Algorytmy Przetwarzania Obrazów

Algorytmy rozpoznawania obrazów oparte o metody sztucznej inteligencji i testowanie działania aplikacji

WYKŁAD 6

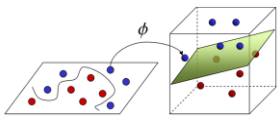
Dla studiów stacjonarnych 2022/2023

Dr hab. Anna Korzyńska, prof. IBIB PAN

Metody klasyfikacji opartej na wektorach wspierających (SVM)

Maszyna wektorów nośnych/wspierających/podpierających

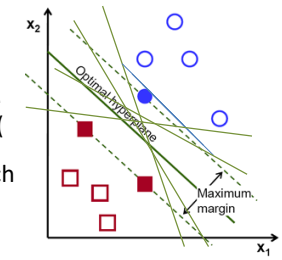
Maszyna wektorów podpierających (*ang*. Support Vector Machine – SVM) to metoda klasyfikacji, która dokonuje transformacji wielowymiarowej przestrzeni cech tak, aby skonstruować na podstawie przykładów (w naszym przypadku obrazów uczących) hiperpłaszczyzny rozdzielające klasy (zwaną hiperpłaszczyzną decyzyjną) w nowej przestrzeni w taki sposób oby były oddzielone z maksymalnym marginesem.



http://enroute.pl/klasyfikacja-metoda-wektorow-nosnych-supporting-vector-machines-svm/

Prosty SVM

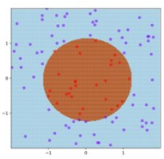
Generujemy hiperpłaszczyzny, które segregują na klasy. Następnie wybieramy właściwą hiperpłaszczyznę o maksymalnej segregacji (równoodległe od najbliższych punktu różnych klas. Z założenie hiperpłaszczyzny powinny być funkcjami liniowymi

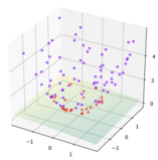


Zaawansowany SVM

Gdy mamy do czynienia z nierozdzielnymi danymi, czyli takimi, których nie można podzielić używając funkcji liniowej w danej przestrzeni algorytm:

- Ignoruje niektóre punkty w przestrzeni cech, uznając je za nieistotnych outsiderów w grupie
- Przenosimy dane do jeszcze wyższych wymiarów (co nazywamy **kernel trick**) z użyciem transformacji opisanej przy użyciu tzw. funkcji jądrowych (**kernel functions**), takich jak: funkcja wielowymiarowa, funkcja Gaussa, hiperboliczna, itp.





https://www.youtube.com/watch?v=efR1C6CvhmE&feature=emb_rel_end
https://tomaszkacmajor.pl/index.php/2016/04/17/support-vector-machine/

Zalety SVM

- Zaletą SVM jest wykorzystywanie do danych i problemów liniowych i nieliniowych
- Obsługuje zmienne ilościowe (ciągłe i zkategoryzowane) oraz jakościowe
- SVM generuje optymalną hiperpłaszczyznę decyzyjną w sposób powtarzalny, który jest wykorzystywany do minimalizowania błędu.
- Znajduje maksymalne odległości (marginesy) pomiędzy grupami punktów
- Efektywna obliczeniowo - złożoność rośnie tylko liniowo wraz z liczbą wymiarów

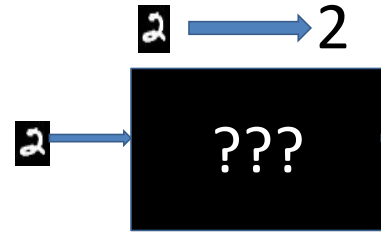
SVM wykorzystujemy nie tylko w klasyfikacji obiektów na obrazach, ale również w uczeniu maszynowym, w analizie regresji i w klasteryzacji.

O rozpoznawaniu z pomocą sieci neuronowych

Rozpoznawanie cyfr zapisanych ręcznie

np.: na czekach - Case study

Jak z obrazu uzyskać informację o cyfrze?



p(obiekt~0)
p(obiekt~1)
p(obiekt~**2**)
p(obiekt~3)
p(obiekt~4)
p(obiekt~5)
p(obiekt~6)
p(obiekt~7)
p(obiekt~8)
p(obiekt~9)

Baza obrazów MNIST

(*Modified National Institute of Standards and Technology)*

- 70 000 obrazów ręcznie pisanych cyfr
- 55 000 uczenie, 5 000 walidacja
- 10 000 testy
- 28x28 pikseli
- Wydany w 1999 roku
- 500 różnych osób
- Obraz + etykieta (jaka to cyfra)
- Skala kolorów znormalizowana do 0.0…1.0

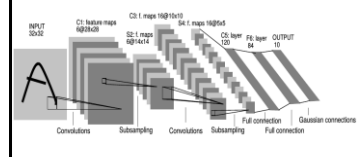
28 px

28 px

28 x 28 = 784 pikseli

Używamy do nauki:
THE MNIST DATABASE of handwritten digits
http://yann.lecun.com/exdb/mnist/

Sieć LeNet




```
lenet5 = keras.models.Sequential()

lenet5.add(keras.layers.Conv2D(filters=6, kernel_size=(5, 5),
                               activation='relu', input_shape=(32,32,1)))
lenet5.add(keras.layers.AveragePooling2D())
lenet5.add(keras.layers.Conv2D(filters=16, kernel_size=(5, 5),
                               activation='relu'))
lenet5.add(keras.layers.AveragePooling2D())
lenet5.add(keras.layers.Flatten())

lenet5.add(keras.layers.Dense(units=120, activation='relu'))
lenet5.add(keras.layers.Dense(units=84, activation='relu'))
lenet5.add(keras.layers.Dense(units=10, activation = 'softmax')) # output
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 28, 28, 6)	156
average_pooling2d_2 (Average	(None, 14, 14, 6)	0
conv2d_3 (Conv2D)	(None, 10, 10, 16)	2416
average_pooling2d_3 (Average	(None, 5, 5, 16)	0
flatten_1 (Flatten)	(None, 400)	0
dense_3 (Dense)	(None, 120)	48120
dense_4 (Dense)	(None, 84)	10164
dense_5 (Dense)	(None, 10)	850

Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0

Praca zaliczeniowa na APO

studenta mgr inż. Brylewa

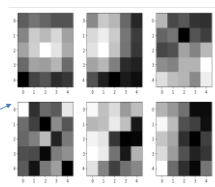
Ref. Y. Lecun, Gradient-Based Learning Applied to Document Recognition, PROCEEDINGS OF THE IEEE. 86 (1998) 47.

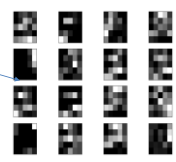
Uczenie się sieci to optymalizacja ze względu na cel

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 28, 28, 6)	156
average_pooling2d_2 (Average	(None, 14, 14, 6)	0
conv2d_3 (Conv2D)	(None, 10, 10, 16)	2416
average_pooling2d_3 (Average	(None, 5, 5, 16)	0
flatten_1 (Flatten)	(None, 400)	0
dense_3 (Dense)	(None, 120)	48120
dense_4 (Dense)	(None, 84)	10164
dense_5 (Dense)	(None, 10)	850

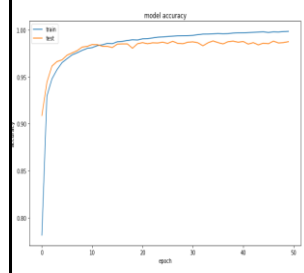
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0



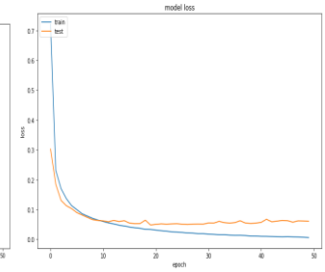


[-0.0789532 -0.02418987 0.10222547 0.00066162 -0.04760809
0.09279951 -0.01421791 -0.07095297 0.05741353 -0.01717872]

Parametry wyuczenia sieci

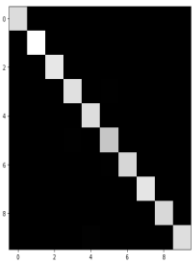


Najlepszego dopasowania



Minimalizacji błędu

Kontrola błędów



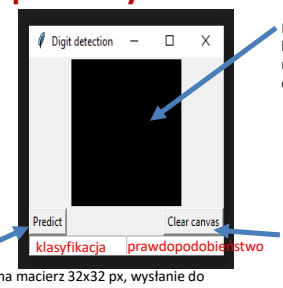
```
[ 974    0    0    0    0    0    2    1    3    0]
[   0 1130    1    0    0    1    2    0    0    1]
[   4    1 1020    0    3    0    0    2    2    0]
[   1    0    2  993    0    6    0    4    2    2]
[   0    0    0    0  976    0    2    2    0    2]
[   2    0    0    7    0  875    2    2    3    1]
[   2    2    0    0    1    5  947    0    1    0]
[   0    3    4    1    1    0    0 1015    2    2]
[   1    0    1    4    3    3    1    2  955    4]
[   2    2    0    4   10    3    0    4    0  984]]
```

```
result = lenet5.evaluate(X_test, y_test)
dict(zip(lenet5.metrics_names, result))
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.0498 - accuracy: 0.9869
{'loss': 0.04901135712862015, 'accuracy': 0.9868999719619751}
```

Wizualizacja macierzy pomyłek

Moduł wykonawczy do rozpoznawania znaków narysowanych przez użytkownika

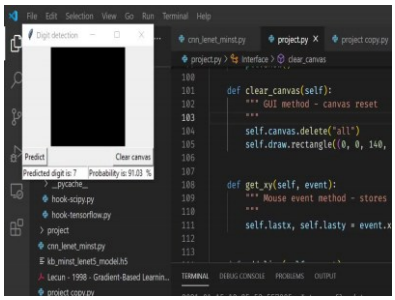


Powierzchnia na której można narysować cyfrę do rozpoznania

Czyszczenie powierzchni do rysowania

Zamiana obrazu na macierz 32x32 px, wysłanie do modelu, wyliczenia prawdopodobieństw przynależności do jednej z 10 klas i wybór prawdopodobieństwa maksymalnego

Działanie programu rozpoznającego



- Python 3.7
- Numpy
- Matplotlib
- Tensorflow 2.4 – wymaga AVX w zestawie instrukcji CPU!!!
- CUDA 10.1
- SciKit
- Tkinter - GUI

Moduł inferencji

Testowanie aplikacji

Testowanie aplikacji

to **proces** zapewnienia jakości **oprogramowania** związany **z weryfikację oraz walidację** oprogramowania:

- weryfikacja oprogramowania pozwala skontrolować, czy wytwarzane oprogramowanie jest **zgodne ze specyfikacją,** a

- wykonana **analiza statystyczna oceniająca działanie oprogramowania** jest jego walidacja.

- Testowanie **nie jest w stanie wykryć wszystkich defektów oprogramowania**, jednak może dostarczyć informacji o jego zgodności z wymaganiami klienta, czy też z jego oczekiwaniami.
- Testowanie nie sprawdza oprogramowania pod kątem wszelkich możliwych warunków początkowych, lecz jedynie w wyselekcjonowanych warunkach.
- Testowanie może we wczesnych fazach projektu wykryć defekty nie tylko oprogramowania, ale i specyfikacji wymagań czy projektu (niejednoznacznymi lub sprzeczności wymagań uniemożliwiające określić poprawnego zachowanie systemu)
- Testowanie oprogramowania sprowadza się również do **analizy statycznej występowania** poprawnych i niepoprawnych odpowiedzi oprogramowania za pomocą twn.: tablicy pomyłek

Rodzaje testów oprogramowania

Testy można podzielić na kilka sposobów:

- ze względu na weryfikowane obiekty (przykładowo testy klas, komponentów, podsystemów, systemu lub zintegrowanych systemów)
- na białoskrzynkowe (strukturalne) - weryfikujące kod źródłowy oraz czarnoskrzynkowe testujące warstwę interfejsu
- bazujące na wymaganiach:
  - testy weryfikujące zgodność implementacji z wymaganiami, np. testy funkcjonalne, testy graficznego interfejsu użytkownika),
  - testy niefunkcjonalne –
    np.: **FURPS(+)** zdefiniowana w ramach Rational Unified Process (RUP)
- ze względu na metodę weryfikacji z wyróżnieniem:
  - testów statycznych, bez uruchomienia aplikacji
  - testów dynamicznych wymagającej pracę na uruchomionym oprogramowaniu
- dodatkowo można wyróżnić testy wykonane w określonym celu:
  - retesty – testy poprawek defektów
  - testy regresywne – testy oprogramowania po wykonaniu zmian, niekoniecznie w kodzie (przykładowo po wykonaniu aktualizacji wersji systemu operacyjnego)

Functionality (Funkcjonalność)
Usability (Używalność)
Reliability (Niezawodność)
Performance (Wydajność)
Supportability (Wsparcie)

- **Testy funkcjonalne.**
  Polegają one na tym, że wcielamy się w rolę użytkownika, traktując oprogramowanie jak „czarną skrzynkę", która wykonuje określone zadania. Nie wnikamy w ogóle w techniczne szczegóły działania programu. Testy są nazywane testami czarnej skrzynki jeśli nie wymagają patrzenia w kod

Uwaga! Częstym błędem jest stawianie znaku równości między testami funkcjonalnymi, a testami czarnej skrzynki. Testy funkcjonalne mogą wymagać umiejętności czytania kodu źródłowego, czego nie wymaga się przy testach interfejsów zewnętrznych.

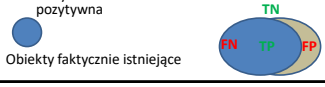
- **Testy strukturalne**.
  Tym razem tester ma dostęp do kodu źródłowego oprogramowania, może obserwować jak zachowują się różne części aplikacji, jakie moduły i biblioteki są wykorzystywane w trakcie testu. Te testy czasami są nazywane testami białej skrzynki

Metryki oceny jakości oparte na tablicy pomyłek

Tablica pomyłek służy do oceny jakości klasyfikacji (ang. Cofusion table/matix)

- *True-Positive (**TP** – prawdziwie pozytywna)*: przewidywanie pozytywne, faktycznie zaobserwowana klasa pozytywna
- *True-Negative (**TN** – prawdziwie negatywna)*: przewidywanie negatywne, faktycznie zaobserwowana klasa negatywna
- *False-Positive (**FP** – fałszywie pozytywna)*: przewidywanie pozytywne, faktycznie zaobserwowana klasa negatywna
- *False-Negative (**FN** – fałszywie negatywna)*: przewidywanie negatywne, faktycznie zaobserwowana klasa pozytywna

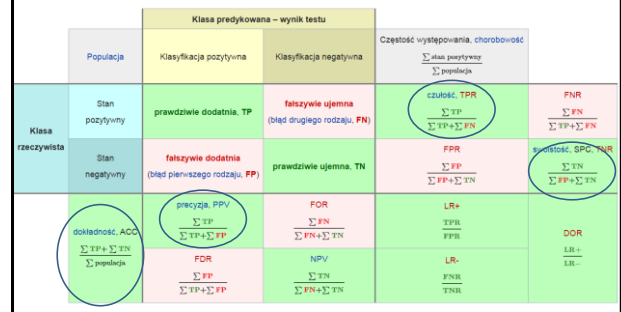
		Stan faktyczny	
		P	***N***
Przewidywanie/znalezienie/sklasyfikowanie	***P***	***TP*** *True-Positive*	***FP*** *False-Positive*
	N	***FN*** *False-Negative*	***TN*** *True-Negative*



Obiekty faktycznie istniejące

Obiekty znalezione/sklasyfikowane

Tablica pomyłek



Podsumowując:

Testowania umożliwia wykrycie błędów we wczesnych stadiach rozwoju oprogramowania, co pozwala zmniejszyć koszty usuwania tego błędu.

- Warto testować na każdym etapie tworzenia oprogramowania.
- Testować należy jak najwcześniej, ponieważ im później wykryty zostanie błąd tym większy jest koszt jego usunięcia.

Przegląd tematów mini projektów

Inr.	Tytuł projektu dla APOZ 2022/2023	Nr indeksu studenta
1	Segmentacja obrazu monochromatycznego/binarnego zawierających cyfry i ich rozpoznawania z zastosowaniem porównania ze wzorcem	18593
2	Rozwinięcie opcji zamiany modeli koloru w obrazach kolorowych i przedstawienia linii profilu dla wszystkich wybranych modeli koloru	17677
3	Implementacje funkcji linii profilu wzdłuż wskazanej myszką prostej lub krzywej w obrazach monochromatycznych i kolorowych	19132
4	Segmentacja obrazu monochromatycznego/binarnego zawierającego znaki pisarskie i wyszukiwanie znaków o kształtach wypukłych np.: litery w tekście: D, O, itp..	16410
5	~~Segmentacja obrazu monochromatycznego/binarnego zawierającego znaki pisarskie i wyszukiwanie znaków o kształtach wklęsłych np.: *, C, itp..~~	
6	~~Segmentacja obiektów na podstawie zakresu intensywności wskazanego na wszystkich kanałach (dającego określony zakres kolorów) oraz implementacja algorytmu zliczania wysegmentowanych obiektów z mapy binarnej z użyciem stosu.~~	
7	Prezentacja obrazu w postaci wykresu 3D	18605
8	Oprogramowanie algorytmu projekcji wstecznej w obrazach i wykorzystanie tego narzędzia do segmentacji	
9	Oprogramowania do segmentacja obrazów monochromatycznych barwionych tkanek z użyciem algorytmu wododziału.	18555
10	Implementacja funkcji detekcji całej twarzy i oczu za pomocą kaskadowego klasyfikatora opartego na cechach Haara oraz implementacja rozpoznawania własnej twarzy	19381
11	Implementacja operacji erozji i dylacji działających na dowolnie zdefiniowanym elemencie strukturalnym/struturyzującym	18673

12	Przeniesienie oprogramowania stworzonego na zajęciach przedmiotu APO tak, aby działał dla systemu operacyjnego iOS	18657
13	~~Implementacja operacji wyliczenia transformaty odległościowej i wykorzystanie jej do rozdzielenia obiektów stykających się~~	
14	~~Segmentacja z obrazu monochromatycznego/binarnego zawierającego znaki pisarskie o różnych kształtach, poszukujemy wśród nich znaków o kształtach wklęsłych np.: *, C, itp..~~	
15	Segmentacja obrazu monochromatycznego/binarnego zawierających cyfry i ich rozpoznawania z zastosowaniem sieci neuronowych	19010
16	Segmentacja obrazów kolorowych z wykorzystaniem klasteryzacji.	18678
17	Implementacja progowania obrazu prawdopodobieństwa przypasowania do zadanej tekstury przy użyciu wariancji jasności	19441
18	Implementacja narzędzia do tworzenia panoramy na postawie serii zdjęć kolorowych	17801
19	Implementacja wybranych funkcjonalności w przestrzeni HSV	15719
20	Implementacja ekstrakcji linii pionowych i poziomych za pomocą operacji morfologicznych	18827
21	~~Segmentacja z obrazu monochromatycznego/binarnego obszarów zawierających obiekty o kształtach zawierających odcinki zbiegające po różnym kątem (rogi) z wykorzystaniem morfologii matematycznej~~	
22	~~Implementacja operacji filtracji logicznych na obrazach binarnych; rozwinięcie możliwości wyświetlania i zapisywania jako obraz uzyskiwanych na podstawie fragmentów histogramu~~	
23	Program prezentacji zasad przebiegu procesu wprowadzania i korekcji zniekształceń radiometrycznych z wykorzystaniem obrazów	18524

24	Program prezentacji sposobu działania metody α-NN (α najbliższych sąsiadów) z wizualizacją przestrzeni cech przed i po fazie uczenia oraz w czasie klasyfikacji	
25	Implementacja operacji przenikania obrazów monochromatycznych	17669
26	Implementacja narzędzia do rozciągania i zawężania histogramu z jednoczesnym zastosowaniem funkcji logarytmicznej i odwróconej funkcji logarytmicznej	17340
27	~~Implementacja operacji wyliczania otoczki wypukłej obiektu w obrazie binarnym~~	
28	Implementacja operacji wyliczenia średniej i średniej kroczącej (okienkowej) z serii obrazów	18060
29	Program ukrywania i odczytu obrazu w pliku graficznym	19074
30	~~Implementacja operacji rekonstrukcji morfologicznej~~	
31	Implementacja odszumiania przez uśredniania obrazów tego samego obiektu oraz implementacja operacji różnicy A-B i B-A	18249
32	Implementacja operacji regulowanego rozciągania zakresów poziomów jasności w obrazach monochromatycznych analogicznie do rozwiązania prezentowanego na wykładzie	15296
33	Implementacja operacji tworzenia histogramu dwuwymiarowego z obrazu monochromatycznego i jego matematycznego przekształcenia oraz rekonstrukcji obrazów z histogramu	18817
34	Segmentacja obrazu monochromatycznego zawierających emotikony z różnymi emocjami	19125
35	Segmentacja obrazu monochromatycznego zawierających drobne obrazy symboliczne obserwowane przez kamerę na taśmie produkcyjnej i otoczenie ich prostokątem dopasowanym do ich rozmiarów.	17946
36	Program do konstrukcji obrazów monochromatycznych z obrazów kolorowych według wskazań użytkownika co do konwersji koloru	18755

37	Implementacja wyrównania histogramu obrazu kolorowego wykorzystując dowolne lub wszystkie kanały kolorów w modelu L*a*b*	17920
38	Implementacja funkcji reprezentacji obrazu monochromatycznego w postaci obrazów o zawężonym zakresie poziomów szarości wyznaczonych według wskazań użytkownika	14223
39	Przeniesienie oprogramowania stworzonego na zajęciach tak, aby działał dla systemu operacyjnego ANDROID	18715
40	Segmentacja obrazu monochromatycznego zawierających cyfry i litery (jak w polskich tablicach rejestracyjnych samochodów).	18584
41	Segmentacja obrazu monochromatycznego zawierających drobne metalowe element obserwowane przez kamerę na taśmie produkcyjnej i otoczenie ich prostokątem dopasowanym do ich rozmiarów.	19027
42	Program do pseudokolorowania obrazów monochromatycznych według skali barw odpowiadającej tęczy	18880
43	Implementacja funkcji wykonywania wyrównania histogramu obrazu kolorowego wykorzystując HSV lub HSI	17832
44	Implementacja funkcji reprezentacji obrazu monochromatycznego w postaci ośmiu binarnych obrazów dla każdego bitu oddzielnie	18141
45	Segmentacja obrazów z wykorzystaniem klasteryzacji kolorów	18644
46	Ręczny podział obiektów sklejonych	17990
47	Rekonstrukcja obrazu z zakłóceniami w postaci jasnych pikseli lub linii	18896
48	~~Implementacja progowania obrazu prawdopodobieństwa przypasowania do zadanej tekstury przy użyciu filtrów Gabora~~	
49	~~Implementacja progowania obrazu prawdopodobieństwa przypasowania do zadanej tekstury przy~~	

50	Implementacja progowania obrazu prawdopodobieństwa przypasowania do zadanej tekstury przy użyciu metody SIFT	18533
51	Implementacja funkcji detekcji całej twarzy i oczu lemurów za pomocą kaskadowego klasyfikatora opartego na cechach Haara	19076
52	Segmentacja z obrazu monochromatycznego obszarów zawierających obiekty o kształtach zawierających linie zbiegające pod różnym kątem (rogi) z wykorzystaniem metody Harris-Stephens'a.	18873
53	Segmentacja obrazu monochromatycznego przedstawiającego klocki lego obserwowane przez kamerę na taśmie produkcyjnej i otoczenie ich prostokątem dopasowanym do ich rozmiarów	17237
54	Segmentacja obrazu kolorowego przedstawiającego ziarna kawy, fasoli i soczewicy obserwowane przez kamerę na taśmie produkcyjnej i klasyfikacje ziaren ze względu na kształt i kolor	18546
55	Segmentacja obrazu kolorowego przedstawiającego klocki lego obserwowane przez kamerę na taśmie produkcyjnej i klasyfikacje klocków ze względu na kolor	18371
56	Implementacje funkcji linii profilu wzdłuż krzywej i łamanej zbudowanej na wskazanych przez użytkownika punktach.	18824
57	~~Program do wykonywania procesu wprowadzania i korekcji zniekształceń radiometrycznych z wykorzystaniem ciemnego i jasnego obrazu odniesienia~~	
58	~~Oprogramowania do segmentacja obrazów monochromatycznych komórek w hodowli z użyciem algorytmu wododziału.~~	
59	~~Implementacje funkcji linii profilu wzdłuż okręgów i elips o zadanych parametrach i wskazanej lokalizacji środka figur.~~	
60	Przeniesienie oprogramowania stworzonego na zajęciach tak, aby działał dla systemu	18709

61	Przeniesienie oprogramowania stworzonego na zajęciach tak, aby działał dla systemu operacyjnego iOS	18773
62	Implementacja narzędzia do tworzenia panoramy na postawie serii zdjęć monochromatycznych	18868
63	Oprogramowanie do wyszukiwania obiektów analogicznych do obiektów wskazanych na próbkach (małych zdjęciach przedstawiających poszukiwany obiekt)	19092
64	Wyszukiwanie i lokalizacja emotikonów umieszczanych w tekście.	17804
65	Program do pseudokolorowania obrazów monochromatycznych według zestawu barw występujących w mapach geograficznych	14407
66	~~Udoskonalenie oprogramowania przygotowanego na zajęciach przez dołożenie operacji zmniejszenia udziału szumu przez uśrednianie obrazów zebranych w takich samych warunkach oraz implementacje progowania odwrotnego do wszystkich trzech progowań implementowanych na zajęciach~~	
67	Udoskonalenie oprogramowania przygotowanego na zajęciach przez implementację operacji przenikania obrazów monochromatycznych według skali podanej w procentach	17694
68	Implementacja operacji unsharpen mask	16745
69	Implementacja operacji tworzenia histogramu dwuwymiarowego z obrazu kolorowanego bazującego na kanałach oraz rekonstrukcji obrazów z histogramu	17203
70	~~Implementacja narzędzia do manipulacji widmem amplitudowym obrazu w celu modyfikacji udziału poszczególnych składowych widma~~	
71	Implementacja narzędzia do manipulacji widmem amplitudowym obrazu w celu likwidacji zakłóceń periodycznych	18608

72	~~Implementacja narzędzia do manipulacji widmem amplitudowym obrazu typu portret w celu tworzenia „uśmiechu Mony Lizy"~~	
73	**Implementacja narzędzia do manipulacji widmem amplitudowym obrazu w celu wydzielania wybranego zakresu częstotliwości**	17916
74	**Segmentacja obrazu kolorowego przedstawiającego klocki lego obserwowane przez kamerę na taśmie produkcyjnej i klasyfikacje klocków ze względu na kolor**	18683
75	~~Segmentacja obrazu monochromatycznego przedstawiającego ziarna kawy, fasoli i soczewicy obserwowane przez kamerę na taśmie produkcyjnej i klasyfikacje klocków ze względu na poziom jasności~~	
76	~~Segmentacja obrazu monochromatycznego przedstawiającego różnej wielkości śruby obserwowane klocków ze względu na poziom jasności przez kamerę na taśmie produkcyjnej i ich klasyfikacja ze względu na rozmiar~~	
77	**Segmentacja obrazu kolorowego przedstawiającego drewniane klocki obserwowane przez kamerę na taśmie produkcyjnej i ich klasyfikacje klocków ze względu rozmiar**	18693
78	**Segmentacja obrazu monochromatycznego przedstawiającego ziarna kawy, fasoli i soczewicy obserwowane przez kamerę na taśmie produkcyjnej i ich klasyfikacje klocków ze względu na kształt**	18647

Wybrano 67 z 78 tematów

Zaliczenie przedmiotu

- Laboratorium zalicza 26-50 punktów zdobytych na zajęciach na podstawie ocenionej przez prowadzącego aplikacji pozostawionej na dysku P w odpowiednim katalogu
- Do egzaminu mogą przystąpić Ci studenci którzy mają minimum 26 i którzy przedstawili działające oprogramowanie projektu egzaminacyjnego

Egzamin

Obrona przygotowanego samodzielnie projektu następuje po jego przykazaniu:

1. Projekt należy przed egzaminem (data zostanie ogłoszona) **włożyć do** tego samego **katalogu**, w którym jest aplikacja rozwijana na zajęciach (**na dysku P**) w formie wykonywalnej wraz z kodem. W niektórych przypadkach będzie to ten sam projekt z rozwiniętymi lub dodanymi funkcjonalnościami.
2. Oprócz projektu egzaminacyjnego, proszę wgrać: **instrukcję dla użytkownika** oraz **7-10 minutowy film** z wykładem prezentującym projekt. Film będzie oceniany według kryteriów:

Co będzie oceniane

- Zrozumienia stosowalności i założeń projektu
- Kompletność prezentacji – podstawy, metody i funkcjonalności
- Sposób prezentacji
- Organizacja prezentacji

Kod będzie sprawdzany pod kątem

- Struktury oprogramowania
- Poziomu komentowania kodu
- Opisu zmiennych, parametrów globalnych i lokalnych
- Interface graficzny - funkcjonalność
- Interface graficzny - walory graficzne
- Oceny zgodności rezultatów z założeniami

Ocena dokumentacji

- Organizacja dokumentacji
- Kompletność dokumentacji
- Szata graficzna
- Dobór obrazów demonstracyjnych
- Dobór zrzutów z ekranów
- Spis treści i wyszukiwanie informacji

Za całość będzie można dostać 50 punktów

Zaliczenie przedmiotu to suma punktów za laboratoria i z egzaminu przeliczona na stopnie

Materiał:

- M.Doros, Przetwarzanie obrazów, skrypt WSISIZ
- Materiały wykładowe POBZ z zeszłego roku na UBIKu
- T.Pavlidis, Grafika i Przetwarzanie Obrazów, WNT Warszawa 1987.
- **I.Pitas,** Digital image processing, algorithms and applications, John Wiley &Sons, Inc. 2000, **pp. 162-166** (**w katalogu …*APOZ**Materialy* na UBIKu**).

Literatura dodatkowa:

•W.Zieliński, M.Strzelecki: Komputerowa analiza obrazu biomedycznego, PWN Warszawa-Łódź 2002, str. 178-214; segmentacja z wykorzystaniem analizy tekstur, mozaika Voronoi (Voronoi tesselation), segmentacja metodą określenia działów wodnych (watershed transform)
•T.Pavlidis: Grafika i Przetwarzanie Obrazów, WNT Warszawa 1987
•R.Tadeusiewicz, P.Korohoda, Komputerowa analiza i przetwarzanie obrazów, Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków 1997. http://winntbg.bg.agh.edu.pl/skrypty2/0098/
•Zasoby sieciowe:
•Segmentacja (w szczególności wododziałowa *(watershed)*)
•https://www.mathworks.com/company/newsletters/articles/the-watershed-transform-strategies-for-image-segmentation.html
•Definicja tekstury:
•http://ai.stanford.edu/~ruzon/tex_seg/node1.html