1 Python correction

1.1 Binary attribute filtering

If bw is a binary image, the different attribute are evaluated with regionprops and filtered according to the given upper and lower thresholds.

```python
def bwFilter(bw, attribute, thresholds):
    """
    binary filtering according to attribute
    bw: binary image
    attribute: string representing attribute, defined by regionprops
    thresholds: threshold values, inside which objects are removed
    returns binary image
    """
    F = bw.copy();
    L = label(bw);
    for region in regionprops(L):
        a = getattr(region, attribute);
        if  a < thresholds[1] and a>= thresholds[0]:
            F[L==region.label] = False;
    return F;
```

1.2 Grayscale filtering

The grayscale filtering is the previous binary filtering process applied to all level-sets of the original image.

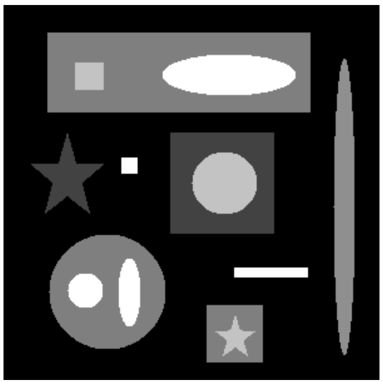
```python
def grayFilter(I, attribute, thresholds):
    """
    grayscale image filtering by attribute
    for 8 bits unsigned images
    I: original grayscale image (N, M) ndarray
    attribute: string representing attribute, defined by regionprops
    thresholds: threshold values, inside which objects are removed
    returns grayscale filtered image
    """
    N, M = I.shape
    F = np.zeros((N, M, 256));
    for s in range(256):
        F[:,:,s] = s * bwFilter(I>=s, attribute=attribute, thresholds=
            ↪ thresholds);

    # reconstruction
    R = np.amax(F, axis=2);
    return R;
```

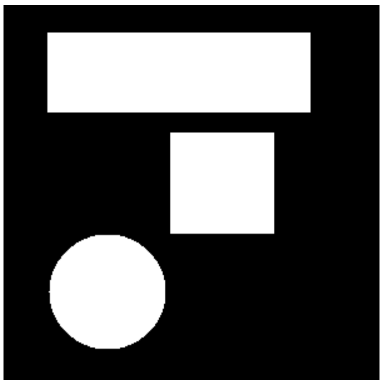
Then, for each level, the binary set is filtered by some attribute, and the resulting image is reconstructed by taking the maximum value on all levels.

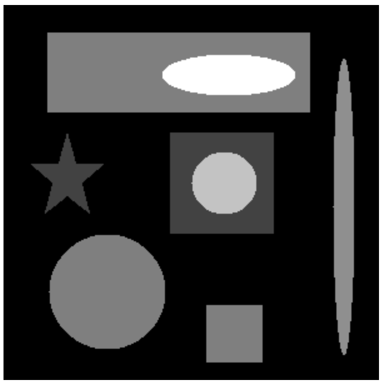
The shape filtering process do not take the results from the regionprops function. The shape is filtered by using the morphological function of reconstruction.

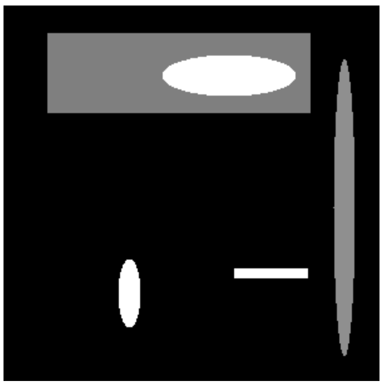
```python
def shapeFilter(I, selem=m.square(25)):
    """
    image filtering when attribute is a shape of a given size, defined by
        ↪ selem
    I: grayscale image
    selem: structuring element
    returns: grayscale filtered image
    """
    N, M = I.shape
    F = np.zeros((N, M, 256));
    for s in range(256):
        F[:,:,s] = s * m.reconstruction(m.opening(I>=s, selem=selem), I>=
            ↪ s);

    # reconstruction
    R = np.amax(F, axis=2);
    return R;
```

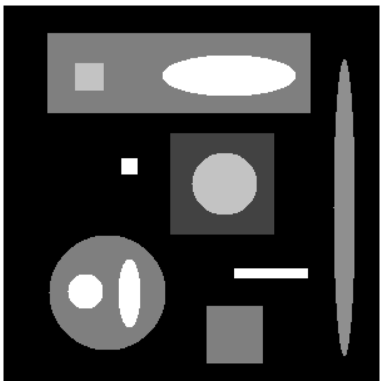
Results are illustrated in Fig.1.


(a) Original image.


(b) Binary filtering by area (level 50 to get the binary image, 5000 pixels as a threshold).


(c) Filtering by area (1000 pixels).


(d) Filtering by elongation (0.75).


(e) Filtering by convexity (0.75).

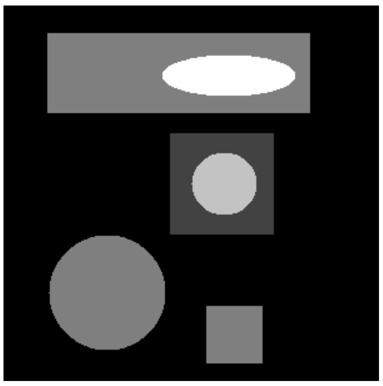

(f) Filtering objects bigger than a square (of size 25 × 25 pixels).

Figure 1: Attribute filtering examples.