Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Informatyki

DICOM i Segmentacja

Ćwiczenia laboratoryjne - Informatyka Medyczna

Paulina Dziwak
Piotr Ludynia

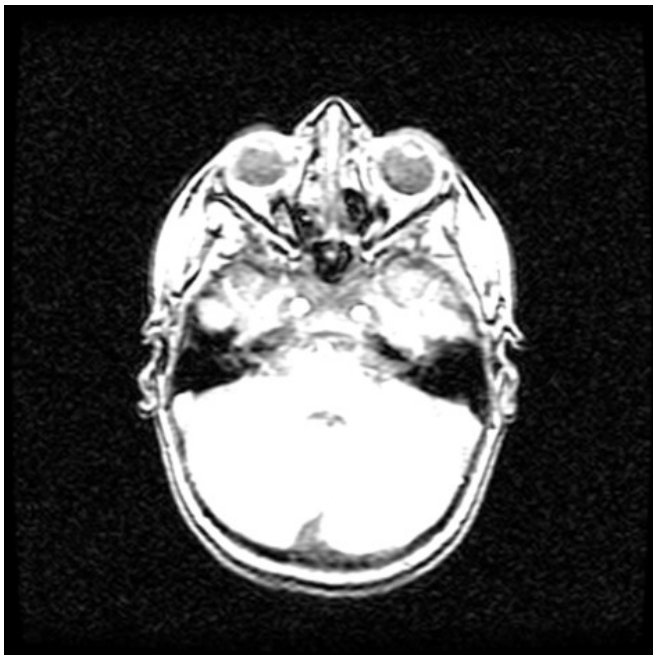
Kraków, 2024

Zadanie 1. a) DICOM:

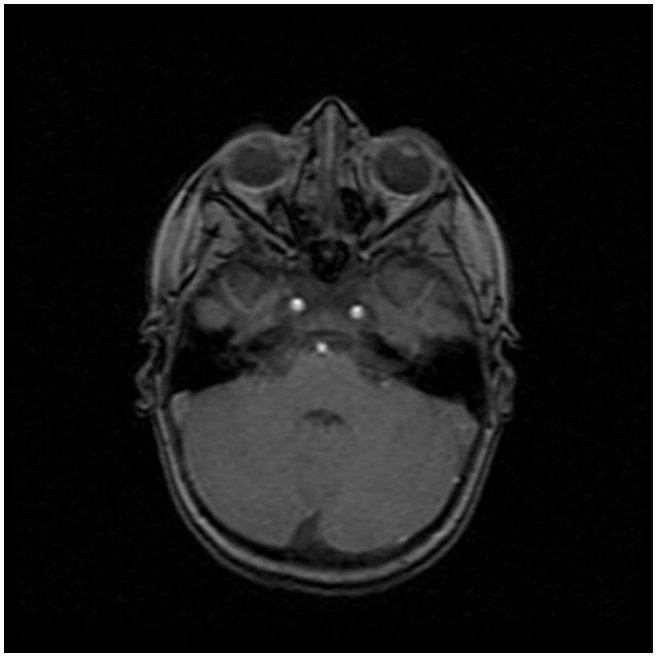
1) Wartości okna:
Width : 958
Center : 479

2) Wykonano transformację danych obrazowych zgodnie z wartościami okna.

Obraz przed transformacją:



Obraz po transformacji:



Zadanie 1. b)

Za uaktualnianie parametrów okna odpowiadają metody init_window oraz update_window. Dodatkowo, zaimplementowano metodę reset_window, która przywraca obraz do pierwotnego stanu po kliknięciu kółka myszy.

```
self.canvas.bind("<Button-1>", self.init_window)
self.canvas.bind("<B1-Motion>", self.update_window)
self.canvas.bind("<Button-2>", self.reset_window)

 def init_window(self, event):
    # todo: save mouse position
    self.start_x, self.start_y = event.x, event.y
    print("x: " + str(event.x) + " y: " + str(event.y))

def update_window(self, event):
    # todo: modify window width and center

    delta_x = event.x - self.start_x
    delta_y = event.y - self.start_y

    self.array2 = self.transform_data(
        self.data, self.winWidth + delta_x, self.winCenter + delta_y
    )
    self.image2 = Image.fromarray(self.array2)
    self.image2 = self.image2.resize((512, 512), Image.LANCZOS)
    self.img2 = ImageTk.PhotoImage(image=self.image2, master=root)
    self.canvas.itemconfig(self.image_on_canvas, image=self.img2)
```

```
    def reset_window(self, event):
        self.array = self.data
        self.image = Image.fromarray(self.array)

        # ANTIALIAS is deprecated. Use LANCZOS instead
        self.image = self.image.resize((512, 512), Image.LANCZOS)
        self.img = ImageTk.PhotoImage(image=self.image, master=root)
        self.image_on_canvas = self.canvas.create_image(0, 0, anchor=NW, image=self.img)
```

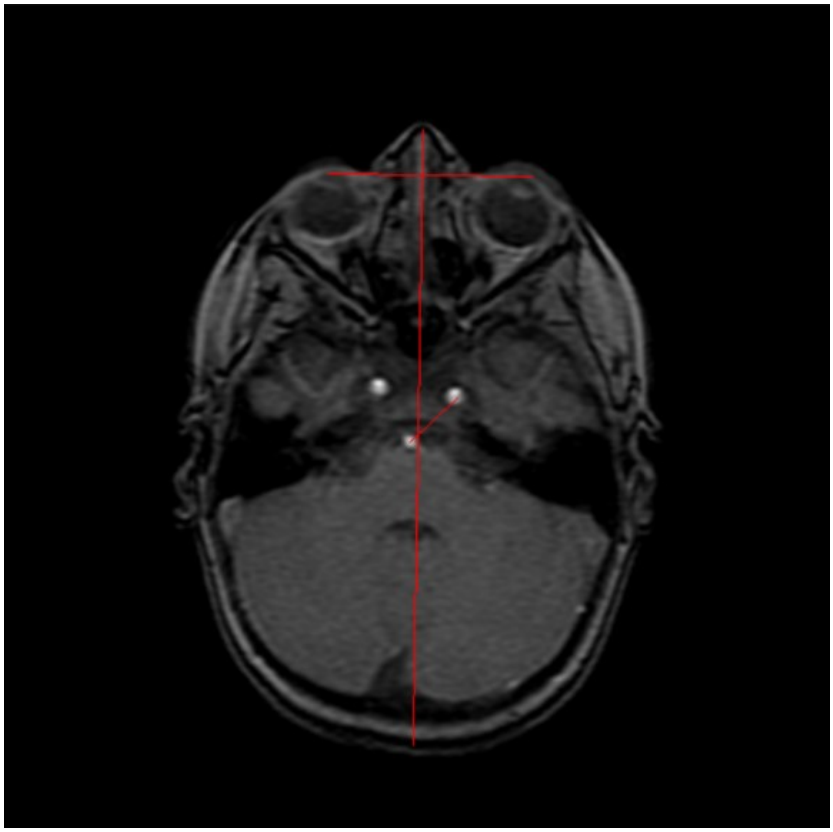
Zadanie 1.c)

Do programu dodano funkcjonalności pomiarowe. Umożliwiono użytkownikowi rysowanie odcinka na obrazie. Funkcjonalność ta została zaimplementowana w metodach init_measurement, update_measurement oraz finish_measurement.

```
        self.canvas.bind("<Button-3>", self.init_measurement)
        self.canvas.bind("<B3-Motion>", self.update_measurement)
        self.canvas.bind("<ButtonRelease-3>", self.finish_measurement)

        def init_measurement(self, event):
        # todo: save mouse position
        # todo: create line

        self.start_measurement_x, self.start_measurement_y = event.x, event.y
        self.measurement_line = self.canvas.create_line(
            self.start_measurement_x,
            self.start_measurement_y,
            event.x,
            event.y,
            fill="red",
        )

        def update_measurement(self, event):
            # todo: update line
            # hint: self.canvas.coords(...)
            self.canvas.coords(
                self.measurement_line,
                self.start_measurement_x,
                self.start_measurement_y,
                event.x,
                event.y,
            )

        def finish_measurement(self, event):
            # todo: print measured length in mm
```

```
print("x: " + str(event.x) + " y: " + str(event.y))
length_pixels = (
    ((event.x - self.start_measurement_x) / self.spacing_x) ** 2
    + ((event.y - self.start_measurement_y) / self.spacing_y) ** 2
) ** 0.5
print("Measured length: {:.2f} mm".format(length_pixels))
```

Jak można zauważyć, nasz program umożliwia precyzyjne zaznaczanie odcinków na obrazie oraz obliczanie ich długości, którą następnie wypisujemy na konsoli. Dzięki temu użytkownik ma możliwość szybkiego i dokładnego pomiaru odległości między punktami na obrazie.



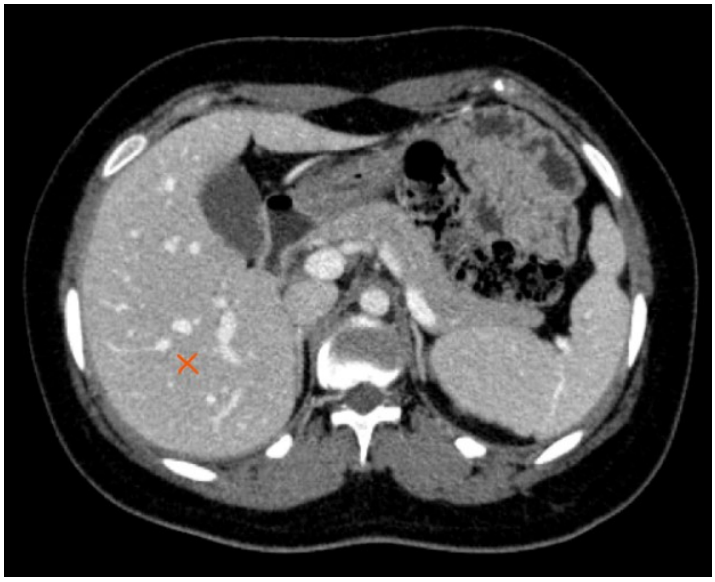
```
x: 201 y: 106
Measured length: 134.42 mm
x: 254 y: 458
Measured length: 404.32 mm
x: 251 y: 272
Measured length: 43.77 mm
```

Zadanie 2.
Stworzono funkcję segmentacji przez rozrastanie obszarów.

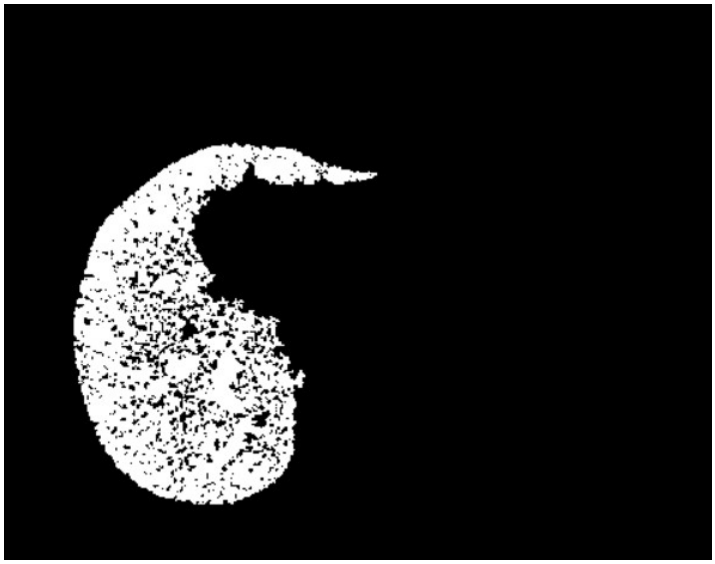
```
def region_growing(image, seed_point, threshold):
    # 0 - not visited pixels
    # 1 - visited pixels
    visited = np.zeros_like(image)
    # Pixel value at the starting point
    seed_value = image[seed_point[1], seed_point[0]]
    stack = [seed_point]

    while stack:
        x, y = stack.pop()

        # Checking whether the pixel has already been visited
        if visited[y, x] == 1:
            continue

        # Checking if the pixel value is similar to the starting point value
        if abs(image[y, x] - seed_value) < threshold:
            visited[y, x] = 1

            # Adding neighboring pixels to the stack
            if x > 0:
                stack.append((x - 1, y))
            if x < image.shape[1] - 1:
                stack.append((x + 1, y))
            if y > 0:
                stack.append((x, y - 1))
            if y < image.shape[0] - 1:
                stack.append((x, y + 1))

    # Marking a designated area in the resulting image
    segmented_image = np.zeros_like(image)
    segmented_image[visited == 1] = 255

    return segmented_image
```

Program wyświetla załadowany obraz, oczekując na kliknięcie myszką w wybrany punkt. Kliknięcie inicjuje wizualizację obszaru pikseli o zbliżonej intensywności do wybranego punktu, przy uwzględnieniu granic określonych przez wartość progową (threshold).

Załadowany obraz z zaznaczonym miejscem kliknięcia:



Wynik działania funkcji:



Wnioski:
Ćwiczenie pozwoliło nam zapoznać się ze standardem DICOM, który określa format i sposób transmisji danych obrazowych w medycynie. Ponadto poznaliśmy techniki segmentacji obrazów, które umożliwiają precyzyjne wyodrębnienie i analizę konkretnych obszarów na obrazach medycznych.