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

Wizualizacja

Ćwiczenia laboratoryjne - Informatyka Medyczna

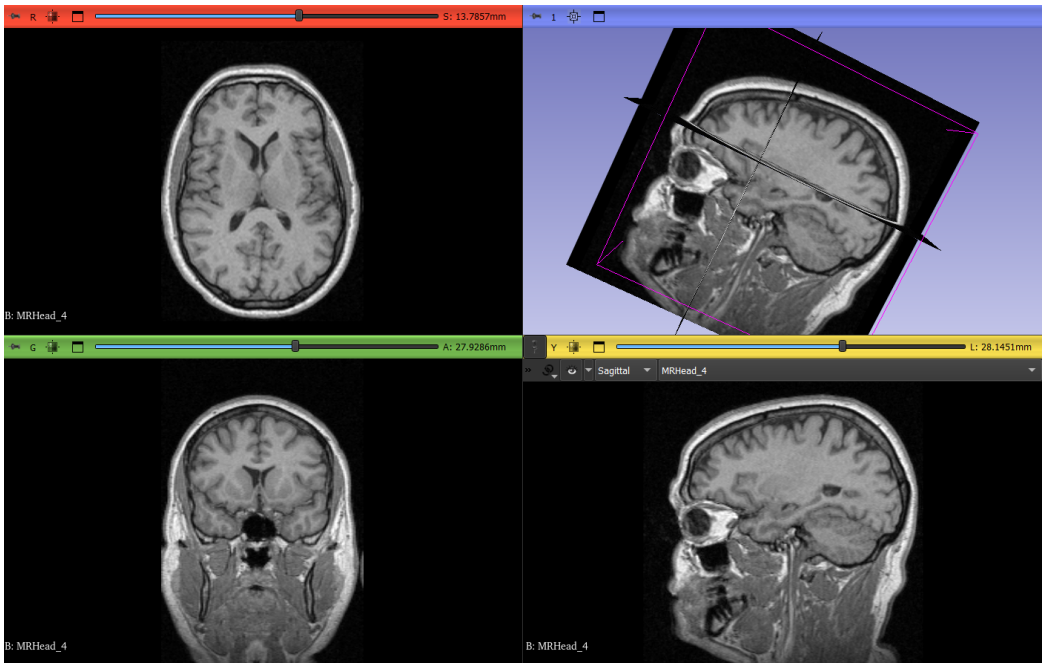
Paulina Dziwak
Piotr Ludynia

Kraków, 2024

Zadanie 1. Slicer 3D

Zadanie pierwsze polegało za zapoznaniu się z platformą do przetwarzania obrazu medycznego Slicer 3D. Jest to potężne oprogramowanie obsługujące wszystkie powszechnie używane zbiory danych, takie jak obrazy, segmentacje, adnotacje, transformacje itp. w 2D, 3D i 4D. Slicer udostępnia ponad 100 wbudowanych modułów, z których każdy implementuje określony zestaw funkcji do tworzenia lub manipulowania danymi. Poprzez generowanie realistycznych przestrzennych wizualizacji użytkownicy mogą oglądać, analizować i manipulować obiektami, co jest niezwykle przydatne w diagnostyce medycznej oraz w badaniach naukowych.

Po zainstalowaniu oprogramowania uruchomiono przykładowy zestaw danych, aby przetestować działanie platformy.



Slicer 3D to wyjątkowe narzędzie, które cechuje się nie tylko wszechstronnością, ale także bogactwem możliwości. Na powyższym obrazie widać, jak Slicer 3D umożliwia wyświetlanie danych w formie trójwymiarowej oraz jednoczesną analizę poprzez przekroje 2D. Trzy domyślne suwaki, wyróżnione kolorami czerwonym, żółtym i zielonym, stanowią wygodne narzędzie do kontroli i wyświetlania poszczególnych przekrojów obrazów.

Zadanie 2. Wizualizacja za pomocą izopowierzchni

W ramach tego zadania przedstawimy proces wizualizacji obrazów medycznych za pomocą izopowierzchni oraz implementację interaktywnego suwaka umożliwiającego zmianę wartości iso wizualizacji.

Poniżej przedstawiono kluczowe elementy implementacji:
Generowanie izopowierzchni wykonano za pomocą filtru vtkContourFilter. Wartość iso ustawiono na podstawie wybranej wartości piksela.

```
iso_value = 100
contourFilter = vtk.vtkContourFilter()
contourFilter.SetInputConnection(reader.GetOutputPort())
contourFilter.SetValue(0, iso_value)
contourFilter.Update()
```

Zdefiniowano funkcję transferu kolorów.

```
color_transfer_function = vtk.vtkColorTransferFunction()
color_transfer_function.AddRGBPoint(0, 1.0, 1.0, 1.0)  # White
color_transfer_function.AddRGBPoint(255, 1.0, 0.5, 0.0)  # Orange
```

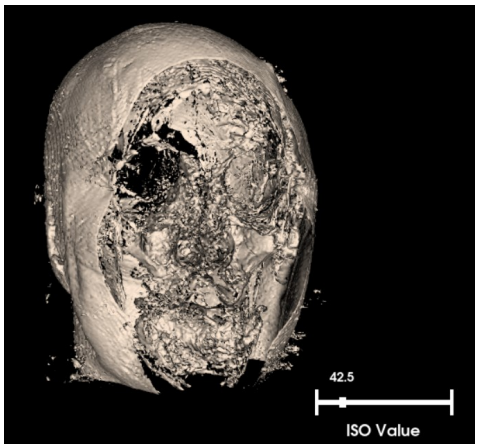
Stworzono aktora i renderera, co jest niezbędne do wyświetlenia wizualizacji w zadanym oknie.

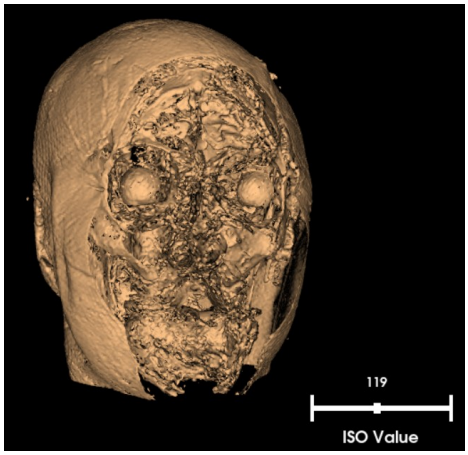
```
# vtkActor
actor = vtk.vtkActor()
actor.SetMapper(mapper)

# Renderer
renderer = vtk.vtkRenderer()
renderer.AddActor(actor)

# Window
renWin = vtk.vtkRenderWindow()
renWin.AddRenderer(renderer)
renWin.SetSize(800, 600)
```

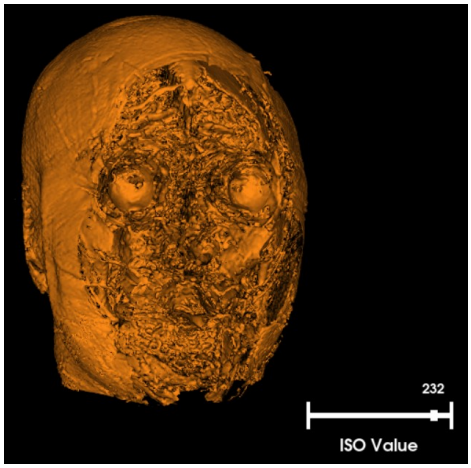
Następnie dodano interaktywny suwak do zmiany wartości iso. Suwak ten jest powiązany z funkcją update_iso_value, która aktualizuje wartość iso w filtrze izopowierzchni po interakcji użytkownika.

```
# Slider to change iso value
iso_slider_rep = vtk.vtkSliderRepresentation2D()
iso_slider_rep.GetPoint1Coordinate().SetCoordinateSystemToNormalizedDisplay()
iso_slider_rep.GetPoint1Coordinate().SetValue(.7, .1)
iso_slider_rep.GetPoint2Coordinate().SetCoordinateSystemToNormalizedDisplay()
iso_slider_rep.GetPoint2Coordinate().SetValue(.9, .1)
iso_slider_rep.SetMinimumValue(0)
iso_slider_rep.SetMaximumValue(255)
iso_slider_rep.SetValue(iso_value)
iso_slider_rep.SetTitleText("ISO Value")

def update_iso_value(value, contourFilter):
    contourFilter.SetValue(0, value)
    renWin.Render()
```

Przykładowy wynik działania programu:







Dzięki możliwości dynamicznej kontroli nad wartością iso użytkownik może dostosować wizualizację do swoich indywidualnych potrzeb diagnostycznych.

Zadanie 3. Wizualizacja objętości z suwakiem funkcji transferu

Zadanie polegało na zaimplementowaniu wizualizacji objętościowego obrazowania medycznego za pomocą bezpośredniego renderowania. Dodatkowo, dodano suwak umożliwiający dynamiczną zmianę punktów w funkcji transferu, kontrolującej kolor lub przezroczystość wyświetlanych struktur.

Do realizacji zadania wykorzystano bibliotekę VTK (Visualization Toolkit), służącą do wizualizacji danych 3D. Poniżej przedstawiono kluczowe elementy implementacji:

Stworzono obiekt vtkSmartVolumeMapper, który jest odpowiedzialny za mapowanie danych objętościowych na scenę 3D.

```
volume_mapper = vtk.vtkSmartVolumeMapper()
volume_mapper.SetInputConnection(reader.GetOutputPort())
```

Zdefiniowano funkcje transferu kolorów oraz przeźroczystości.

```
# --- Color Transfer Function ---
color_transfer_function = vtk.vtkColorTransferFunction()
color_transfer_function.AddRGBPoint(0, 1.0, 1.0, 1.0)  # White
color_transfer_function.AddRGBPoint(255, 0.0, 0.0, 1.0)  # Blue

# --- Opacity Transfer Function ---
opacity_transfer_function = vtk.vtkPiecewiseFunction()
```

Następnie stworzono aktora objętościowego, którego dodano do rendera na którym będą wyświetlane obiekty w 3D, natomiast render umieszczono w oknie renderowania.

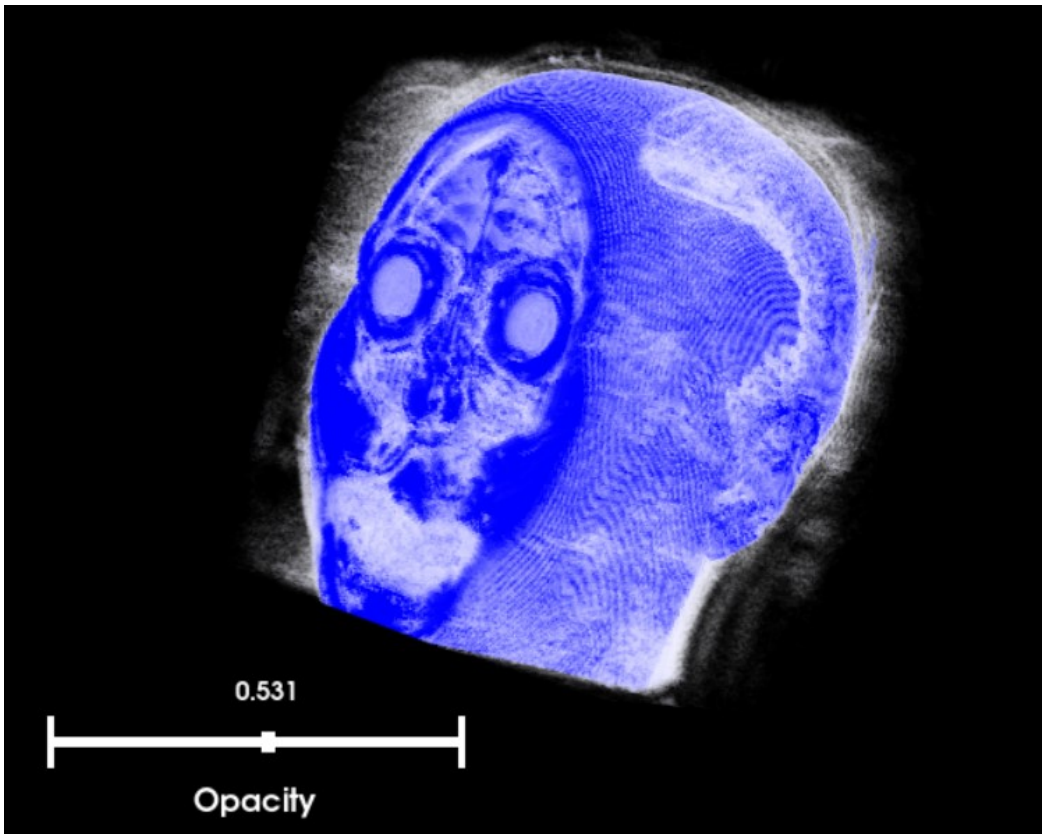
```
# --- Volume Actor ---
volume_actor = vtk.vtkVolume()
volume_actor.SetMapper(volume_mapper)
volume_actor.SetProperty(volume_property)

# --- Renderer ---
renderer = vtk.vtkRenderer()
renderer.SetBackground(0.0, 0.0, 0.0)
renderer.AddVolume(volume_actor)

# --- Window ---
render_window = vtk.vtkRenderWindow()
render_window.SetWindowName("Volume Rendering")
render_window.SetSize(800, 600)
render_window.AddRenderer(renderer)
```

Ponadto zaimplementowano suwak umożliwiający zmianę punktu w funkcji transferu.

```
tf_slider_rep = vtk.vtkSliderRepresentation2D()
tf_slider_rep.GetPoint1Coordinate().SetCoordinateSystemToNormalizedDisplay()
tf_slider_rep.GetPoint1Coordinate().SetValue(0.1, 0.1)
tf_slider_rep.GetPoint2Coordinate().SetCoordinateSystemToNormalizedDisplay()
tf_slider_rep.GetPoint2Coordinate().SetValue(0.4, 0.1)
tf_slider_rep.SetMinimumValue(0.0)
tf_slider_rep.SetMaximumValue(1.0)
tf_slider_rep.SetValue(0.5)
tf_slider_rep.SetTitleText("Opacity")

tf_slider_widget = vtk.vtkSliderWidget()
tf_slider_widget.SetInteractor(interactor)
tf_slider_widget.SetRepresentation(tf_slider_rep)
tf_slider_widget.SetAnimationModeToAnimate()
tf_slider_widget.EnabledOn()
tf_slider_widget.AddObserver(
```

```
    "InteractionEvent",
    lambda obj, event: opacity_transfer_function.AddPoint(
        127, obj.GetRepresentation().GetValue()
    ),
)
```

Przykładowy wynik działania programu:



Wnioski:
Zadania dostarczyły nam praktycznej wiedzy na temat procesu wizualizacji danych medycznych oraz pokazały, jak zaawansowane techniki wizualizacyjne mogą mieć pozytywny wpływ na diagnostykę i leczenie pacjentów.