Wykrywanie krawędzi i wierzchołków na obrazie w celu uzyskania siatki terenu miejskiego

Praca końcowa z przedmiotu Neuroplastyczność a neurodegeneracja: Mózg a komputer

Izabela Leszczyńska s16499

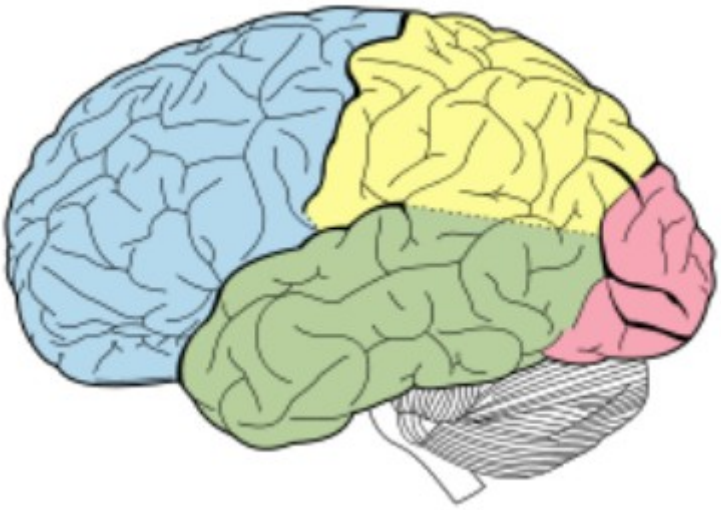
1. Wstęp

W tej pracy chcę zająć się opracowaniem mapy otoczenia na podstawie zdjęcia poprzez program komputerowy. Chcę wykorzystać w tym celu wykrywanie krawędzi oraz przetworzenie obrazu na ich podstawie. Rozwiązanie jest zaimplementowane w języku Python, ze względu na optymalność rozwiązania (implementacja od zera w Pythonie ma bardzo długi czas wykonywania ze względu na zbędne operacje wykonywane przez interpreter, biblioteki są modułami w języku C++, które minimalizują czas wykonania i zużycie pamięci), zostały użyte biblioteki dostępne w tym języku: CV2, Numpy i Matplotlib. Użyte zostało środowisko wykonawcze Colab.

2. Analiza obrazu w mózgu

Kora mózgowa, w której ulegają analizie informacje ze zmysłu wzroku, dzieli się na 4 płaty w obu z półkul:

- płat czołowy odpowiadający za myślenie, planowanie, pamięć, podejmowanie decyzji, ocenę sytuacji, wyuczone działania ruchowe, przewidywanie konsekwencji
- płat ciemieniowy odpowiadający za integrację bodźców wzrokowych oraz bodźców z innych zmysłów w celu orientacji przestrzennej
- płat skroniowy odpowiadający za język, bodźce słuchowe oraz pamięć długotrwałą i emocje
- płat potyliczny odpowiadający za percepcję wzrokową oraz przetwarzanie informacji wzrokowych



Rysunek 1: Mózg wraz z zaznaczonymi płatami: czołowy - niebieski, cieminiowy - żółty, skroniowy - zielony, potyliczny - czerwony

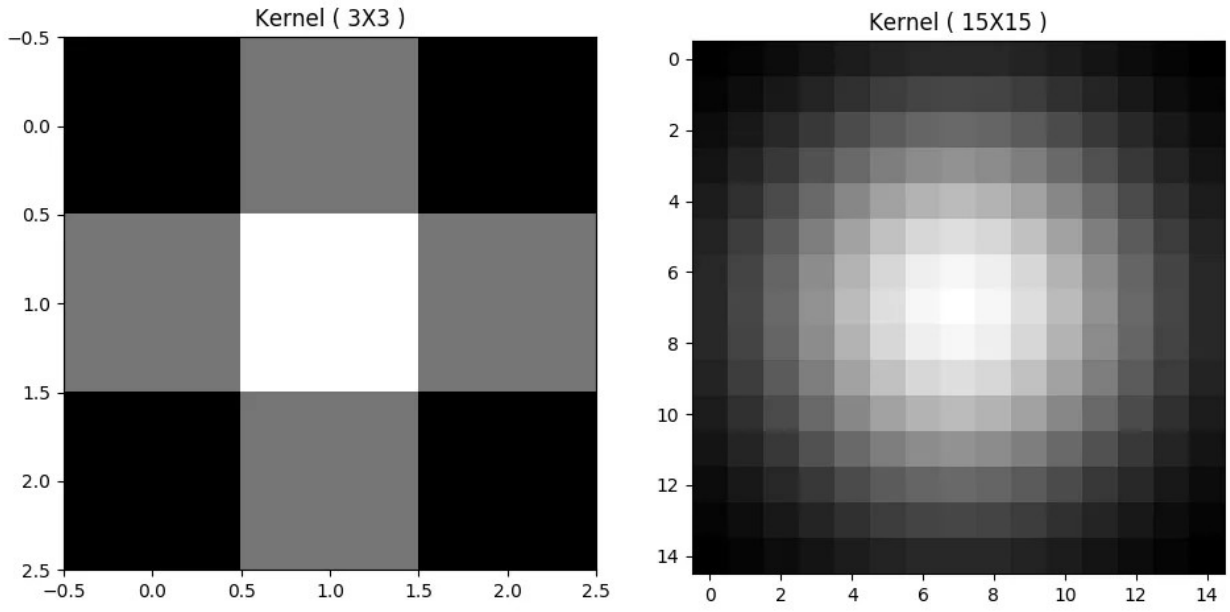
Tak więc orientacja w przestrzeni zachodzi następująco: przez nerw wzrokowy, impulsy trafiają do płata potylicznego, gdzie zachodzi przetwarzanie obrazu, a następnie przetworzone już informacje wzrokowe trafiają do płata ciemieniowego, gdzie wraz z informacjami z innych zmysłów takich jak słuch, dotyk, czucie głębokie, zmysł równowagi, pozwalają na ustalenie położenia w przestrzeni oraz do płata skroniowego, gdzie rozpoznawane jest, co znajduje się na obrazie. Są to dwa główne strumienie w korze wzrokowej. Rozpoznawanie kształtów odbywa się również tam, gdzie zostały one jedynie zasugerowane.

3. Detekcja krawędzi metodą Canny

Rozpoznawanie krawędzi metodą Canny odbywa się następującymi etapami:

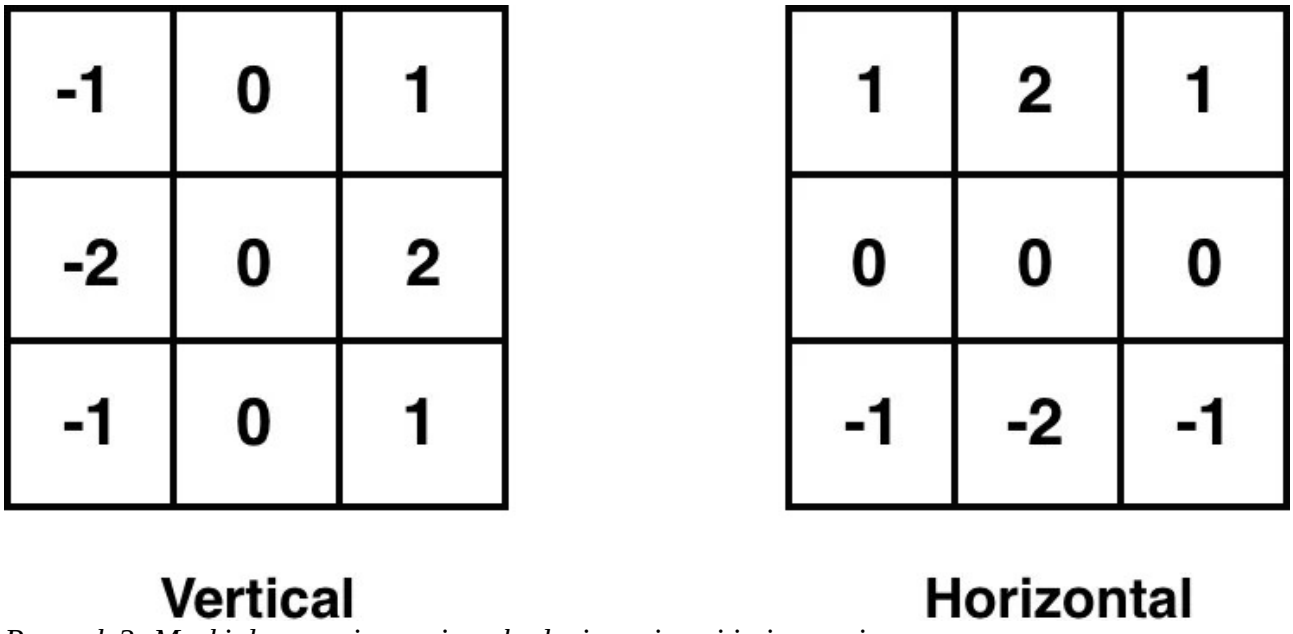
1. Wygładzenie obrazu metodą Gaussa
2. Znalezienie wielkości i kierunku gradientu
3. Supresja non-max
4. Użycie progu histerezy

Wygładzenie metodą Gaussa jest wykonywane w celu usunięcia szumu, ponieważ wykrywanie krawędzi jest wrażliwe na szum. Funkcja potrzebuje wielkości jądra oraz odchylenie standardowe. Można przyjąć odchylenie standardowe równe pierwiastkowi kwadratowemu wymiaru macierzy jądra. Tzn. na przykład dla jądra 4*4 będzie wynosić 2. Do obrazu dodawane są piksele po brzegach, następnie wykonywana jest operacja konwolucji, tzn. właściwego rozmycia, w pikselu umieszczana jest suma wartości okolicznych pikseli pomnożonych przez adekwatny współczynnik w jądrze przekształcenia. Konwolucja jest to nałożenie funkcji na siebie nawzajem, które produkuje funkcję wynikową, która reprezentuje jak jedna funkcja jest modyfikowana przez drugą.

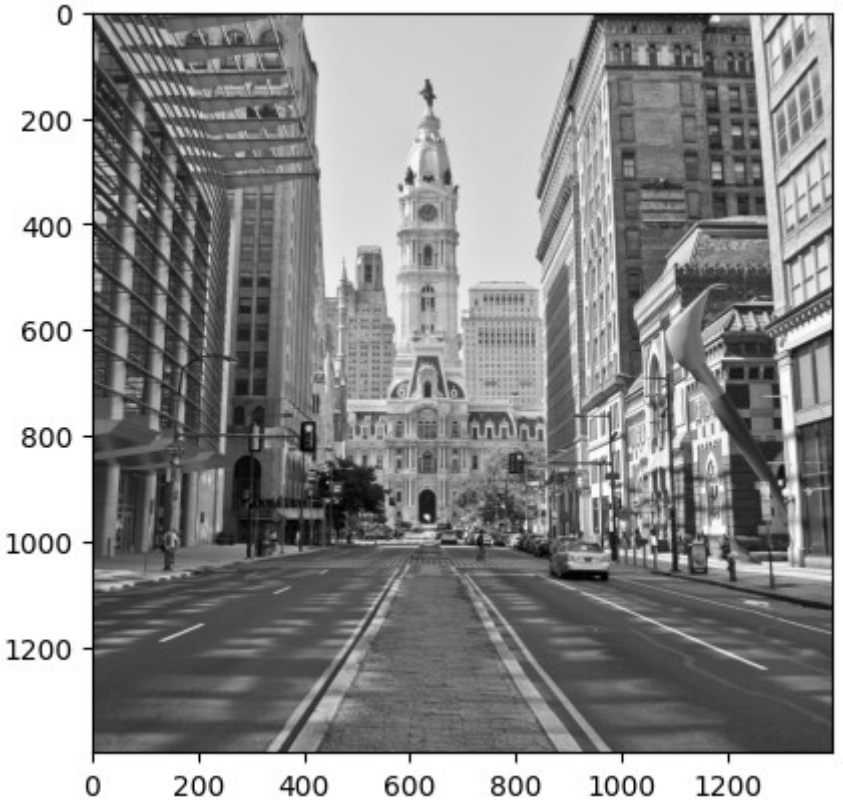


Rysunek 2: Jądro rozmycia Guassa, sigma równa pierwiastkowi kwadratowemu wymiaru macierzy

Następnie, po nałożeniu rozmycia, należy znaleźć gradient. Pierwszym etapem jest znalezienie jego wartości. Odbywa się to przez ponowne nałożenie na obraz filtra za pomocą konwolucji:



Rysunek 3: Maski dwuwymiarowej pochodnej: poziomej i pionowej

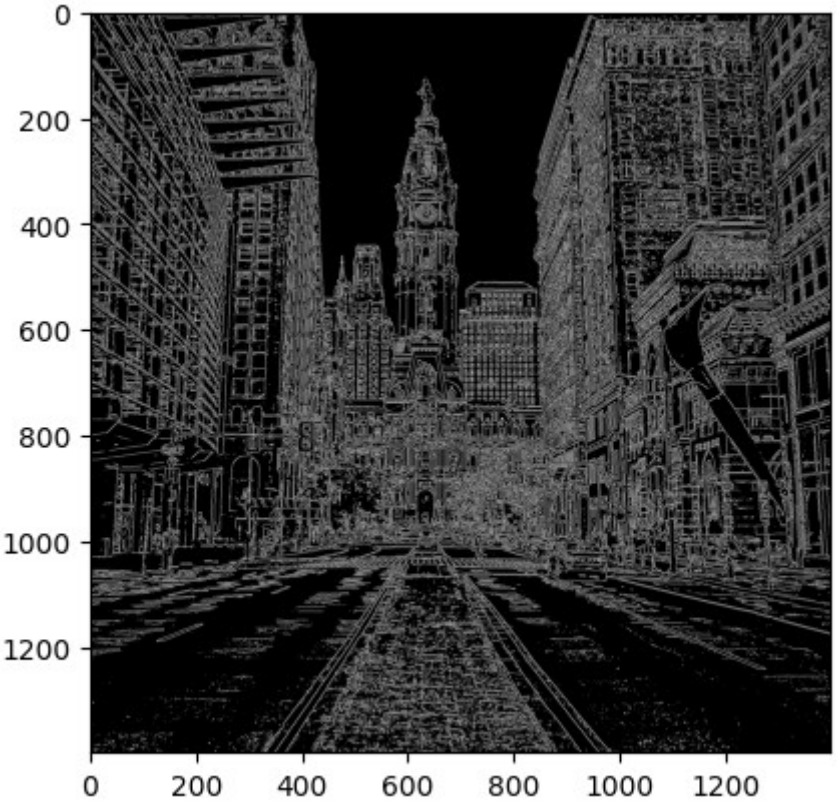


Rysunek 4: Obraz miasta przed detekcja krawędzi

Nałożone filtry są to maski pochodnej w kierunku poziomym i pionowym. Należy zastosować obie, a następnie zsumować je wektorowo – wynikowa wielkość będzie pierwiastkiem kwadratowym kwadratów gradientów w dwóch różnych prostopadłych orientacjach. Ta metoda nazywa się wykrywaniem krawędzi Sobela. Kierunek wyznacza się poprzez znalezienie funkcji *arctan* jednej z tych wielkości podzielonej przez drugą.

Supresja non-max ma na celu redukcję zbędnych i zduplikowanych krawędzi wykrytych przez detekcję krawędzi Sobela. Wykonuje ona porównania pomiędzy sąsiadującymi pikselami.

W funkcji histerezy, najpierw wykonywane jest przydzielenie pikseli do silnych i słabych punktów krawędzi za pomocą progów. Powyżej pewnego progu, piksel jest zaliczany do silnych, poniżej niego, ale powyżej drugiego, niższego progu, jest zaliczany do słabych, poniżej drugiego progu jego wartość jest ustalana na zero. Funkcja histerezy determinuje czy dany słaby piksel sąsiaduje z pikselem już wyznaczonym jako silny.



Rysunek 5: Detekcja krawędzi Canny wykonana na obrazie miasta

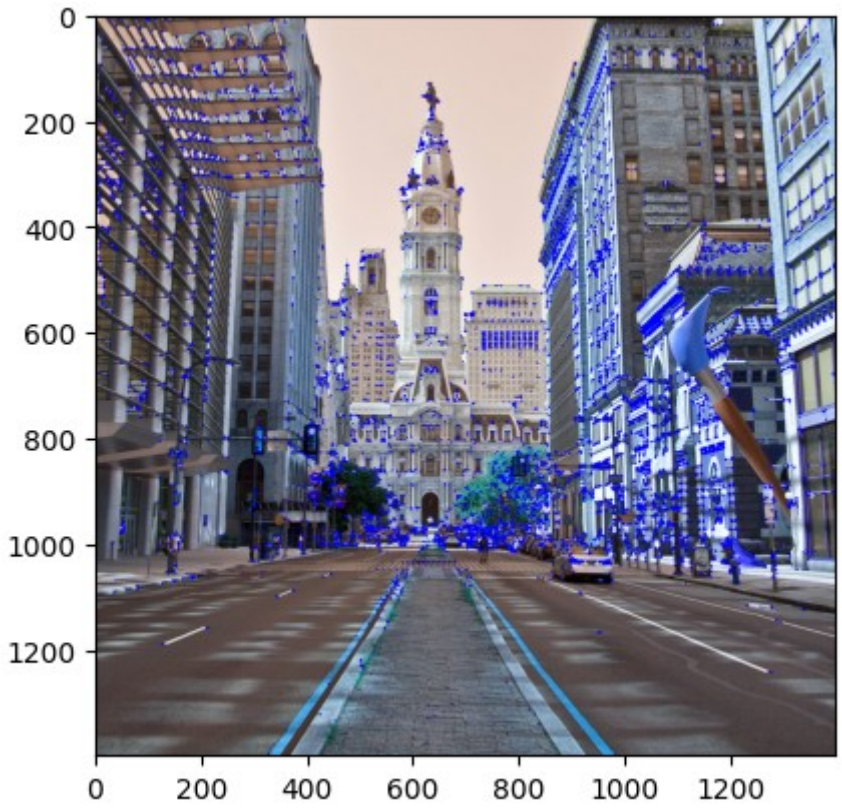
4. Inne operacje na obrazie

Następnym narzędziem, które może ułatwić rozpoznawanie obiektów oraz perspektywy na obrazie jest detekcja wierzchołków.

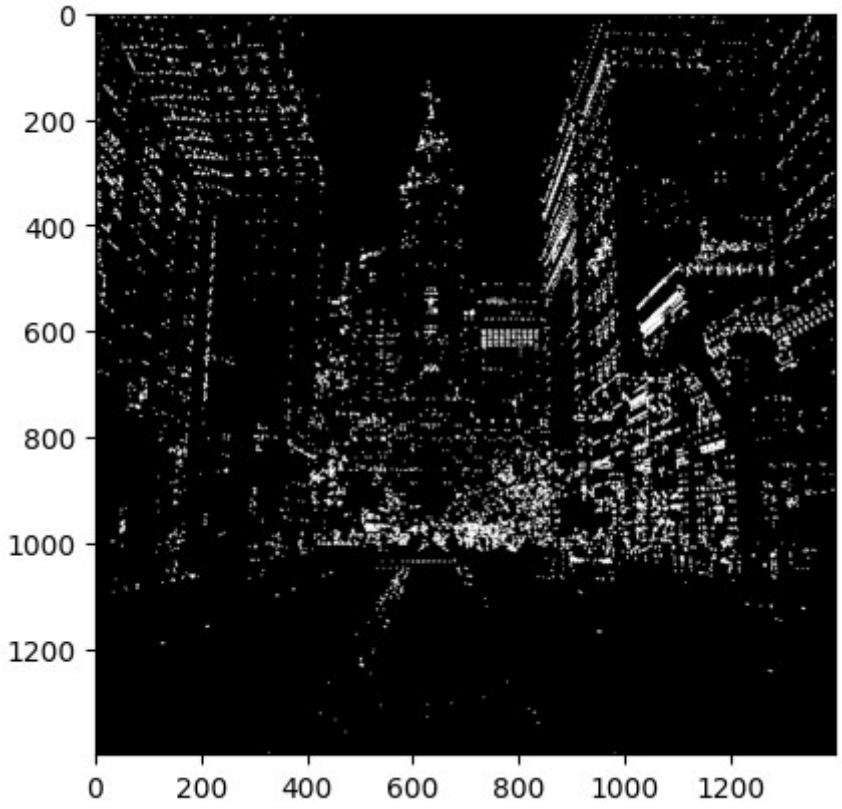
Detekcja wierzchołków metodą Harrisa pozwala na identyfikacje obszarów, w których jest duża zmienność gradientu intensywności (zarówno jego wielkości jak i kierunku).

Następnym krokiem w rozpoznawaniu lokalizacji budynków może być rozważenie granic kolorystycznych.

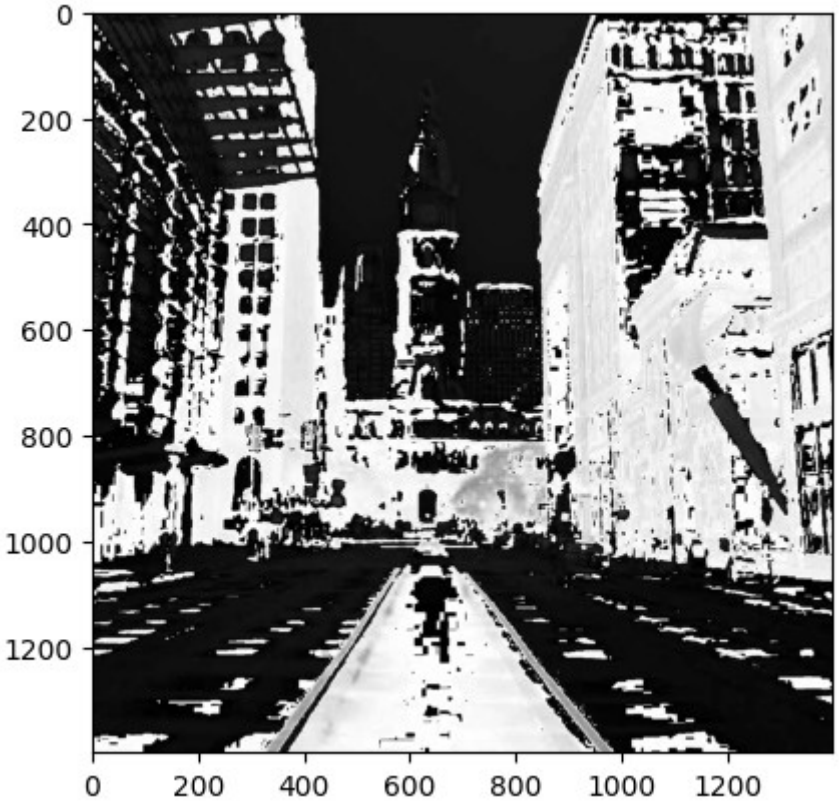
Celem tych wszystkich operacji ma być wytworzenie siatek budynków na podstawie wykrycia ich krawędzi i wierzchołków. Problem stanowi zdecydowanie, które wierzchołki i krawędzie stanowią krawędzie i wierzchołki budynku jako całości, a nie np. okien, rzucanych cieni. Nałożenie na siebie różnych danych o krawędziach, wierzchołkach, kolorach i granicach obszarów kolorystycznych może dostarczyć istotnych danych w celu wytworzenia siatki terenu.



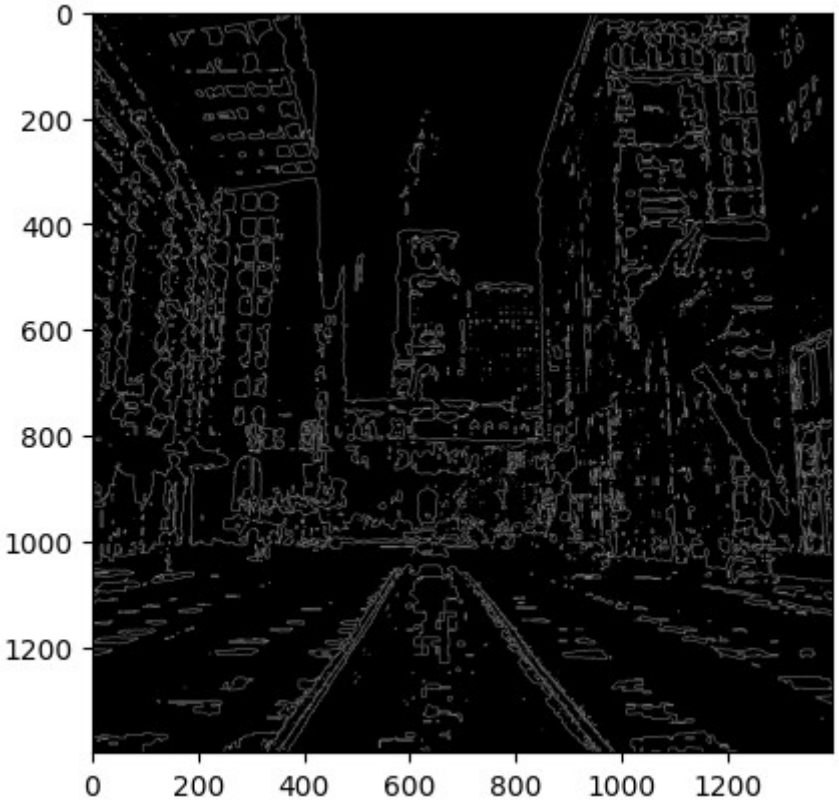
Rysunek 6: Obraz miasta z nałożonymi wykrytymi wierzchołkami



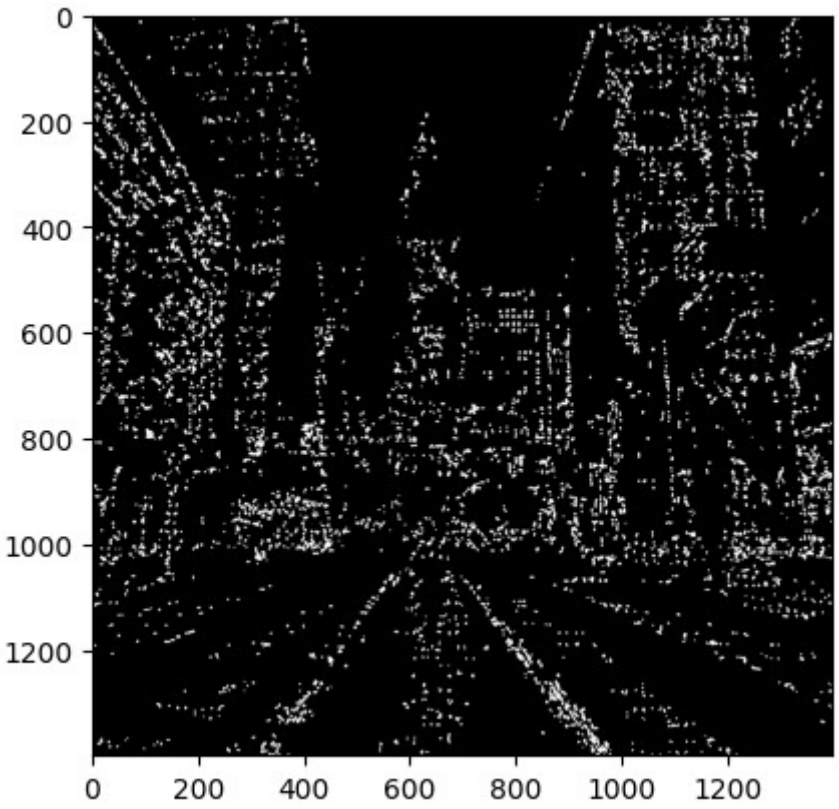
Rysunek 7: Wierzchołki nie nałożone na obraz



Rysunek 8: Różnica kanałów kolorów czerwonego i zielonego



Rysunek 9: Wykryte krawędzie na obrazie różnicy kanałów kolorów



Rysunek 10: Wykryte wierzchołki na obrazie różnicy kanałów kolorów

6. Podsumowanie

W celu orientacji przestrzennej np. wytworzeniu siatek obiektów znajdujących się na obrazie można zastosować metody obróbki obrazu takie jak detekcja krawędzi oraz wierzchołków. Należy w tym celu obliczyć gradienty – ich wartości oraz zwroty, które mówią o granicach obiektów na obrazie. Rozmycie obrazu filtrem Gaussa również wymaga zastosowania na obrazie konwolucji. W mózgu tym oraz wielu innym zadaniom związanym z przetwarzaniem obrazu – orientacji przestrzennej oraz rozpoznawaniu obiektów, poświęcone są całe płaty kory mózgowej: potyliczny, ciemieniowy, częściowo skroniowy.

Źródła

Andrzej Przybyszewski, *Neuroplastyczność a neurodegeneracja: Mózg a komputer*, wykłady.

Abhisek Jana, *Implement Canny edge detector using Python from scratch*, A Developer Diary. 20 maja 2019. Dostępny na stronie: https://www.adeveloperdiary.com/data-science/computer-vision/implement-canny-edge-detector-using-python-from-scratch/

Abhisek Jana, *Applying Gaussian Smoothing to an image using Python from scratch*, A Developer Diary. 19 maja 2019. Dostępny na stronie: https://www.adeveloperdiary.com/data-science/computer-vision/applying-gaussian-smoothing-to-an-image-using-python-from-scratch/

Abhisek Jana, *How to implement Sobel edge detection using Python from scratch*, A Developer Diary. 19 maja 2019. Dostępny na stronie: https://www.adeveloperdiary.com/data-science/computer-vision/how-to-implement-sobel-edge-detection-using-python-from-scratch/

Convolution, Wikipedia. Dostępne na stronie: https://en.wikipedia.org/wiki/Convolution

Python | Corner detection with Harris Corner Detection method using OpenCV, GeeksForGeeks, 4 stycznia 2023. Dostępne na stronie: https://www.geeksforgeeks.org/python-corner-detection-with-harris-corner-detection-method-using-opencv/

Źródła obrazów

Rysunek 1: Andrzej Przybyszewski, *Neuroplastyczność a neurodegeneracja: Mózg a komputer*, wykłady.

Rysunek 2: Abhisek Jana, *Applying Gaussian Smoothing to an image using Python from scratch*, A Developer Diary. 19 maja 2019. Dostępny na stronie: https://www.adeveloperdiary.com/data-science/computer-vision/applying-gaussian-smoothing-to-an-image-using-python-from-scratch/

Rysunek 3: Abhisek Jana, *How to implement Sobel edge detection using Python from scratch*, A Developer Diary. 19 maja 2019. Dostępny na stronie: https://www.adeveloperdiary.com/data-science/computer-vision/how-to-implement-sobel-edge-detection-using-python-from-scratch/

Rysunki 4-10: poddany obróbce obraz znajdujący się na stronie: https://www.vox.com/the-goods/2018/10/26/18025000/walkable-city-walk-score-economy

Załącznik: kod programu

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img_path=#path
img = cv2.imread(img_path)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(img_gray, cmap="gray")
plt.show()

img_edges = cv2.Canny(img_gray, threshold1=100, threshold2=100)
plt.imshow(img_edges, cmap="gray")
plt.show()

img_red_min_green = img[:, :, 0]-img[:, :, 1]
plt.imshow(img_red_min_green, cmap="gray")
plt.show()

corners = cv2.cornerHarris(np.float32(img_gray), 2, 5, 0.07)
corners = cv2.dilate(corners, None)
img_cp = img.copy()
img_cp[corners > 0.01 * corners.max()] = [0, 0, 255]
plt.imshow(img_cp)
plt.show()

corners[corners > 0.01 * corners.max()] = 255
corners[corners!=255] = 0
plt.imshow(corners, cmap="gray")
plt.show()

rg_contrast = img_red_min_green.copy()
rg_contrast[rg_contrast>150] = 255
rg_contrast[rg_contrast!=255] = 0
rg_edges = cv2.Canny(img_red_min_green, threshold1=100, threshold2=100)
plt.imshow(rg_edges, cmap="gray")
plt.show()

rg_corners = cv2.cornerHarris(np.float32(rg_contrast), 2, 5, 0.07)
rg_corners = cv2.dilate(rg_corners, None)
rg_corners[rg_corners > 0.01 * rg_corners.max()] = 255
rg_corners[rg_corners!=255] = 0
plt.imshow(rg_corners, cmap="gray")
plt.show()
```