Reconhecimento de Texto

Cupons Fiscais

Alinhamento

```
def matrix_rotation(width, height, alpha):
    return cv2.getRotationMatrix2D((width/2, height/2), alpha, 1)

def rotate(alpha, image, verbose=False):
    if verbose:
        print "Rotate image"
    height = image.shape[0]
    width = image.shape[1]
    mtz_rotation = matrix_rotation(width, height, alpha)
    return cv2.warpAffine(image, mtz_rotation, (width, height))
```

Alinhamento

```
def angle(image_gray, nlines, alpha, verbose=False):
    if verbose:
        print "Get angle"
    width = image_gray.shape[1]
    edges = cv2.Canny(image_gray, 50, 150, apertureSize=3)
    angle = None
    begin = 0
    end = width
    while angle is None:
        middle = (begin + end)/2
        lst_angle = []
        lines = cv2.HoughLines(edges, 1, np.pi/180, middle)
        try:
            for rho, theta in lines[0]:
                theta = 180*theta/np.pi
                if theta < alpha:
                    lst_angle.append(theta)
                elif theta > 180 - alpha:
                    lst_angle.append(-(180 - theta))
            if len(lst_angle) == nlines or end - begin == 1:
                angle = np.average(lst_angle)
            elif len(lst_angle) < nlines:
                end = middle
            elif len(lst_angle) > nlines:
                begin = middle
        except TypeError:
            end = middle
    return angle
```

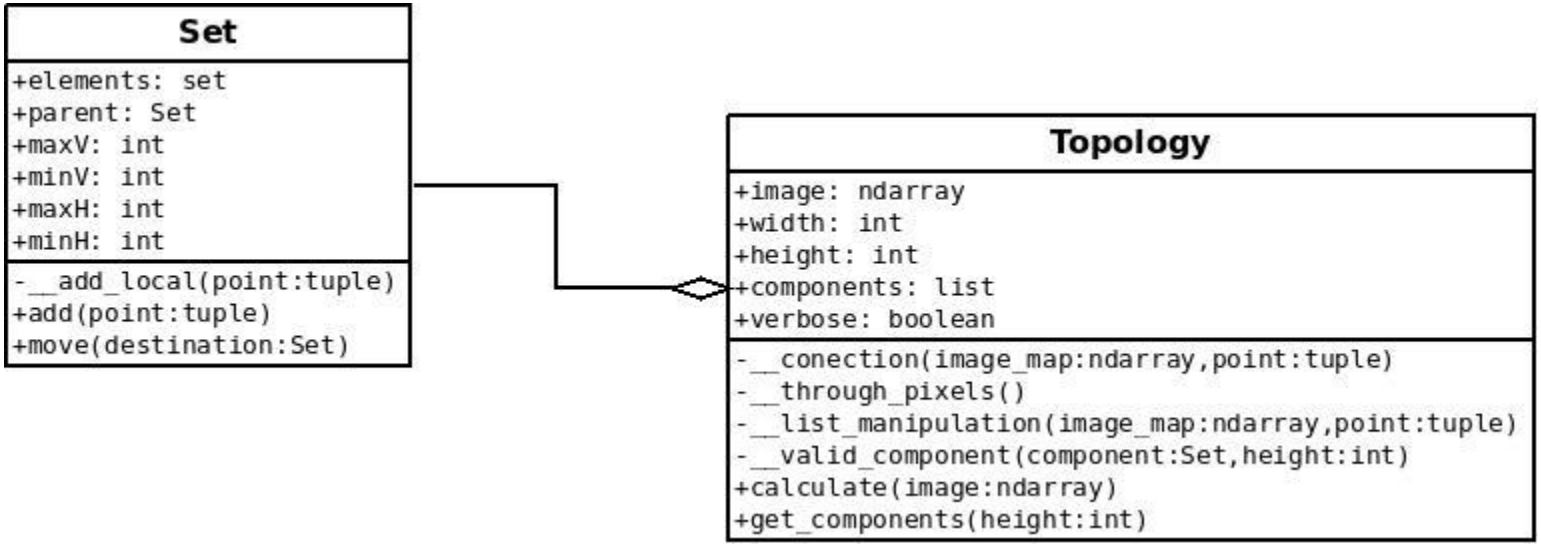
Limiarização

```
def adaptive_threshold(image_gray, blur=True, verbose=False):
    if verbose:
        print "Thresholding"
    if blur:
        img = cv2.medianBlur(image_gray, 3)
        img = cv2.fastNlMeansDenoising(img, None, 10, 7, 21)
    return cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
```

Fechamento Horizontal

```
def horizontal_close(image_bin, lenght=None, verbose=False):
    height, width = image_bin.shape
    if verbose:
        print "Making close"
    if lenght is None:
        lenght = width/30
    cv2.bitwise_not(image_bin, image_bin)
    kernel = np.ones((1, int(lenght)), np.uint8)
    image_bin = cv2.morphologyEx(image_bin, cv2.MORPH_CLOSE, kernel)
    cv2.bitwise_not(image_bin, image_bin)
    return image_bin
```

Topologia


Set
+elements: set
+parent: Set
+maxV: int
+minV: int
+maxH: int
+minH: int
-__add_local(point:tuple)
+add(point:tuple)
+move(destination:Set)
Topology
+image: ndarray
+width: int
+height: int
+components: list
+verbose: boolean
-__conection(image_map:ndarray,point:tuple)
-__through_pixels()
-__list_manipulation(image_map:ndarray,point:tuple)
-__valid_component(component:Set,height:int)
+calculate(image:ndarray)
+get_components(height:int)