In [1]:
```
import sys
import cv2
import numpy as np
from matplotlib import pyplot as plt
from matplotlib.patches import Rectangle
sys.path.append('mask_bracket')
import params

%matplotlib inline
```

```
Using TensorFlow backend.
```

In [28]:
```
plt.rcParams["axes.edgecolor"] = "black"
plt.rcParams["axes.linewidth"] = 1

def show_image(image, title=None):
    channels = image.shape[2] if len(image.shape) == 3 else 1
    if channels == 3:
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    oldfigsize = plt.rcParams["figure.figsize"]
    plt.rcParams["figure.figsize"] = (image.shape[1]//200+1,image.shape[0]/
    plt.autoscale(enable=True, axis='both', tight=False)
    plt.axis('off')
    plt.imshow(image, aspect='equal', interpolation='gaussian', cmap='gray'
    plt.show()
    plt.rcParams["figure.figsize"] = oldfigsize
```

Using our trained UNET model to highlight brackets. Wrapped it in highlight_brackets function

Train script is located at mask_bracket/train.py

In [29]:
```
def highlight_brackets(image):
  input_size = params.input_size
  model = params.model_factory()
  model.load_weights('mask_bracket/baseline.h5')
  img = params.simplify_image(image)

  orig_height, orig_width = img.shape
  img = cv2.resize(img, (input_size, input_size))
  img = np.array(img, np.float32) / 255
  preds = model.predict(img.reshape(1,input_size,input_size,1))
  preds = np.squeeze(preds, axis=3)
  pred = preds[0]
  pred = cv2.resize(pred, (orig_width, orig_height))
  res = (pred >= 0.5).astype(int)*255
  return res.astype("uint8")
```

In [30]:
```
original_image = cv2.imread('sample6.jpg')
original_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)
image = cv2.imread('sample6.jpg', cv2.IMREAD_GRAYSCALE)
```

Getting masks for brackets

In [31]:
```
highlighted = highlight_brackets(image)
show_image(highlighted)
```

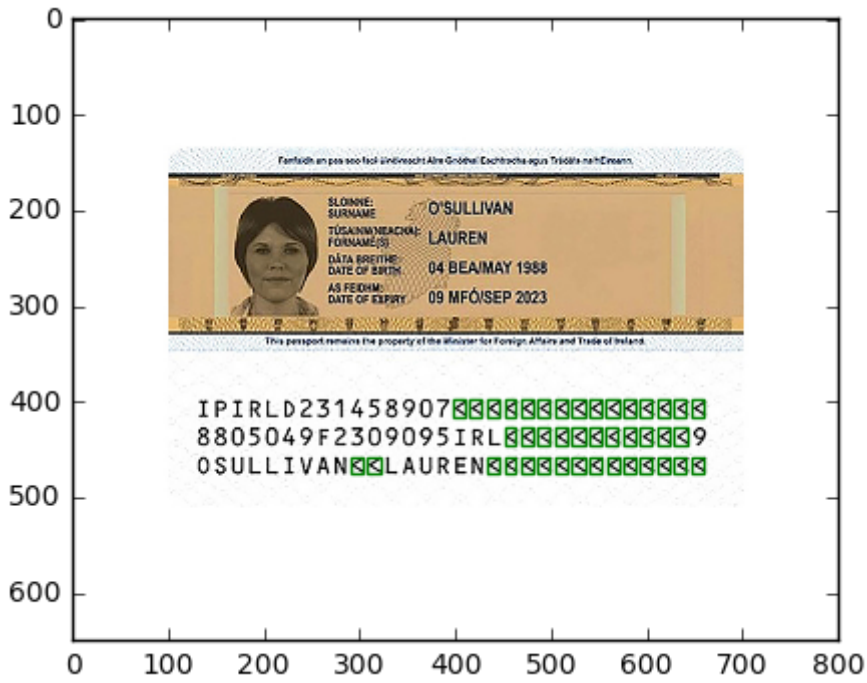
Then, it will be easy to find boundaries with cv2.findContours

In [32]:
```
def get_boundaries(imgray):
    _, contours, _  = cv2.findContours(imgray, cv2.RETR_LIST, cv2.CHAIN_APP
    boxes = []
    for cords in contours:
         cords = np.squeeze(cords)
         if len(cords.shape)>1:
             x = [a[0] for a in cords]
             y = [a[1] for a in cords]
             minx, maxx = np.min(x), np.max(x)
             miny, maxy = np.min(y), np.max(y)
             boxes.append((minx,miny,maxx,maxy))
    return boxes

def draw_boundaries(image, boxes):
    fig,ax = plt.subplots(1)
    ax.imshow(image)
    for (minx,miny,maxx,maxy) in boxes:
         rect = Rectangle((minx-1,miny-3),maxx-minx+1,maxy-miny+3,linewidth
         ax.add_patch(rect)
```

Let's see the result for this step

In [33]:
```
boxes = get_boundaries(highlighted)
draw_boundaries(original_image, boxes)
```

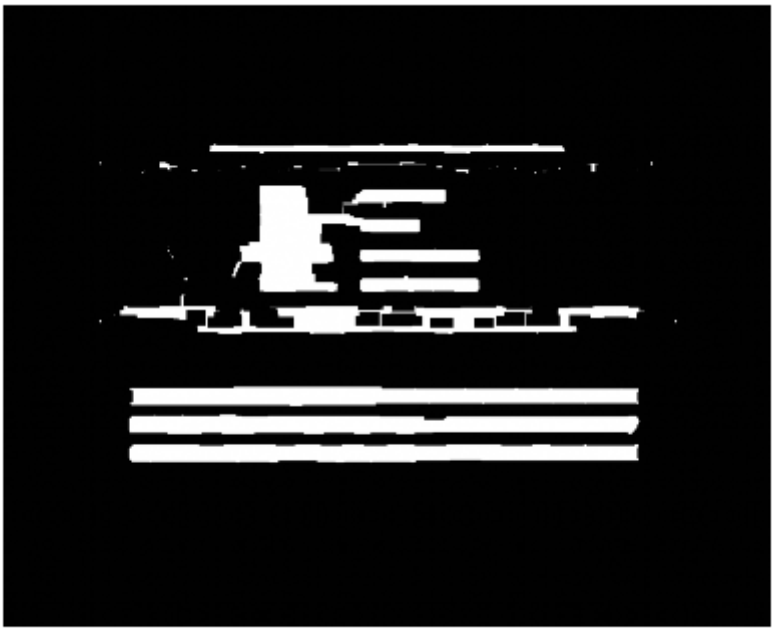


In [34]:
```
simplified_image = params.simplify_image(image)
```

Converting image to united lines

In [35]:
```
def get_lines(img):
    gradX = cv2.morphologyEx(simplified_image, cv2.MORPH_CLOSE, cv2.getStru
    return cv2.threshold(gradX, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU
lines = get_lines(simplified_image)
```

In [36]:
```
show_image(lines)
```



Knowing bracket positions we can get miny and maxy for each bracket row

In [37]:
```
same_row_dist = 5
bracket_min_y = np.unique([int(box[1]/same_row_dist) for box in boxes])*sam
bracket_max_y = np.unique([int(np.ceil(box[3]/same_row_dist)) for box in bo
bracket_heights = list(zip(bracket_min_y, bracket_max_y))
```

For each bracket row we can find minX and maxX boundaries

In [38]:
```
brackets_lines = []
for i in range(len(bracket_heights)):
    miny, maxy = bracket_heights[i]
    line = np.where(lines[miny:maxy,:] > 100)[1]
    brackets_lines.append( (np.min(line), miny, np.max(line),maxy) )
```

Showing the final result

In [39]:
```
draw_boundaries(original_image, brackets_lines)
```