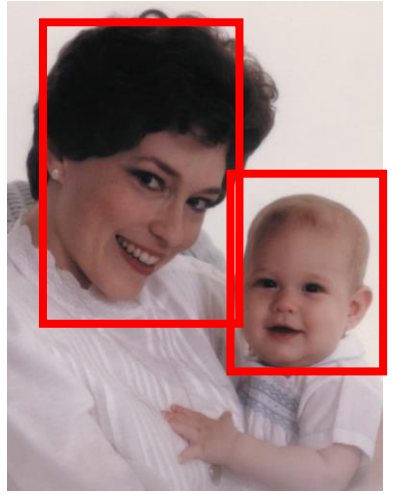
Object Detection

What is object detection?

- Object detection is a computer vision technique for locating instances of objects in images or videos.
  - An object can be a cup, a chair, a person, etc.



Object detection

- Why object detection is a hard problem?
  - Objects can exhibit variations in scale, illumination, viewpoint, intra-class variations
- Where is object detection used in the real-world?
  - Detect the presence of a person/a face in an image
    - Security systems
  - Object tracking in video
  - Self-driving cars

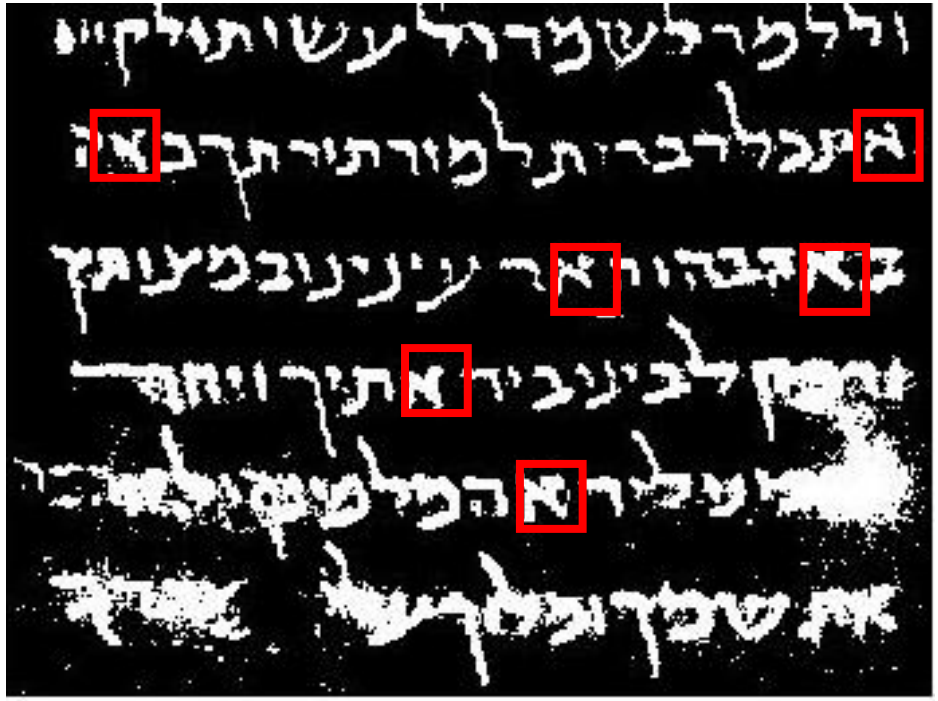
Types of object detection

- Object detection can be classified into machine learning-based approaches (ML) and deep learning-based (DL) approaches
  - ML-based approaches
    - computer vision techniques are used to look at various features of an image, such as the HOG or SIFT or color histograms, to identify groups of pixels that may belong to an object.
  - DL-based approaches
    - utilize convolutional neural networks (CNNs) to object detection
    - features don’t need to be defined by user

Template matching

- Template matching is a form of object detection
  - A template image: contains the object we are searching
  - Source image: the image where we search for our object

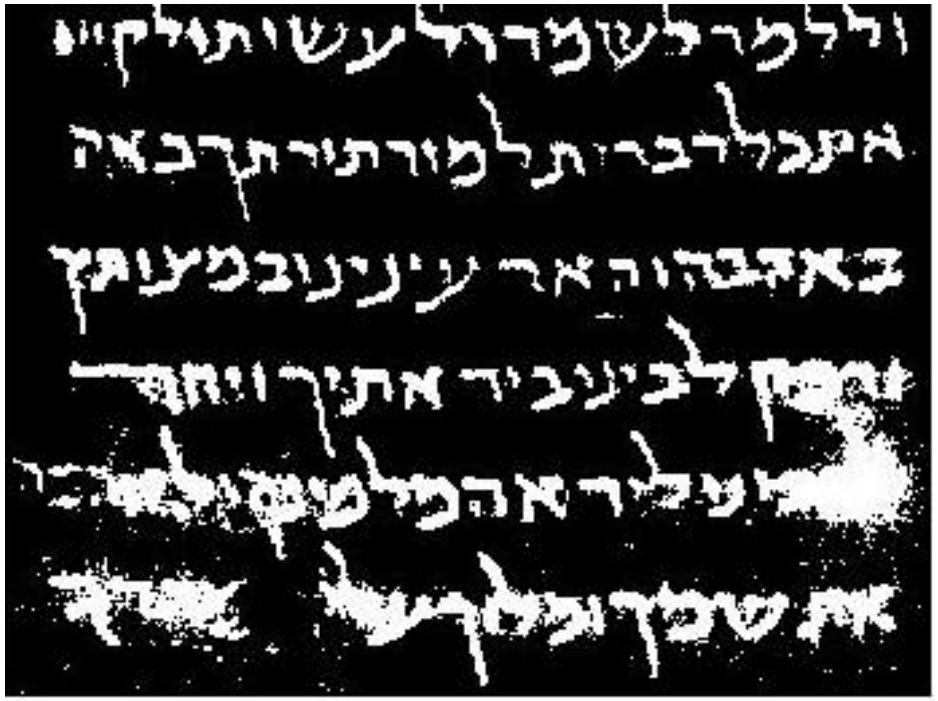
Template/query/ model



Source image

Template matching

- A template is slided from left-to-right and top-to-bottom across the source image
- At each *(x, y)* location, a metric is calculated to represent how “good” or “bad” the match is
- For different metrics see the [OpenCV documentation]



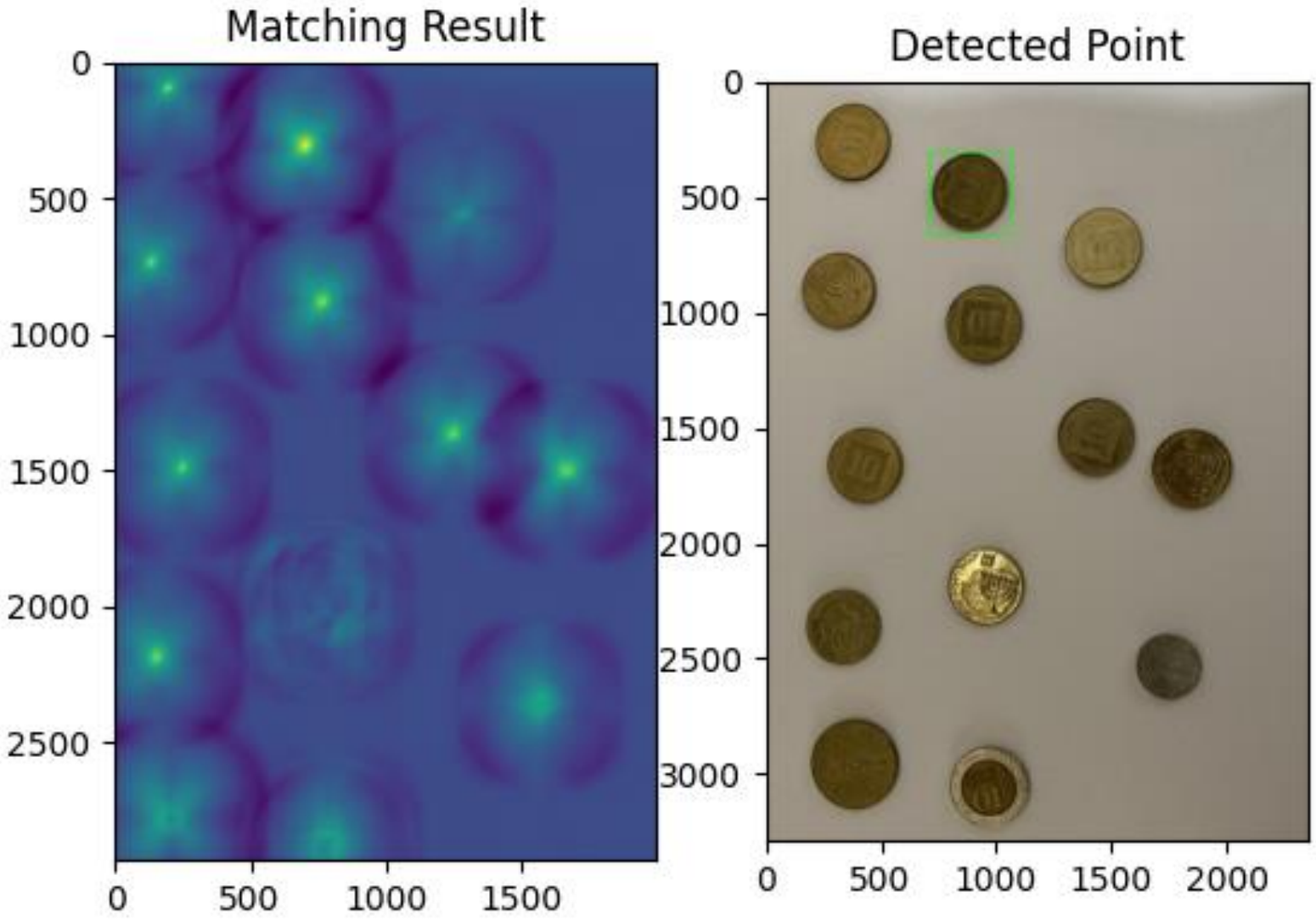
Template matching in OpenCV

result=cv2.matchTemplate(image, templ, method[, result[, mask]])

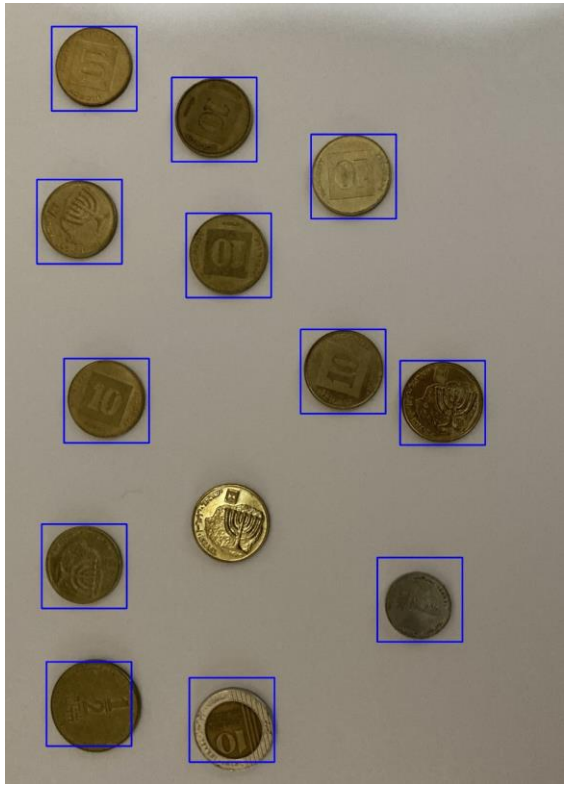
mage	Image where the search is running. It must be 8-bit or 32-bit floating-point.
templ	Searched template. It must be not greater than the source image and have the same data type.
result	Map of comparison results. It must be single-channel 32-bit floating-point. If image is $W \times H$ and templ is $w \times h$, then result is $(W-w+1) \times (H-h+1)$.

```
img = cv2.imread(sourceFilename)
imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
template = cv2.imread(templateFilename)
templateGray = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)
h,w = template.shape[:2]

# Apply template Matching
res = cv2.matchTemplate(imgGray, templateGray, cv2.TM_CCOEFF)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)


top_left = max_loc
bottom_right = (top_left[0] + w, top_left[1] + h)

imgCopy = img.copy()
cv2.rectangle(imgCopy, top_left, bottom_right, (0,255,0), 6)

imgCopy = cv2.cvtColor(imgCopy, cv2.COLOR_BGR2RGB)
plt.subplot(121), plt.imshow(res)
plt.title('Matching Result'), plt.xticks(), plt.yticks()
plt.subplot(122), plt.imshow(imgCopy)
plt.title('Detected Point'), plt.xticks(), plt.yticks()
plt.show()
```

Template


Matching Result
0
500
1000
1500
2000
2500
0
500
1000
1500
Detected Point
0
500
1000
1500
2000
2500
3000
0
500
1000
1500
2000

Multiple objects detection



```
res = cv2.matchTemplate(imgGray, templateGray, cv2.TM_CCOEFF)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
threshold = 0.3 * max_val
imgCopy = img.copy()

while max_val >= threshold:
    top_left = max_loc
    bottom_right = (top_left[0] + w, top_left[1] + h)
    cv2.rectangle(imgCopy, top_left, bottom_right, (0, 0, 255), 6)
    # non-maximum supression
    res[max([0,max_loc[1]-h//2]):max_loc[1]+h,max([0,max_loc[0]-w//2]):max_loc[0]+w] = 0
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
```

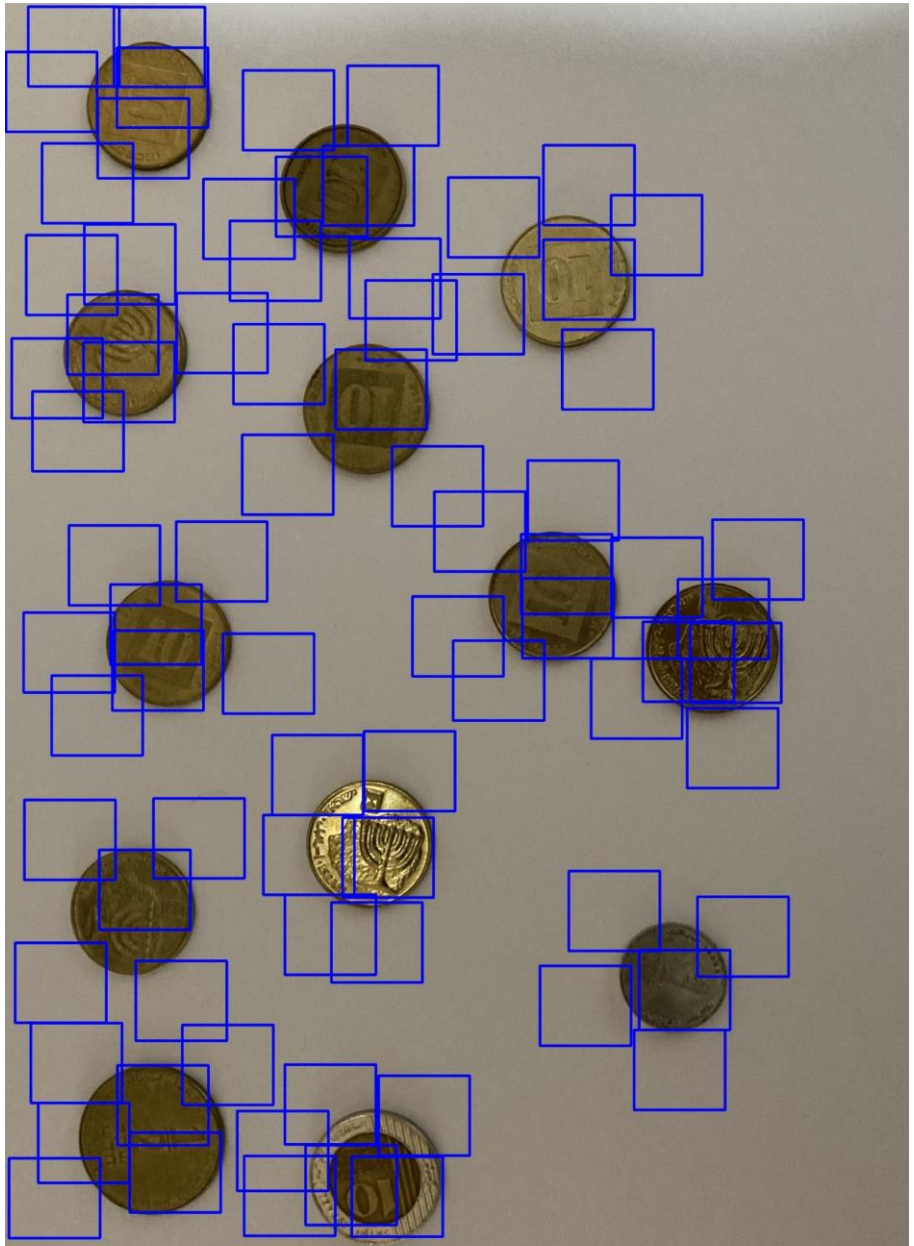
What can you see?

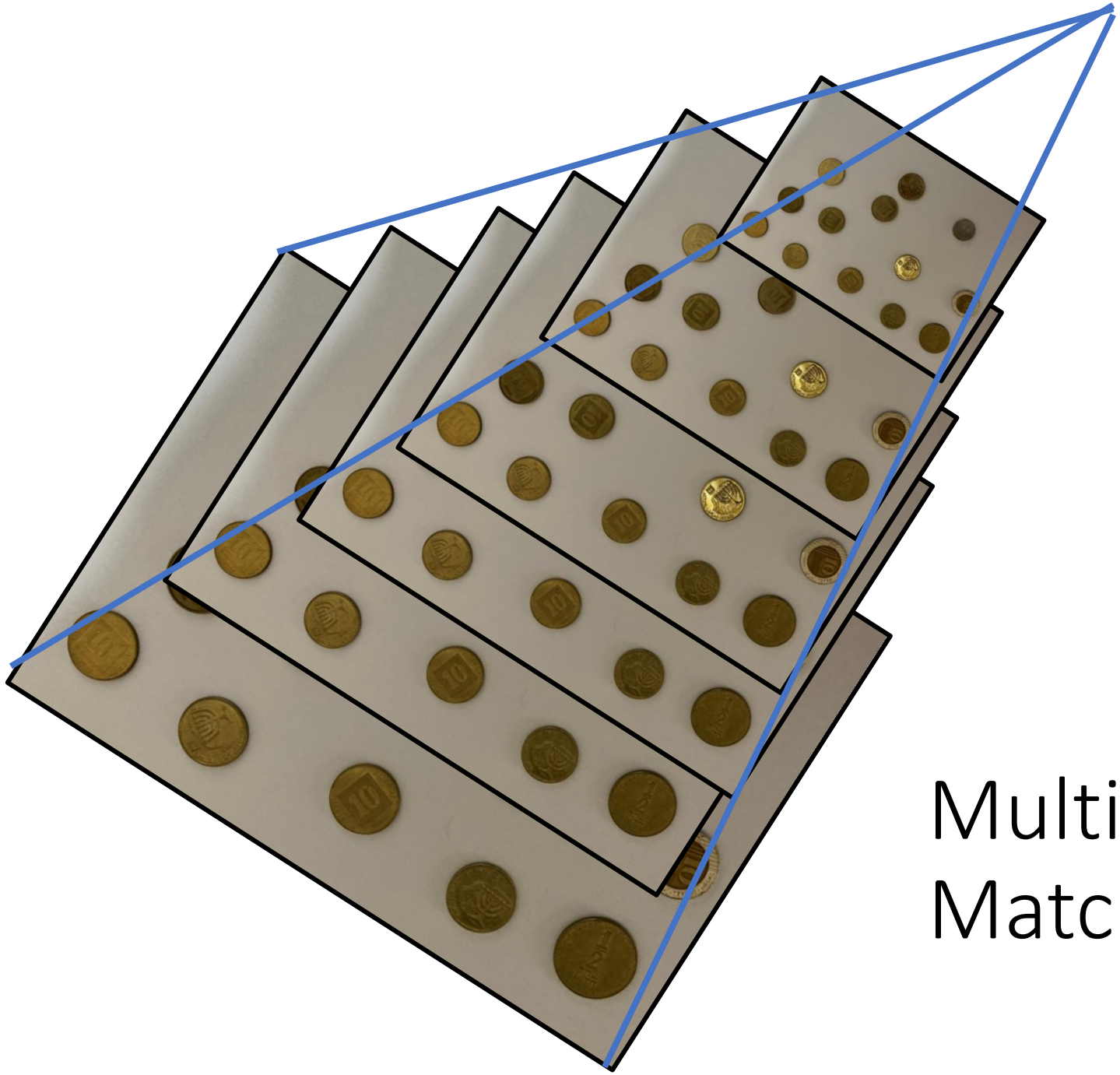
Matching on the edge map representation

```
imgCanny= cv2.Canny(imgGray, 50, 200)
templateCanny = cv2.Canny(templateGray, 50, 200)
res = cv2.matchTemplate(imgCanny, templateCanny, cv2.TM_CCOEFF)


min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
threshold = 0.1 * max_val
imgCopy = img.copy()

while max_val >= threshold:
    top_left = max_loc
    bottom_right = (top_left[0] + w, top_left[1] + h)
    cv2.rectangle(imgCopy, top_left, bottom_right, (0, 0, 255), 6)
    # non-maximum supression
    res[max([0, max_loc[1] - h // 2]):max_loc[1] + h, max([0, max_loc[0] - w // 2]):max_loc[0] + w] = 0
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)

cv2.imwrite('TemplateMatchingResCanny.jpg', imgCopy)
```

What about different scales?

Template





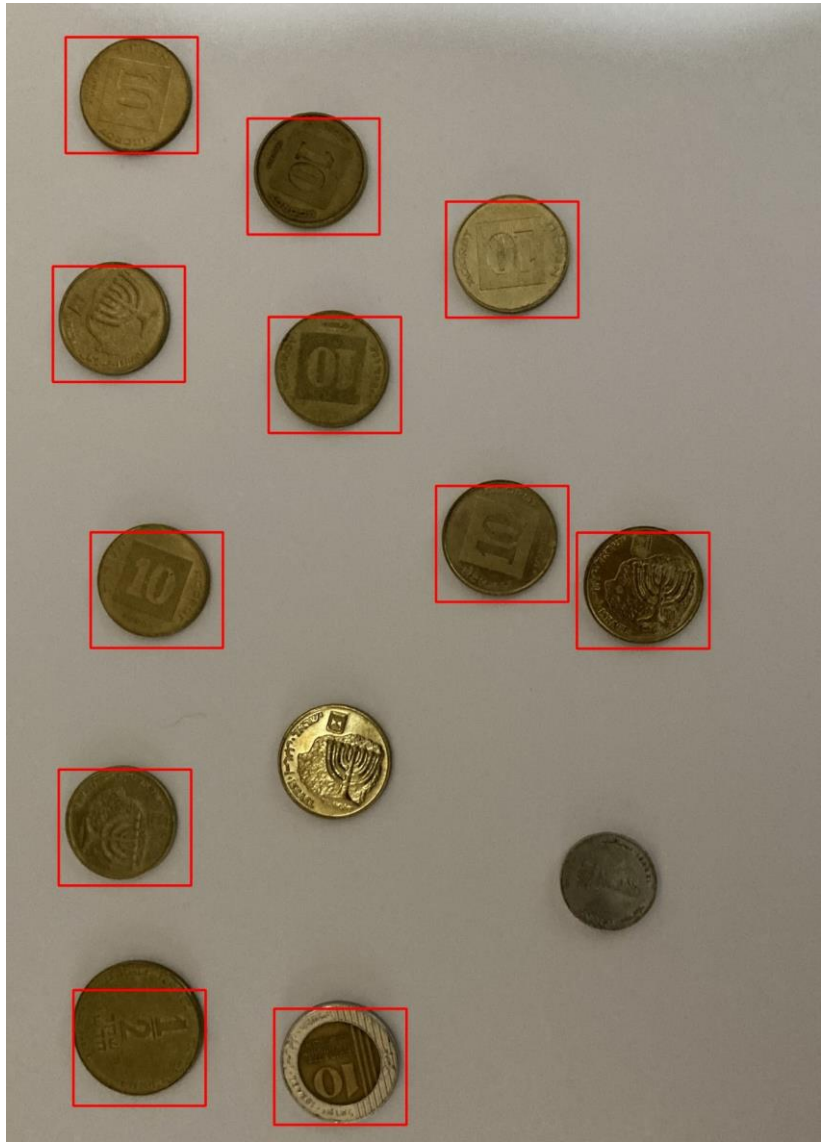
Multi-Scale template Matching

Multi-Scale template Matching

1. Loop over the input image at multiple scales
2. Apply template matching at each scale and keep track of the match with the largest correlation coefficient
3. After looping over all scales, take the region with the largest correlation coefficient

Multi-Scale template Matching

Template



Template matching - Summary

- Pros
  - Simple
  - Easy to implement
- Cons
  - Works under controlled conditions
  - Not robust to significant changes in viewpoint, occlusion and non-affine transformations