Uniwersytet Zielonogórski
Instytut Sterowania i Systemów Informatycznych

Rozpoznawanie Obrazów

Semestr II, Informatyka

mgr inż. Marcin Skobel

2022

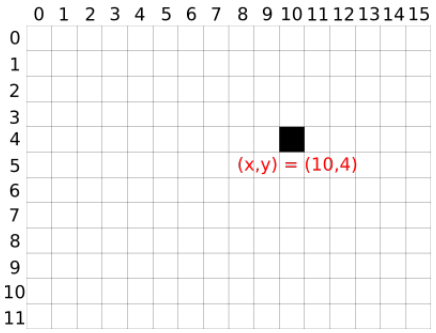
Laboratorium nr 1: Programy i pakiety do rozpoznawania obrazów oraz kadrowanie i operacje afiniczne.

I. Zagadnienia teoretyczne

Wstęp

Rozpoznawanie obrazów stanowi jedno z kluczowych zagadnień informatyki związanych z przetwarzaniem obrazów. Z tego powodu większość języków programowania dobrze nadaje się jako narzędzie do rozpoznawania obrazów. Można łatwo wymienić kilka środowisk, które szczególnie dobrze radzą sobie z zadaniem rozpoznawania obrazów np.: Matlab, Java, Python, C++, R. Wybór odpowiedniego języka programowania nie jest zatem sprawą oczywistą, ale ze względu na rosnącą popularność wśród programistów oraz rozbudowaną liczbę pakietów do rozpoznawania i przetwarzania obrazów najlepszym wyborem wydaje się być język Python.

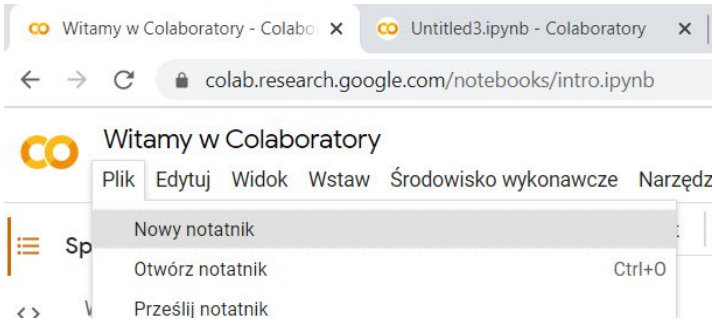
Obraz cyfrowy stanowi prostokątny zbiór pikseli. Położenie poszczególnych pikseli w przestrzeni obrazu może być definiowane przy zastosowaniu współrzędnych x i y. Innymi słowy obraz składa się z pikseli ułożonych w kolumny oraz wiersze, które tworzą układ przypominający macierz. Stosując układ współrzędnych możemy jednoznacznie określić położenie poszczególnych pikseli:



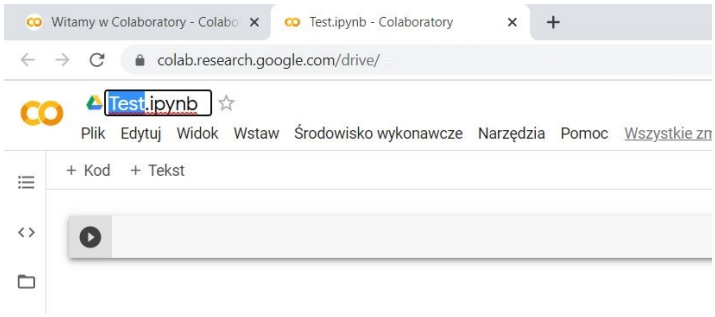
Początek układu współrzędnych obrazu cyfrowego najczęściej definiowany jest w lewym górnym rogu. Współrzędne początkowe obrazka przyjmują wartość (0,0) oznacza to, że kolumny i wiersze są numerowane od liczby 0. Biblioteka, która będzie przydatna w wykonaniu zadania nosi nazwę Pillow (`https://pillow.readthedocs.io/en/stable/`).

Colaboratory - Google Colab

Colab stanowi narzędzie pracy z językiem Python. Do uruchomienia wystarczy aktywne konto Google oraz przeglądarka internetowa. W celu uruchomienia narzędzia Colab wystarczy skorzystać z tego linku: https://colab.research.google.com/. Po wejściu na stronę można utworzyć nowy notatnik:

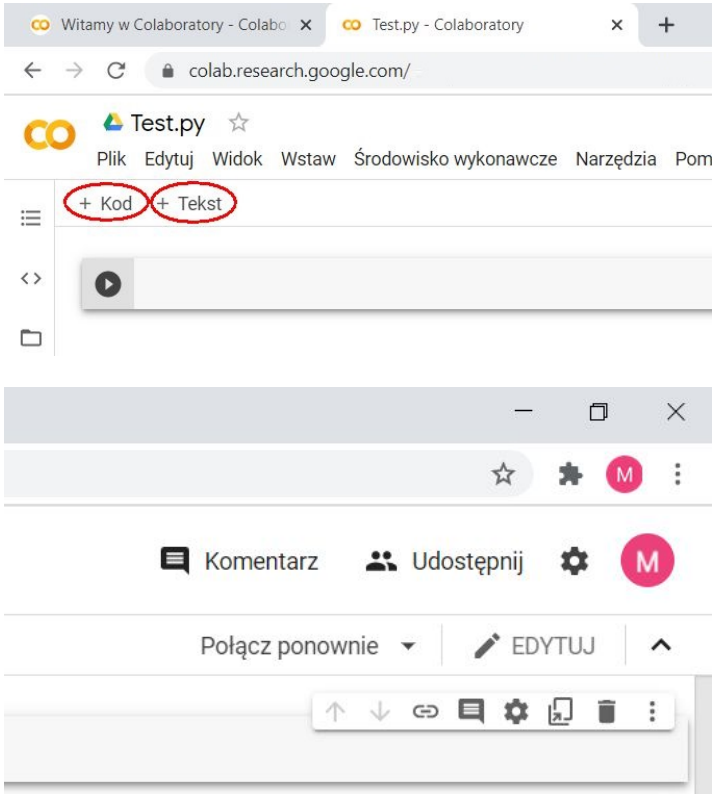


Utworzenie nowego notatnika rozpoczyna pracę z językiem Python, większość narzędzi do przetwarzania obrazów jest już zainstalowana. W przypadku gdy chcemy użyć jakiejś biblioteki której jeszcze nie ma w Colabie wystarczy użyć systemu pip. Po instalacji Colab zapamięta wszystkie zainstalowane pakiety. Po uruchomieniu notatnika można zmodyfikować nazwę pliku:



Notatnik Colaba domyślnie pracuje na plikach z rozszerzeniem *.ipynb jednak można również pracować na standardowych plikach .*py. Na pasku menu można odnaleźć standardowe etykiety środowiska programistycznego takie jak: Plik, Edytuj, Widok, Wstaw, Środowisko wykonawcze, Narzędzia i Pomoc. Poniżej paska menu środowisko Colab posiada dwa ciekawe przyciski (+ Kod i + Text):

Szczególnie interesujący może być przycisk "+ Kod", którego zadaniem jest utworzenie nowej komórki kodu. Każdą kolejną komórkę kodu możemy wywoływać oddzielnie lub korzystając z "Menu → Środowisko wykonawcze → Uruchom wszystko"jak nazwa wskazuje uruchomić cały kod zawarty w pliku (notatniku). Po drugiej stronie ekranu w prawym górnym rogu umieszczono dalszą część paska menu. Zaczynając od góry możemy tu znaleźć narzędzie do komentarzy, udostępniania oraz ustawienia, w których można na przykład zmienić motyw na ciemny.



Poniżej mamy narzędzie do modyfikacji połączeń z serwerem wykonawczym, narzędzie edycji, oraz narzędzie do chowania menu. Poniżej natomiast mamy jeszcze jedno małe menu służące do zarządzania aktywną komórką kodu. Ponadto w komórce kodu prawym przyciskiem myszy można wywołać menu kontekstowe.

Kadrowanie obrazu

Pierwszym zagadnieniem poruszanym w bieżącym laboratorium jest kadrowanie fragmentu obrazu cyfrowego. Kadrowanie polega na wycięciu fragmentu obrazu pierwotnego. Do poprawnego wykonania zadania kadrowania niezbędne jest posiadania kilku podstawowych informacji. Informacja pierwsza dotyczy rozmiaru obrazka wejściowego czyli liczby kolumn i wierszy w całym obrazku. Druga informacja to współrzędne początku kadru, natomiast trzecia informacja to wysokość i szerokość kadru.

Przeskalowanie

Zmiana skali obrazu cyfrowego jest dość powszechnie używana. Redukcja rozdzielczości może znaleźć zastosowanie np.: w redukcji wielkości pliku obrazu do wymagań portali i stron internetowych. Z teoretycznego punktu widzenia przeskalowanie definiujemy przy użyciu wzoru:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} u & 0 & 0 \\ 0 & v & 0 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \tag{1}$$

gdzie u odnosi się do skali rozciągnięcia w kierunku poziomym, natomiast v w kierunku pionowym. Można zatem zauważyć, że przeskalowanie w obu kierunkach nie musi być jednakowe.

W zależności od potrzeby można obraz rozciągnąć (spłaszczyć) inaczej w pionie, a inaczej w poziomie.

Translacja

Kolejnym przekształceniem afinicznym jakie należy poznać jest translacja (przesunięcie) obrazu. Przekształcenie to polega na jednakowym przesunięciu wszystkich pikseli na obrazie w ustalonym kierunku. Do przesunięcia niezbędny będzie zatem wektor przesunięcia. Translacja obrazu może zostać przedstawiona przy użyciu wzoru:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & m \\ 0 & 0 & n \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \tag{2}$$

gdzie m to wartość przesunięcia w kierunku poziomym, natomiast n to wartość przesunięcia w kierunku pionowym.

Obrót

Obrót (rotacja) obrazu stanowi kolejny rodzaj podstawowej geometrycznej transformacji obrazu. Obracanie obrazu wykonywane jest względem środka obrazu. Transformacja obrazu przy użyciu rotacji wymaga zdefiniowania macierzy obrotu:

$$R(\theta) = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \tag{3}$$

Po zdefiniowaniu macierzy obrotu możemy jej użyć do rotacji wektorów kolumnowych:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4}$$

Z tego względu poszczególne współrzędne (x', y') można zapisać przy użyciu wzorów:

$$\begin{cases} x' = x\ cos\theta & - & y\ sin\theta, \\ y' = x\ sin\theta & + & y\ cos\theta \end{cases} \tag{5}$$

II. Przykład praktyczny

Rozpoznawanie obrazów wiąże się z przetwarzaniem plików, w związku z tym konieczna będzie praca z Dyskiem Google: https://drive.google.com/drive/my-drive. Będąc już w katalogu głównym dysku warto założyć folder roboczy w którym będą przechowywane obrazy potrzebne do realizacji zajęć laboratoryjnych. Jeśli ktoś już uruchomił Colab, w katalogu głównym Dysku Google Powinien się pojawić folder Colab Notebooks. Katalog roboczy z obrazkami można utworzyć bezpośrednio w katalogu głównym lub wewnątrz katalogu Colab Notebooks. Nazwa folderu roboczego może być oczywiście dowolna np.: Images.

Aby korzystać z zasobów Dysku Google w aplikacji Colab należy zamontować Dysk w programie poleceniem:

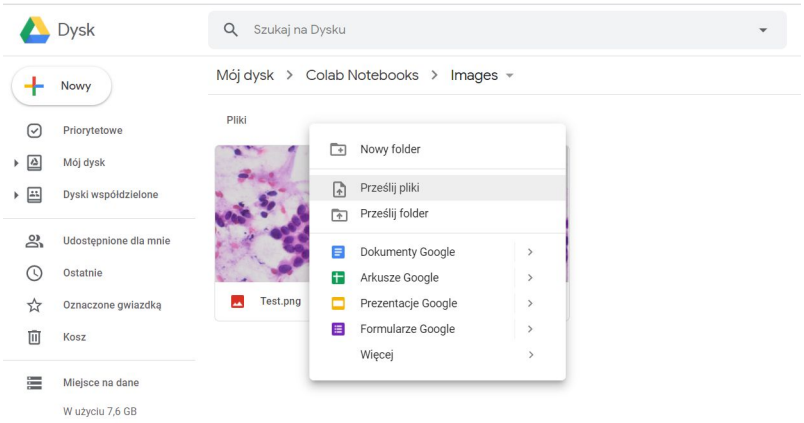
```
from google.colab import drive
drive.mount('/content/drive')
```

Po uruchomieniu powyższego polecenia system Colab wygeneruje bezpieczny odnośnik do kodu uwierzytelniającego, którego treść należy skopiować do powstałego okienka. Po prawidłowym zamontowaniu pojawi się komunikat:
Mounted at /content/drive
Następnie w tym samym oknie kodu wpisujemy kolejne dwie linijki:

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

Polecenie importują zbiór funkcji pyplot służący do wyświetlania obrazów oraz moduł image służący do wykonywania podstawowych operacji wczytywania skalowania i wyświetlania obrazu. W tym momencie można kliknąć na przycisk wykonania kodu a następnie kliknąć w przycisk "+ Kod", aby utworzyć nowe okno kodu.

Obecnie katalog Images jest pusty. Pierwszym zadaniem jakie należy wykonać jest wrzucenie dowolnego obrazka testowego do katalogu z obrazami:



Można też pobrać plik testowy z tego odnośnika: http://staff.uz.zgora.pl/mskobel/Test.tif

Następnie wracamy do zakładki z Colabem i wpisujemy w nowym oknie kod:

```
image_path = "drive/My Drive/Colab Notebooks/Images/"
img = mpimg.imread(image_path+'Test.tif')
imgplot = plt.imshow(img)
plt.show()
```

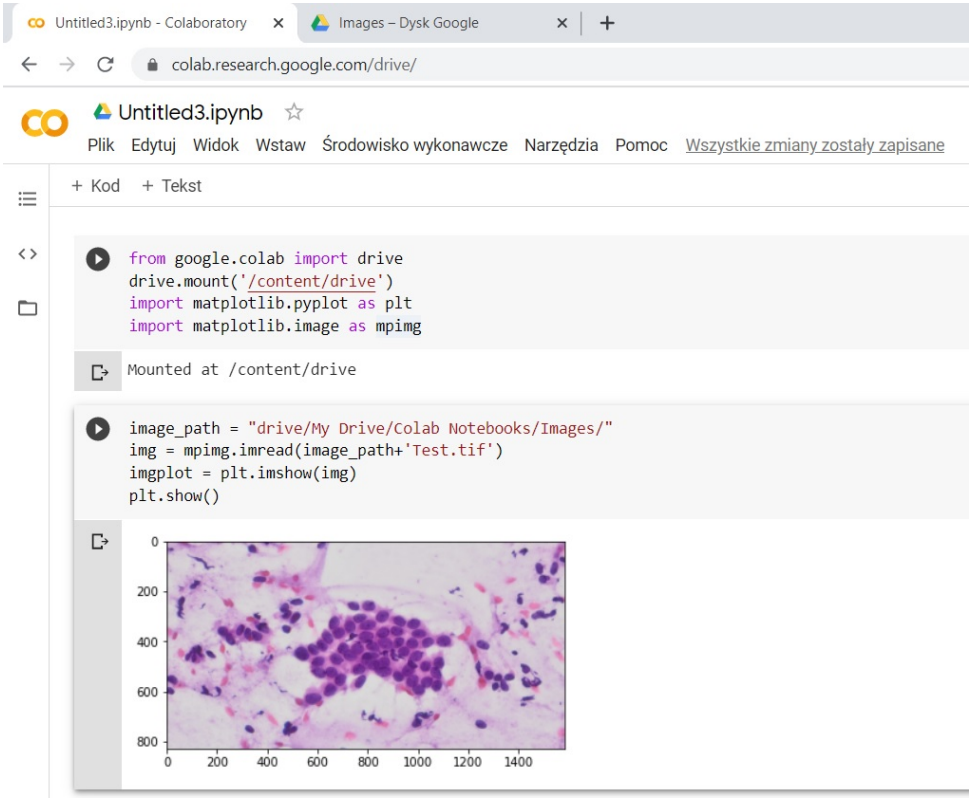
Przedstawiony kod ma za zadanie wyczytać obrazek do zmiennej img a następnie wyświetlić go w oknie przeglądarki:

Przykład praktyczny - kadrowanie i operacje afiniczne

Wstępna konfiguracja

Drugi przykład praktyczny tym razem rozpoczynamy od zaimportowania biblioteki Pillow i matplotlib.pyplot do Colaba oraz zamontowaniu Dysku Google:

```
import PIL
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount('/content/drive')
```



W kolejnym oknie kodu można dokonać podstawowej konfiguracji obrazu testowego:

```
testImagePath = 'drive/My Drive/Colab Notebooks/Images/'
testImage = PIL.Image.open(testImagePath+'Test.tif')
width, height = testImage.size
print(width, height)
imgplot = plt.imshow(testImage)
```

Pierwsza zmienna (*testImagePath*) jest ścieżką do folderu zawierającego obrazy. Zmienna *testImage* przechowuje obraz zaimportowany przy użyciu biblioteki *PIL.Image*. Możemy bardzo łatwo uzyskać zmienne dotyczące szerokości i wysokości obrazu (*width, height*) wpisując polecenie *testImage.size*. Wymiary obrazka można w tym momencie wyświetlić funkcją *print* a następnie wyświetlić obraz przy użyciu biblioteki *matplotlib.pyplot*.

Kadrowanie

Obraz testowy został wczytany przy użyciu biblioteki *PIL.Image* zatem możemy od razu rozpocząć zadanie kadrowania. W tym celu należy zdefiniować współrzędne okna kadrowania (x1, y1, x2, y2). Współrzędne x1 oraz y1 oznaczają lewy górny róg okna kadrowania natomiast współrzędne x2 oraz y2 oznaczają prawy dolny róg tego okna. Oczywiście uzyskany obraz wynikowy można wyświetlić przy użyciu biblioteki *matplotlib.pyplot*:

```
cropCoordinates = (500, 150, 1012, 662)
croppedImage = testImage.crop(cropCoordinates)
imgplot = plt.imshow(croppedImage)
```

Przeskalowanie

W kolejnym oknie kodu warto przetestować działanie funkcji przeskalowania:

```
resizedImage = testImage.resize((1800,500))
imgplot = plt.imshow(resizedImage)
```

W tym przykładzie mamy do czynienia z dwiema wartościami. Pierwsza z nich (1800) oznacza wynikową szerokość obrazu po przeskalowaniu natomiast druga wartość (500) oznacza wysokość obrazu po przeskalowaniu.

Translacja

Nieco bardziej skomplikowanym zagadnieniem z punktu widzenia kodu Python jest translacja obrazu. Przed wykonaniem translacji należy wprowadzić 6 zmiennych pomocniczych (a, b, c, d, e, f)

```
a = 1
b = 0
c = -300
d = 0
e = 1
f = -450
translatedImage = testImage.transform(testImage.size, PIL.Image.AFFINE, (a, b, c, d, e, f))
imgplot = plt.imshow(translatedImage)
```

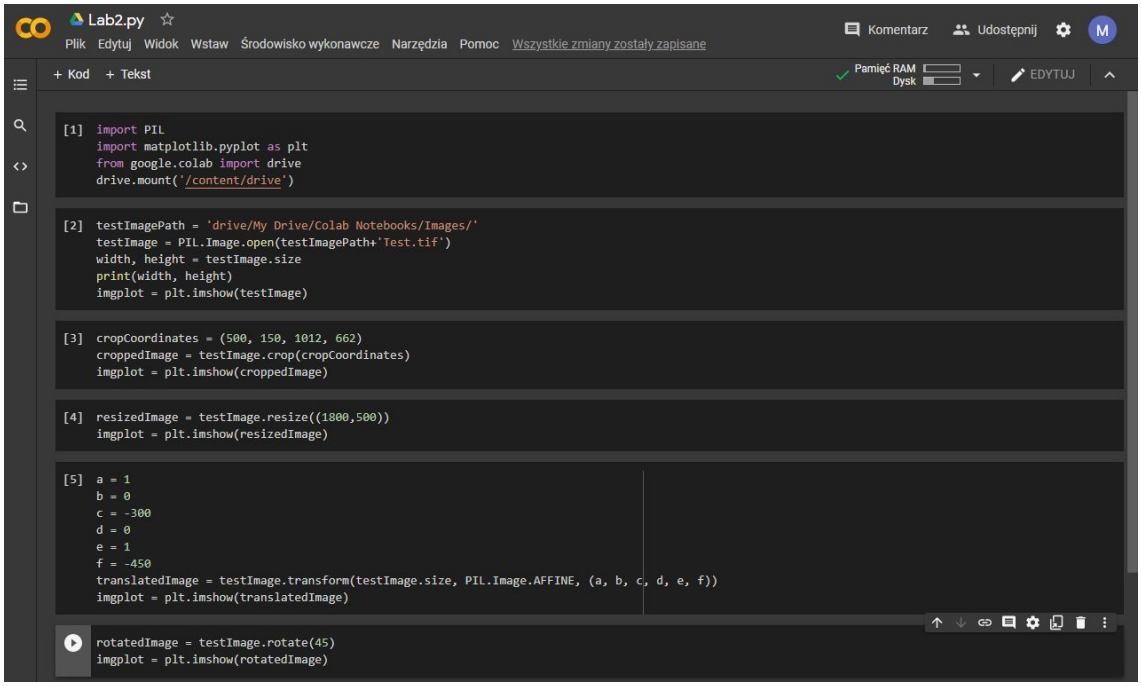
Do wykonania translacji obrazu przy użyciu biblioteki *PIL.Image.AFFINE* najważniejsza jest zmienna *c* oraz zmienna *f*. Zmienna *c* definiuje przesunięcie w kierunku lewo/prawo (-/+) , natomiast zmienna *f* oznacza przesunięcie góra/dół (+/-).

Obrót

Zaskakująco łatwo definiuje się również obrót obrazu przy użyciu biblioteki *PIL.Image*:

```
rotatedImage = testImage.rotate(45)
imgplot = plt.imshow(rotatedImage)
```

Obrót definiowany jest przy użyciu kąta obrotu określonego przy użyciu miary kąta w stopniach. Obrót obrazu, podczas gdy kąt jest zdefiniowany liczbą dodatnią, wykonywany jest odwrotnie do ruchu wskazówek zegara. Aby uzyskać obrót zgodny z ruchem wskazówek zegara należy używać ujemnych wartości kątów obrotu.



Rysunek 1: Efekt końcowy

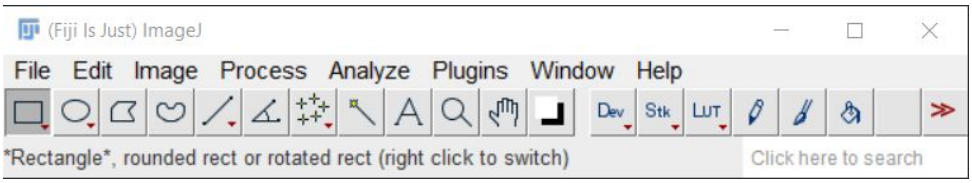
III. Uwagi

Python, Jupyter, Anaconda

Zajęcia laboratoryjne przygotowane zostały z myślą o środowisku Colab. Dzięki temu zadania mogą być wykonywane zarówno na komputerze jak i na innych urządzeniach posiadających dostęp do internetu oraz przeglądarkę internetową takich jak tablety telefony komórkowe czy nawet współczesne telewizory, a nawet konsole do gier. Oczywiście zadania przygotowane w niniejszym laboratorium można wykonywać w odpowiednio skonfigurowanym środowisku Python na komputerze osobistym. Opcja ta może być interesująca dla zaawansowanych użytkowników języka Python, a także innych osób, które z różnych przyczyn wolą pracować w "tradycyjny" sposób. W takim przypadku warto się zapoznać oraz zainstalować środowisko Jupyter oraz system zarządzanie bibliotekami Anaconda. Mimo wszystko gorąco zachęcam do korzystania ze środowiska Colab.

ImageJ

Druga uwaga dotyczy informacji na temat jednej z najlepszych aplikacji do rozpoznawania obrazów, dzięki której można sprawdzić czy uzyskane przy użyciu Pythona wyniki są identyczne do wyników wygenerowanych przez profesjonalne oprogramowanie. Mowa tu o programie ImageJ. Interface programu na pierwszy rzut oka nie zdradza potężnych możliwości programu:



Jednakże po rozwinięciu zawartości różnych przycisków w menu można się przekonać o sporych możliwościach programu oraz licznych dodatkach w zakładce Plugins. Oprócz narzędzi do przetwarzania i rozpoznawania obrazów program ImageJ posiada edytor makr, który pozwala na pisanie własnych kodów wykonawczych w własnym języku IJ1 oraz w innych językach takich jak: Bean Shell, Clojure, Groovy, Java, Java Script, Python, R, Ruby, Scala. Program ImageJ będzie niezbędny do wykonania zajęć laboratoryjnych oraz projektowych. Można go pobrać (najlepiej od razu) ze strony https://imagej.net/Downloads. Po instalacji warto też od czasu do czasu sprawdzić i wykonać aktualizacje, które są dość częste w tym oprogramowaniu.

niektóre zadania zostały wykonane przy użyciu biblioteki Pillow.Image. Import całej biblioteki Pillow nie jest konieczny, wystarczy zaimportować sam moduł Image używając na przykład polecenia:

```
from PIL import Image
```

Do wykonania zadań można również użyć innych bibliotek języka Python jednak znajomość przedstawionych zagadnień z biblioteki Pillow będzie wymagana na zajęciach.

IV. Lista zadań

1. Wczytaj dowolny obrazek oraz wyświetl go przy użyciu biblioteki matlibplot
2. Wczytaj obrazek do zmiennej przy użyciu biblioteki cv2 oraz zapisz obrazek do pliku pod zmienioną nazwą.
3. Wczytaj obrazek do zmiennej oraz zapisz obrazek w innych formacie niż wejściowy.
4. Wczytaj dowolny obrazek oraz odczytaj i wyświetl podstawowe informacje o nim takie jak: rozmiar, rozszerzenie, kanały
5. Wczytaj dowolny obrazek do programu ImageJ zmień typ na 8-bitowy, a następnie zapisz w formacie *.png.
6. Wykonaj modyfikację kodu Kadrowania tak aby na wejściu podawać zmienne lewego górnego rogu okna kadrowania oraz jego szerokość i wysokość.
7. Wykonaj modyfikację kodu Przeskalowania tak aby można było modyfikować szerokość i wysokość obrazu przy użyciu zmiennych skali (jedna zmienna będzie skalować szerokość a druga długość obrazka wynikowego).
8. Wykonaj modyfikację kodu Obrotu tak aby na wejściu podawać wartość kąta w Radianach.
9. Napisz skrypt, który wycina z obrazu wejściowego krok po kroku fragmenty obrazu o rozmiarze 256x256 oraz zapisuje je do plików. Następnie skrypt po kolei wczytuje utworzone pliki obraca je o 180 stopni i ponownie zapisuje do nowego pliku z przyrostkiem 180.

Notatnik z kodami oraz obrazek wynikowy z ImageJ należy przesłać do Prowadzącego najpóźniej do przyszłych zajęć. Brak plików oznacza minus, natomiast 3 minusy obniżają ocenę końcową o pół stopnia.