Zaawansowne algorytmy wizyjne

materiały do ćwiczeń laboratoryjnych

Piotr Pawlik, Tomasz Kryjak

PUBLISHED BY AGH

First printing, March 2018

Contents

1 — Transformata Fouriera-Mellina

TODO Rozudowac

1.1 Cel zajęć

- zapoznanie z zagadnieniem wyszukiwania wzorców niezależnie od ich rozmiaru i orientacji za pomocą transformaty Fouriera-Mellina

1.2 Wyszukiwanie wzorca za pomocą korelacji

- Ze strony kursu pobierz archiwum z danymi do ćwiczenia i rozpakuj je we własnym katalogu roboczym. UWAGA - W dzisiejszym ćwiczeniu do wyświetlania obrazów lepiej jest używać funkcji imshow z modułu pyplot, a nie cv2.
- Utwórz nowy skrypt. Wyszukaj wzorzec zapisany w obrazie 'wzor.pgm' na przeszukiwanym obrazie 'domek_r0.pgm'. Oba obrazy wczytaj w odcieniach szarości. Zacznij od uzupełnienia obraz wzorca zerami do rozmiaru obrazu przeszukiwanego - przykładowy kod:

  ```
  wzorzec_z_zerami = np.zeros(obraz_przeszukiwany.shape)
  wzorzec_z_zerami[0:wzorzec.shape[0], 0:wzorzec.shape[1] = wzorzec
  ```

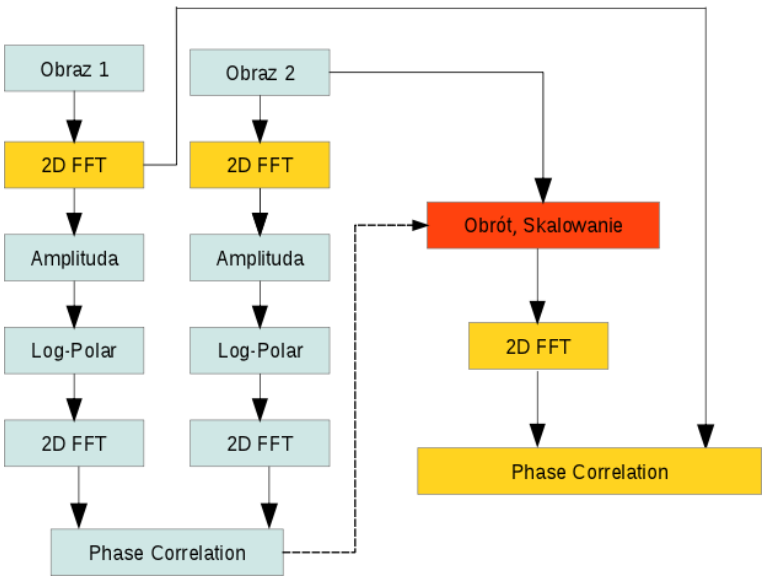
  Przeprowadź korelację obrazu przeszukiwanego i obrazu wzorca z zerami w dziedzinie częstotliwości. Do wyliczenia transformaty Fouriera i transformaty odwrotnej wykorzystaj funkcje fft2 oraz ifft2 z modułu numpy.fft. Współrzędne maksimum w obrazie modułu transformaty odwrotnej można uzyskać przy użyciu instrukcji:
  y,x = np.unravel_index(np.argmax(modul_odwrotnej), modul_odwrotnej.shape)
  Moduł liczby zespolonej uzyskujemy za pomocą funkcji np.abs.
  Sprawdź, czy maksimum wypada w miejscu występowania wzorca na obrazie przeszukiwanym (raczej nie powinno).
- Zmień korelację na korelację fazową. Sprawdź, czy maksimum wypada w miejscu występowania wzorca na obrazie przeszukiwanym (teraz powinno być dobrze - maksimum to lewy górny róg wzorca).
- Zwizualizuj przesunięcie przekształcając obraz wzorca z zerami za pomocą funkcji realizującej przekształcenie afiniczne - cv2.warpAffine. Przykładowy kod:

```
macierz_translacji = np.float32([[1,0,dx],[0,1,dy]])  # gdzie dx, dy -
    wektor przesuniecia
obraz_przesuniety = cv2.warpAffine(obraz_wzorca_z, macierz_translacji,
    (obraz_wzorca_z.shape[1], obraz_wzorca_z.shape[0]))
```

1.3 Wyszukiwanie wzorca niezależnie od obrotu i skali

W poprzednim punkcie nie miało znaczenia położenie składowej stałej w obrazie po FFT. Obecnie jednak składowa stała musi zajdować się w środku obrazu. W tym celu po fft2() należy wykonywać fftshift().

- Wczytaj obraz przeszukiwany 'domek_r30.pgm' oraz obraz wzorca 'domek_r0_64.pgm'. Zrealizuj obliczenia z poniższego schematu uwzględniając następujące uwagi:



- Uzupełnij zerami mniejszy obraz tak jak w punkcie 2 ale z uwzględnieniem okna Hanninga uzyskanym funkcją:

```
def hanning2D(n):
    h = np.hanning(n)
    return np.sqrt(np.outer(h,h))
```

  Okno przemnażamy przez obraz przed uzupełnieniem go zerami. n to rozmiar obrazu w pionie lub poziomie (zakładamy, że są takie same)
- Przed transformacją log-polar przefiltruj obrazy filtrem górnoprzepustowym uzyskanym funkcją:

```
def highpassFilter(size):
    rows = np.cos(np.pi*np.matrix([-0.5 + x/(size[0]-1) for x in range(
    size[0])]))
    cols = np.cos(np.pi*np.matrix([-0.5 + x/(size[1]-1) for x in range(
    size[1])]))
    X = np.outer(rows,cols)
    return (1.0 - X) * (2.0 - X)
```

  size to shape obrazu filtrowanego. Filtrujemy w dziedzinie częstotliwości, a więc polega to na przemnożeniu obrazów amplitud przez filtr.
- Transformatę log-polar można zrealizować za pomocą funkcji cv2.logPolar. Środek przekształcenia to środek obrazu, natomiast parametr M proszę ustawić na: 2*R/np.log(R)

gdzie R to max. promień, czyli połowa rozmiaru pionowego lub poziomego. Parametr flags to: cv2.INTER_LINEAR + cv2.WARP_FILL_OUTLIERS. Czyli przykładowe użycie tej funkcji to:

```
srodek = (obraz_abs_z_fft.shape[0]//2, obraz_abs_z_fft.shape[1]//2)
M = obraz_abs_z_fft.shape[0]/np.log(obraz_abs_z_fft.shape[0]//2)
cv2.logPolar(obraz_abs_z_fft, srodek, M, cv2.INTER_LINEAR + cv2.
    WARP_FILL_OUTLIERS)
```

- Uzyskane w wyniku pierwszej korelacji fazowej współrzędne maksimum (wsp_kata, wsp_logr) przeliczamy na skalowanie i stopnie wg wzorów:

```
skala = np.exp(wykl/M)    #   gdzie M to parametr funkcji cv2.logPolar,
    a wykl wyliczamy jako:

if wsp_logr > rozmiar_wsp_logr//2:
        wykl =rozmiar_wsp_logr - wsp_logr
else:
        wykl = - wsp_logr

kat1  = - A                  # gdzie    A =  (wsp_kata * 360.0 ) /
    rozmiar_wsp_kata
kat2  =  180 - A
```

  Kąty są dwa, gdyż ze względu na symetrię modułu widma częstotliwościowego wykrywane są obroty tylko do 180 stopni. Dlatego w nastepnym kroku trzeba sprawdzić oba kąty i wybrać ten, który daje lepszą korelację.
- Wyliczone kąty i skalę należy użyć w przekształceniu afinicznym, podobnie jak w zadaniu 1.2. Tym razem macierz translacji będzie wyglądała następująco:

```
macierz_translacji = cv2.getRotationMatrix2D((srodekTrans[0],
    srodekTrans[1]), kat, skala) gdzie srodekTrans to srodek obrazu:
srodekTrans = [math.floor((obraz.shape[0] + 1) / 2), math.floor((obraz.
    shape[1] + 1 ) / 2)]
```

- Przetransformowane obrazy należy poddać transformacie Fouriera i skorelować z widmemem obrazu przeszukiwanego. Z wyniku dającego większą korelację wyliczamy współrzędne przesunięcia. Zwizualizuj przesunięcie analogicznie jak w zadaniu 1.2.
- Sprawdź poprawność detekcji wzorca dla pozostałych obrazów - obróconych (domek_rxx) i przesuniętych (domek_sx).

Bibliography