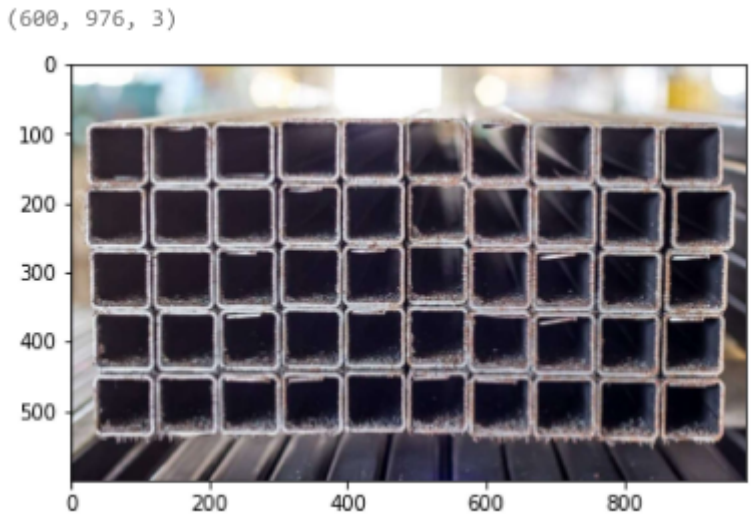
Metody Rozpoznawania Obrazów

Laboratorium 2

Maria Polak

1. Obraz wejściowy

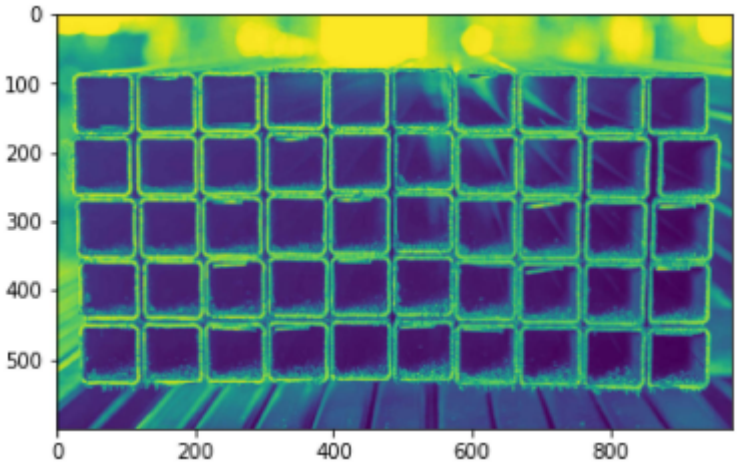


Rysunek1. Wczytany obraz

2. Skala szarości

Przekształcenie obrazu do skali szarości odbyło się przez użycie funkcji Conv2D z jednym kanałem na wyjściu, paddingiem “same”, stride równym 1 oraz wagami [0.4 , 0.4, 0.2].

```
layers.Conv2D(1, (1, 1), input_shape=pipes.shape, padding='same', use_bias=False,
                                  weights=[np.array([[[[0.4], [0.4], [0.2]]]])],
trainable=False)
```

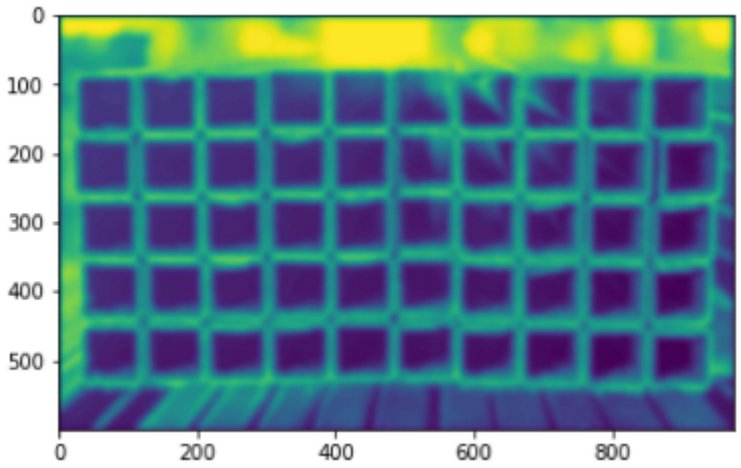


Rysunek 2. Obraz w skali szarości

3. Gaussowskie rozmycie obrazu

Celem rozmycia została przygotowany filtr rozmiaru 27, sigma została ustawiona na wartość 2.5. Obydwie liczby zostały wybrane poprzez przeprowdzanie kilku eksperymentów z ich wartościami.

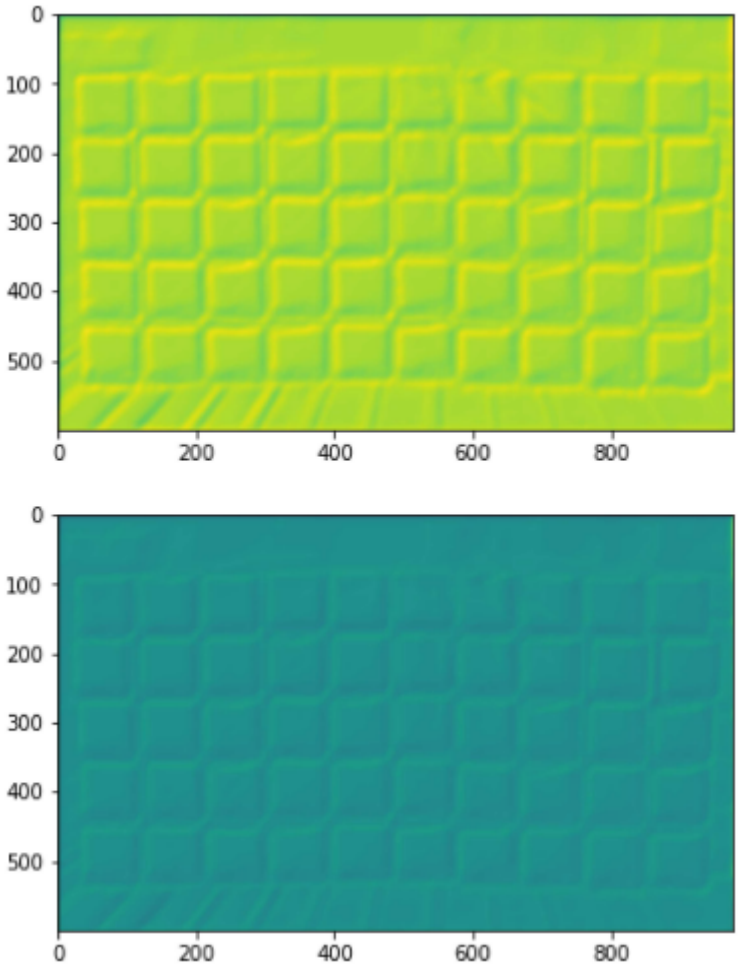
```
def gaussian_filter(size, sigma=1):
    k = int((size - 1) / 2)
    prob = [np.exp(-z * z / (2 * sigma * sigma)) / np.sqrt(2 * np.pi * sigma * sigma)
for z in range(-k, k + 1)]
    return np.outer(prob, prob)
n = 27
sigma = 2.5
layers.Conv2D(1, (n, n), use_bias=False,
                          weights=[gaussian_filter(n, sigma).reshape((n, n, 1, 1))],
                          padding='same', trainable=False)
```



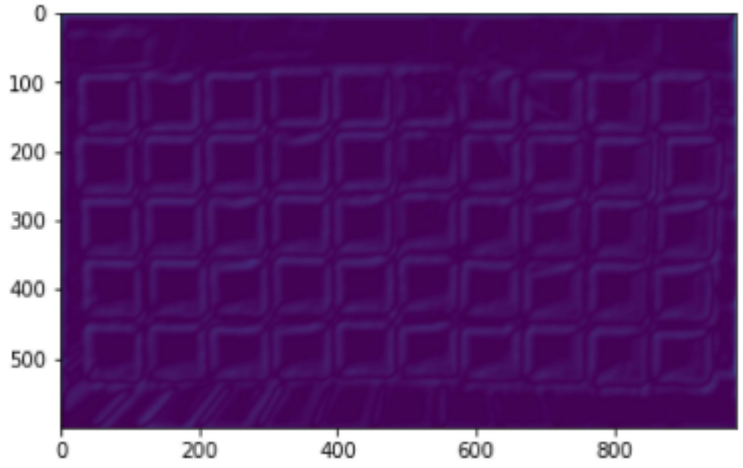
Rysunek 3. Obraz po rozmyciu

4. Filtry Sobela

Aby poznać składowe gradientu obraz został przetworzony przez filtry Sobela.



Rysnek 4. Zawartość obydwóch kanałów po zastosowaniu filtrów Sobela



Rysunek 5. Połączone kanały

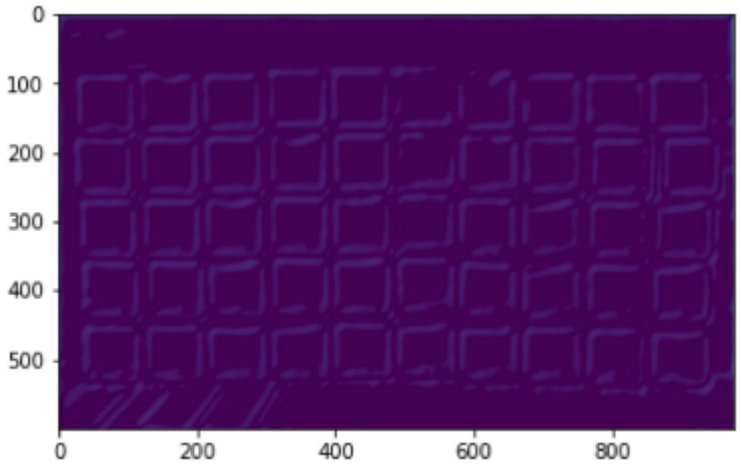
5. Intensywność gradientu

Następnie kanały zostały scalone przez wykonanie przekształceń sprecyzowanych w instrukcji.

```
pow2_layer = layers.Lambda(lambda x: x * x)
sum_layer = layers.Conv2D(1, (1, 1), padding='same', use_bias=False,
                          weights=[np.array([[[[1], [1]]]])],
trainable=False)
sqrt_layer = layers.Lambda(lambda x: tf.math.sqrt(x))

inputs = pow2_layer(inputs)
inputs = sum_layer(inputs)
inputs = sqrt_layer(inputs)
```

Odflitrowne zostały regiony z niewielkim gradientem. Wartość progu została dobrana empirycznie.



Rysunek 6. Wykryte krawędzie

6. Max Pooling

Rozdzielczość obrazu została zmniejszona poprzez użycie max poolingu. Pool_size został ustawiony na wartość 8.

```
layers.MaxPool2D(pool_size=(8, 8), padding='same', trainable=False)
```

7. Głosowanie

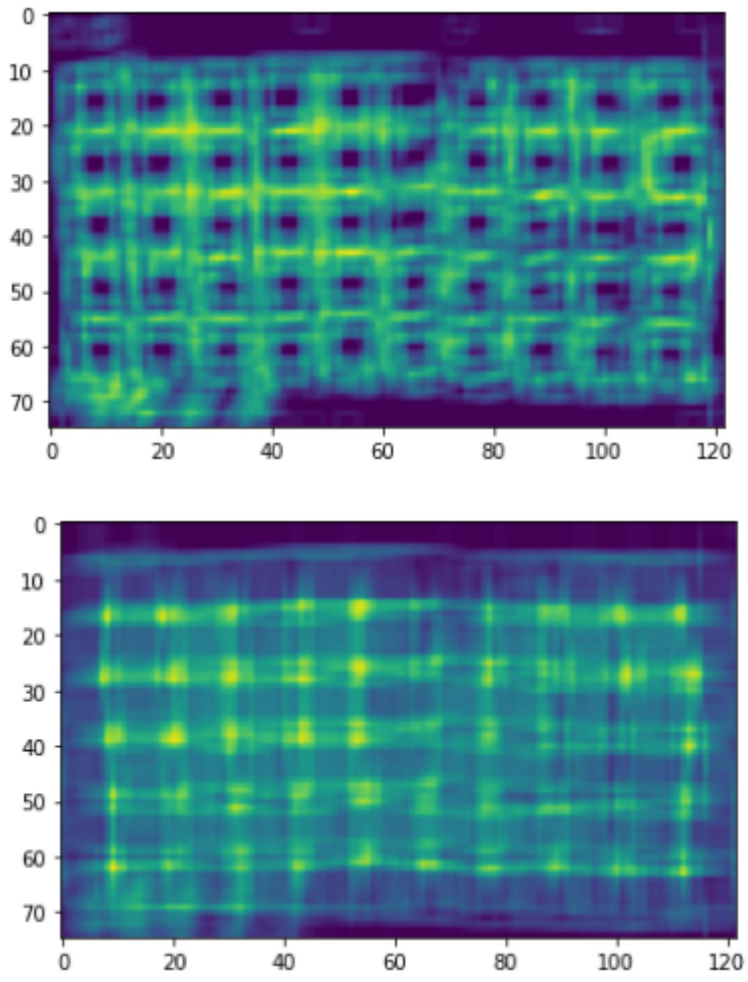
Przeprowadzone zostało głosowanie (transformata Hougha). Przygotowane zostały kwadraty (filtry) w rozmiarach 5x5 - 14x14, a następnie za pomocą funkcji Conv2DTranspose wykonane przeprowadzone zsotało głosowanie.

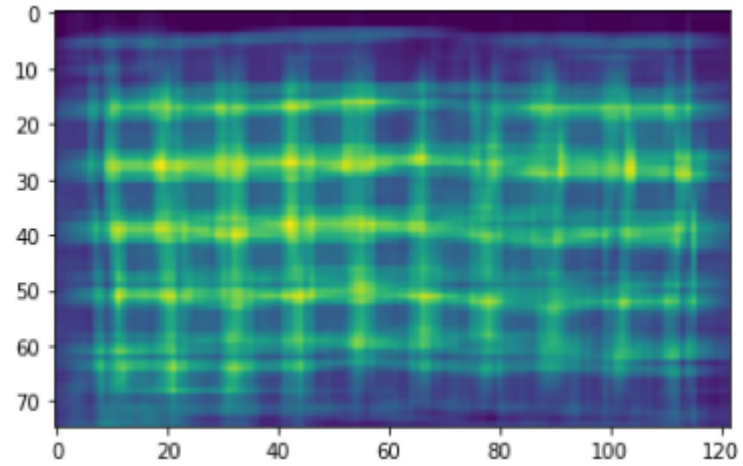
```
outputs = []
for x in range(5, 15):
```

```
   weights = np.zeros(shape=(x, x, 1, 1))
   weights[0, :] = 1
   weights[-1, :] = 1
   weights[:, 0] = 1
   weights[:, -1] = 1

   output = layers.Conv2DTranspose(1, kernel_size=(x, x),
weights=[weights], padding='same',
                                   trainable=False,
use_bias=False)(inputs)
   outputs.append(output)
```

Poniżej zamieszczono wyniki głosowania dla kilku rozmiarów:

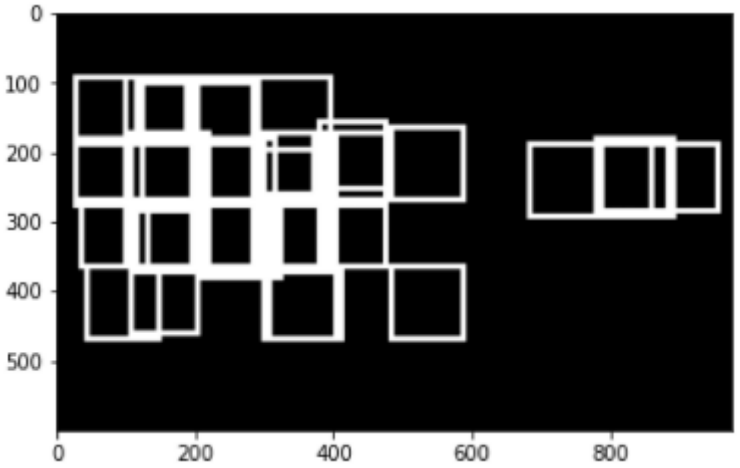




Rysunek 7. Wyniki głosowania dla rozmiarów 5x5, 11x11, 14x14

Z uzyskanego głosowania odfiltorwane zostały wszystkie wyniki poza 50 największymi.

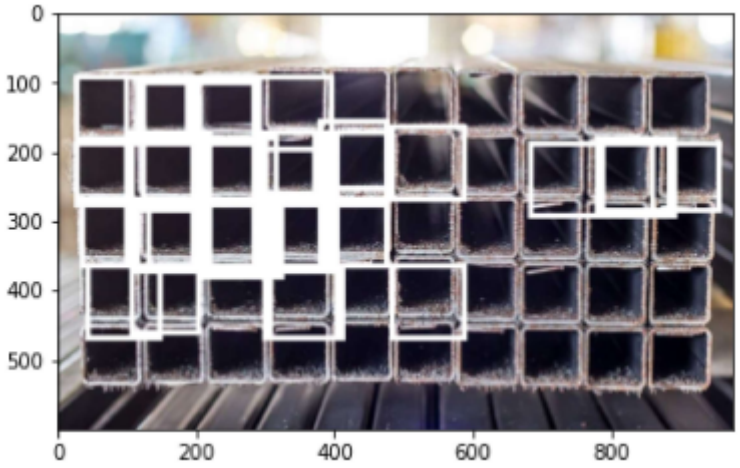
Zastosowana została potem konwolucja transponowana, aby uzyskać znalezione we wcześniejszych krokach kwadraty. A następnie został zastosowany upscaling, żeby wrócić do początkowego rozmiaru obrazka.



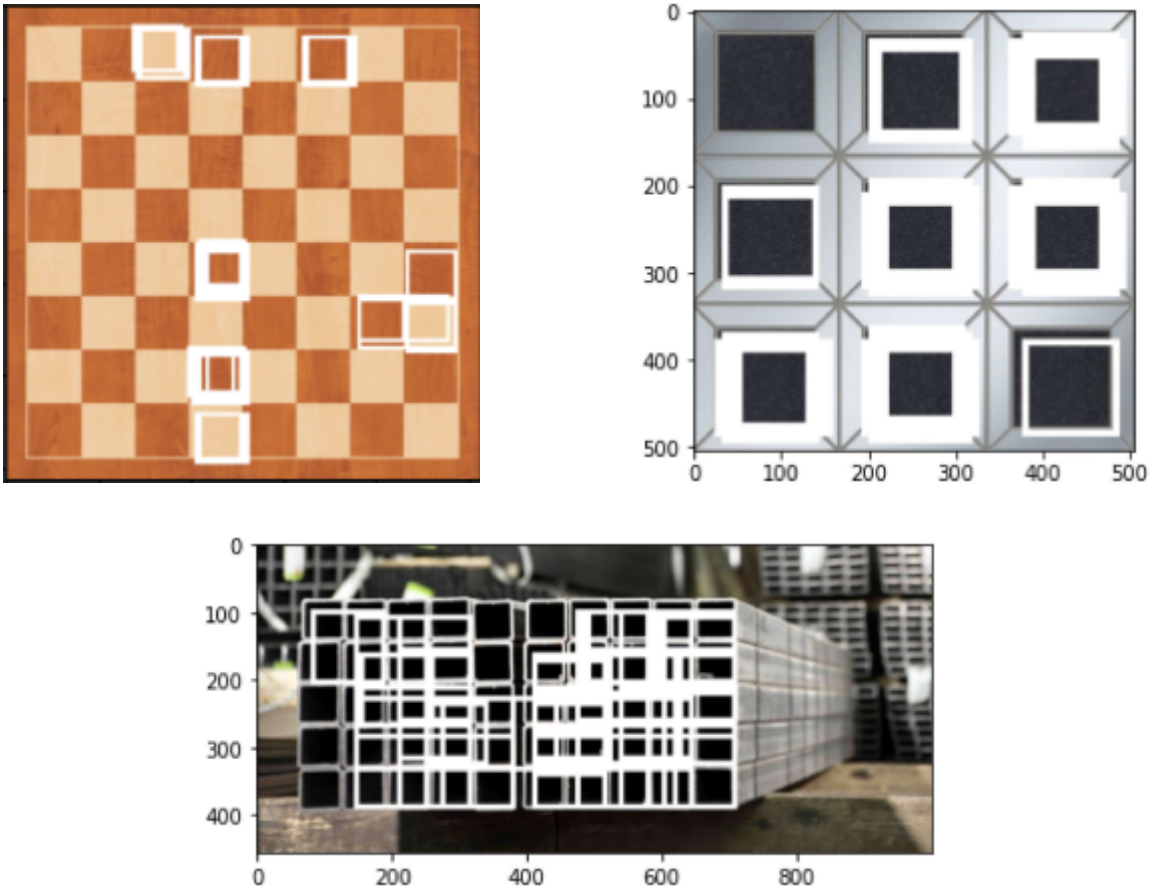
Rysunek 8. Wykryte kwadraty (różnych rozmiarów, po upscalingu)

8. Wyniki

Uzyskane wyniki nie są perfekcyjne. Tylko kilka kwadratów widocznych na obrazie zostało poprawnie odnalezionych, lecz stanowcza ich większość jest złego rozmiaru lub została całkowicie zignorowana. Pozycje tych kwadratów, które udało się wykryć, stosunkowo dobrze pokrywają się z pierwowzorem. Błędy najprawdopodobniej są spowodowane niedokładnością dobieranych parametrów - na niektórych etapach parametry były wybierane “na oko” bez wcześniejszej wiedzy, jak najlepiej je dostosować do implementowanego algorytmu. Dodatkowo zaimplementowana wersja metody Cannego jest wersją ukróconą o niektóre istotne kroki, co na pewno obniża jakość wykrycia krawędzi.



Rysunek 9. Wynik wykrycia kwadratów na orginalnym obrazie



Rysunek 10. Efekty wykrycia kwadratów na innych obrazach

Wyniki uzyskane na innych obrazach są podobne. Algorytm świetnie poradził sobie z mozaiką (mało dobrze widocznych kwadratów). Na szachownicy wykrył tylko kilka z obecnych na zdjęciu kwadratów. Natomiast na zdjęciu rur, był w większości przypadków dosyć niedokładny (kwadraty są przesunięte względem ich orginalnej pozycji).