Uniwersytet Zielonogórski
Instytut Sterowania i Systemów Informatycznych

Rozpoznawanie Obrazów

Semestr VII, Informatyka

mgr inż. Marcin Skobel

2021

Laboratorium nr 1: Programy i pakiety do rozpoznawania obrazów.

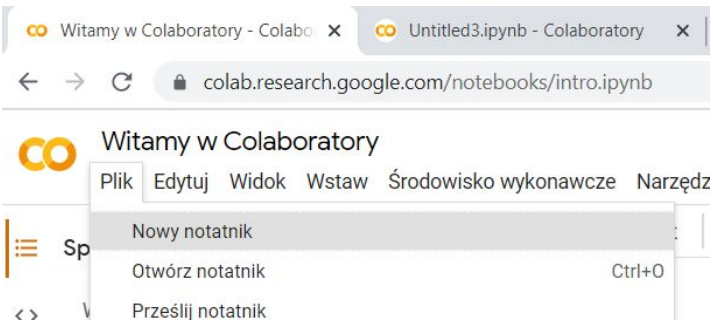
I. Zagadnienia teoretyczne

Wstęp

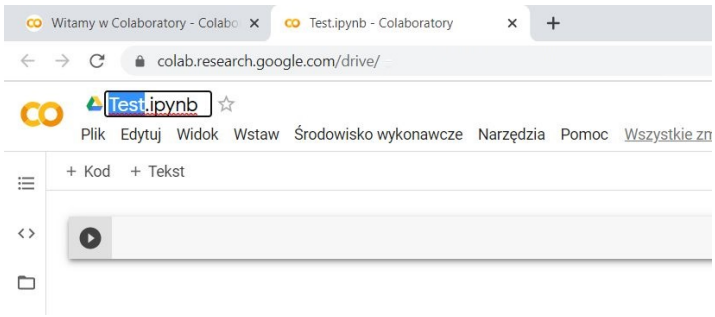
Rozpoznawanie obrazów stanowi jedno z kluczowych zagadnień informatyki związanych z przetwarzaniem obrazów. Z tego powodu większość języków programowania dobrze nadaje się jako narzędzie do rozpoznawania obrazów. Można łatwo wymienić kilka środowisk, które szczególnie dobrze radzą sobie z zadaniem rozpoznawania 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.

Colaboratory - Google Colab

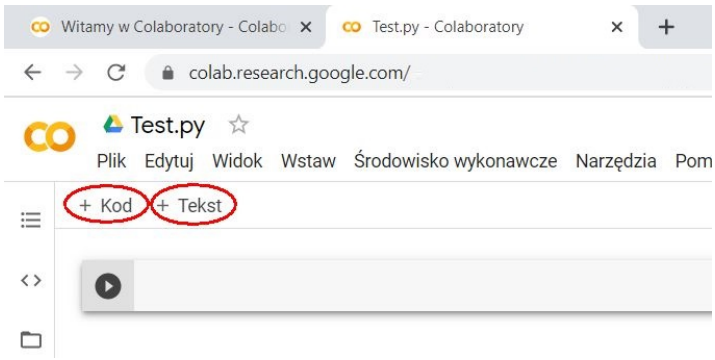
Colab stanowi narzędzie pracy z językiem Python. Do uruchomienia wystarczy aktywne konto Google oraz przeglądarka internetowa. W celu uruchomienia narzędzia Colab wystarczy skorzystać z tego linku: https://colab.research.google.com/. Po wejściu na stronę można utworzyć nowy notatnik:



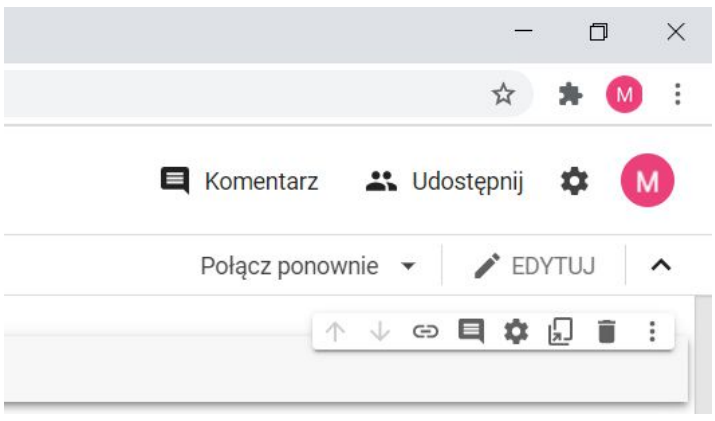
Utworzenie nowego notatnika rozpoczyna pracę z językiem Python, większość narzędzi do przetwarzania obrazów jest już zainstalowana. W przypadku gdy chcemy użyć jakiejś biblioteki której jeszcze nie ma w Colabie wystarczy użyć systemu pip. Po instalacji Colab zapamięta wszystkie zainstalowane pakiety.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

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:

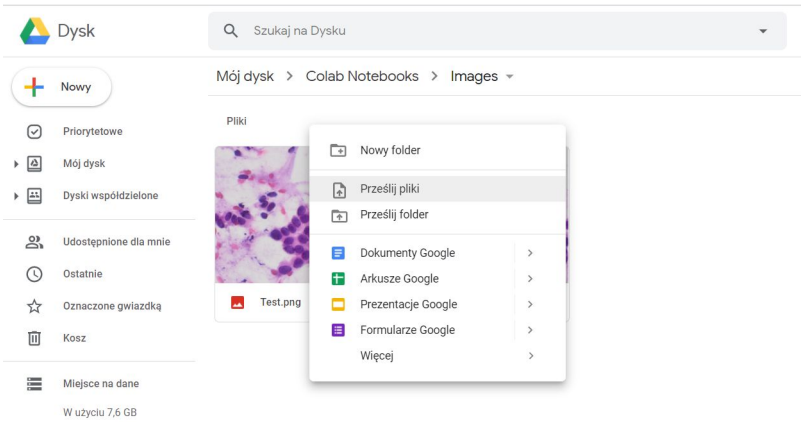
```
from google.colab import drive
drive.mount('/content/drive')
```

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:

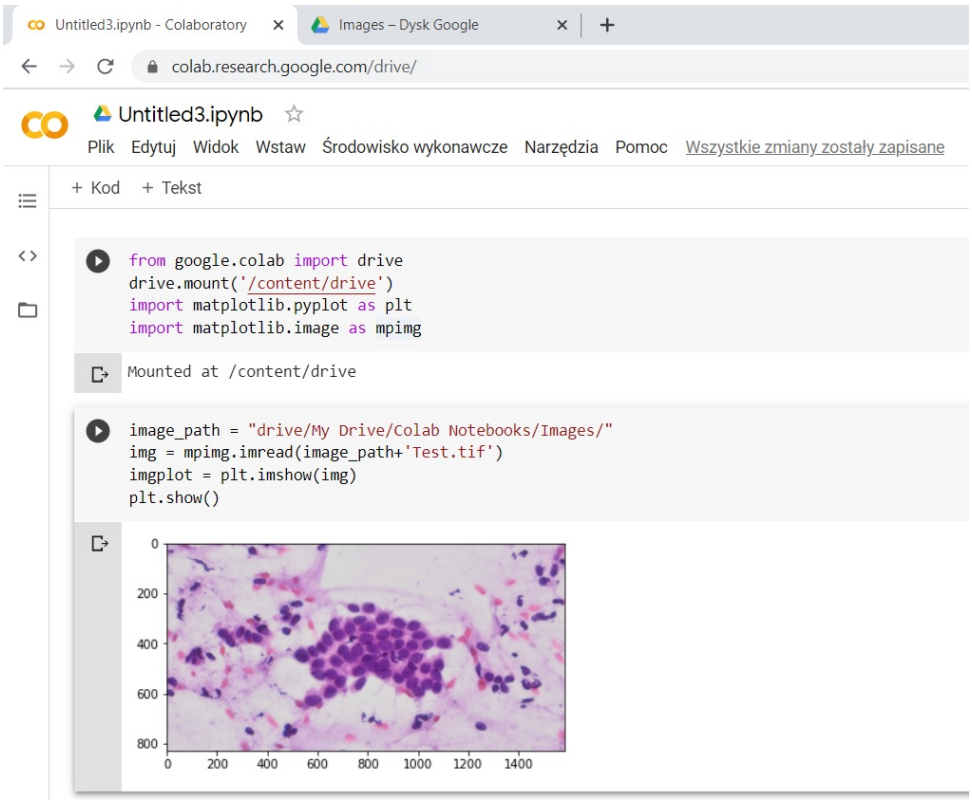


Można też pobrać plik testowy z tego odnośnika: http://staff.uz.zgora.pl/mskobel/Test.tif

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:



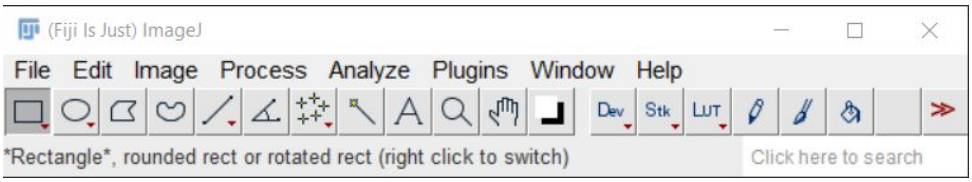
III. Uwagi

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.

IV. Lista zadań

1. Wczytaj dowolny obrazek oraz wyświetl go przy użyciu biblioteki matplotlib
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.

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.