use of gradient information and local minima as markers, enabling precise segmentation. The segmentation process involves grayscale conversion, thresholding, and noise reduction, followed by the calculation of the distance transform to identify foreground, background, and "unknown" regions. Each step is visually represented in separate subplots. Further refinements are required to optimize segmentation quality and identify specific objects of interest, marking the groundwork for future research and applications.