# Przetwarzanie informacji obrazowej
## Temat 4

Piotr Syga

2022

# Przekształcenia afiniczne

Podstawowe przekształcenia obrazu:

- obroty
- przesunięcia
- przeskalowanie

$$x' = Ax + By + C$$
$$y' = Dx + Ey + F$$

Zmiana układu współrzędnych – Jakobian

$$J = \left| \frac{\partial(x', y')}{\partial(x, y)} \right| = \left| \begin{pmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} \\ \frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y} \end{pmatrix} \right|$$

Dla powyższej transformacji $J = AE - BD$.
Dodatkowa wiedza jeśli $J = 0$ lub $J = 1$.

# Przekształcenia afiniczne

Zastosowania:

- identyfikacja obiektu na zasadzie słownikowej,
- detekcja ruchu,
- korekcja nagrania.

## Przykłady

Obrót: $x' = x\cos\phi + y\sin\phi,\ y' = -x\sin\phi + y\cos\phi$, zatem $J = 1$

Przesunięcie $x' = x + t_x,\ y' = y + t_y,\ J = 1$

Pochylanie: $x' = x + y\,\mathrm{tg}\,\phi,\ y' = y,\ J = 1$

Reskalowanie: $x' = Ax,\ y' = By,\ J = AB$

# Obroty i przesunięcia

Obrót: $x' = x\cos\phi + y\sin\phi$, $y' = -x\sin\phi + y\cos\phi$,
Przesunięcie $x' = x + t_x$, $y' = y + t_y$.
Zamiast punktów, obróćmy osie, zatem
$e'_x = e_x\cos\phi + e_y\sin\phi$, $e'_y = -e_x\sin\phi + e_y\cos\phi$, dla wektorów jednostkowych $e_i$.
Analogicznie, dodając wektory przesunięcia:
$xe_x + ye_y = t_x e_x + t_y e_y + x'e'_x + y'e'_y$, dla wektorów jednostkowych $e_i$.

$$xe_x + ye_y = t_x e_x + t_y e_y + x'(e_x\cos\phi + e_y\sin\phi) + y'(-e_x\sin\phi + e_y\cos\phi)$$

Zapisując macierzowo:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = R' \begin{bmatrix} x' \\ y' \end{bmatrix} + T'$$

# Obroty i przesunięcia

Obrót: $x' = x\cos\phi + y\sin\phi,\ y' = -x\sin\phi + y\cos\phi,$
Przesunięcie $x' = x + t_x,\ y' = y + t_y.$
Zamiast punktów, obróćmy osie, zatem
$e'_x = e_x\cos\phi + e_y\sin\phi,\ e'_y = -e_x\sin\phi + e_y\cos\phi,$ dla wektorów jednostkowych $e_i$.
Analogicznie, dodając wektory przesunięcia:
$xe_x + ye_y = t_x e_x + t_y e_y + x'e'_x + y'e'_y$, dla wektorów jednostkowych $e_i$.

$$xe_x + ye_y = t_x e_x + t_y e_y + x'(e_x\cos\phi + e_y\sin\phi) + y'(-e_x\sin\phi + e_y\cos\phi)$$

Kompresując do jednej operacji

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} R' & T' \\ \vec{0}^T & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

# Obroty i przesunięcia

Obrót: $x' = x\cos\phi + y\sin\phi,\ y' = -x\sin\phi + y\cos\phi,$
Przesunięcie $x' = x + t_x,\ y' = y + t_y.$
Zamiast punktów, obróćmy osie, zatem
$e'_x = e_x\cos\phi + e_y\sin\phi,\ e'_y = -e_x\sin\phi + e_y\cos\phi,$ dla wektorów jednostkowych $e_i$.
Analogicznie, dodając wektory przesunięcia:
$xe_x + ye_y = t_x e_x + t_y e_y + x'e'_x + y'e'_y$, dla wektorów jednostkowych $e_i$.

$$xe_x + ye_y = t_x e_x + t_y e_y + x'(e_x\cos\phi + e_y\sin\phi) + y'(-e_x\sin\phi + e_y\cos\phi)$$

Zatem

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} R' & T' \\ \vec{0}^T & 1 \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# Idea

- dotąd operacje pozostawiały ten sam rozmiar obrazu wynikowego, zmieniając obiekty na nim
- co jeśli chcemy uzyskać obraz odpowiednio większy/mniejszy?
- co jeśli nie znamy rozdzielczości wynikowej, jaką chcemy osiągnąć?
- techniki bardzo istotne przy wykrywaniu i klasyfikacji obiektów (np. reprezentacja piramidowa dla detekcji twarzy)

# Zmiana rozdzielczości

- zwiększenie rozdzielczości obrazu pierwotnego (upscaling) to interpolowanie pewnych nieznanych wartości intensywności pikseli
- zmniejszenie rozdzielczości obrazu (downscaling) to decymacja, eliminujemy zbędne wartości intensywności

**Uwaga:** Zarówno up- jak i downscaling mogą wprowadzać nowe informacje do obrazu.

# Upscaling

Przypomnijmy sobie filtrowanie:

$$\hat{f}(x,y) = f * k = \sum_{i,j} f(i,j)k(x-i, y-j)$$

Wynikiem był obraz o tych samych rozmiarach (z dokładnością do paddingu), o zmienionych intensywnościach pikseli. Zmieńmy zatem rozdzielczość kernela.

$$\hat{f}(x,y) = f * k = \sum_{i,j} f(i,j)k(x-ri, y-rj), \ r > 1$$

# Downscaling

Koncepcja

- wygładź obraz (np. Gauss)
- zostaw co $r$-tą próbkę

$$\hat{f}(x,y) = f * k = \sum_{i,j} f(i,j)k(rx - i, ry - j), \;\; r > 1$$

# Przykłady interpolacji

- Nearest neighbor
- Bilinear (karty graficzne, często z MIPmap)
- Bicubic (programy graficzne)
- Lanczos (segmentacja, bannery; „generalizacja“ bicubic)

# Najbliższe sąsiedztwo





# Bilinear





# Bicubic



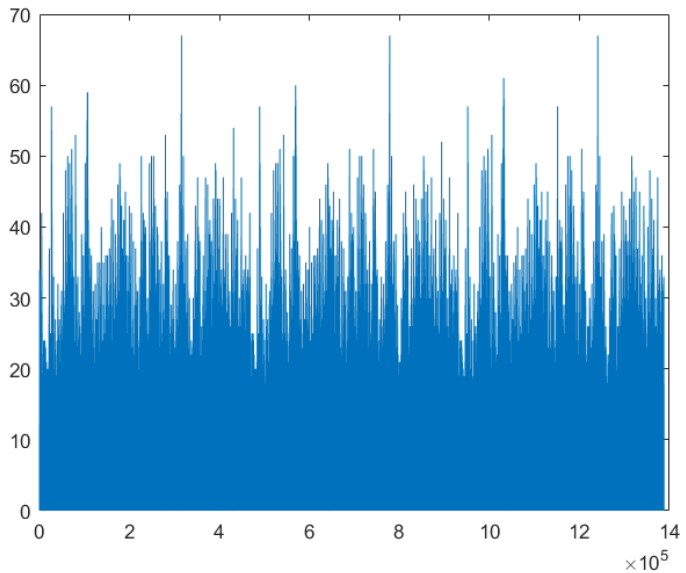


# Lanczos

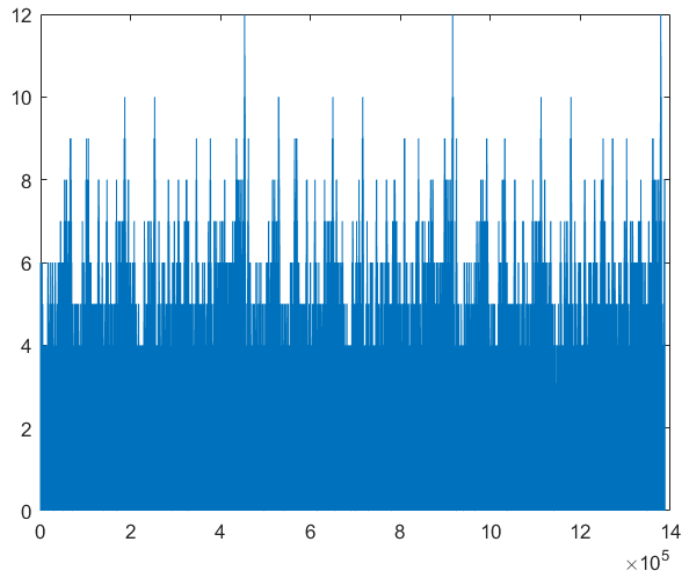




# Przykłady różnic




Nearest Neighbour vs Lanczos




Lanczos vs Cubic

# Bilinear

Znamy 4 sąsiednie wartości z $N_D(p_0)$.
Wartość piksela $f(x, y) = a_{11} f(p_{11}) + a_{12} f(p_{12}) + a_{21} f(p_{21}) + a_{22} f(p_{22})$,
dla $p_{ij} = (x_i, y_j)$ i współczynników $a_{ij}$:

$$\begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \end{bmatrix} = \left( \begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_1 & y_2 & x_1 y_2 \\ 1 & x_2 & y_1 & x_2 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \end{bmatrix}^{-1} \right)^T \begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix}$$

# Bicubic

Zakładamy, że możemy obliczyć pochodną wartości funkcji (lub dyskretne różnice) $f_x$, $f_y$, $f_{xy}$.
W przypadku obrazu potrzebujemy 8-sąsiedztwa.

$$f(x,y) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ y \\ y^2 \\ y^3 \end{bmatrix},$$

dla $a_{ij}$ wyliczanych z:

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) & f_y(0,0) & f_y(0,1) \\ f(1,0) & f(1,1) & f_y(1,0) & f_y(1,1) \\ f_x(0,0) & f_x(0,1) & f_{xy}(0,0) & f_{xy}(0,1) \\ f_x(1,0) & f_x(1,1) & f_{xy}(1,0) & f_{xy}(1,1) \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Alternatywnie – splot.

# Lanczos

Obcięta do okna funkcja sinc, pozwala na wygładzenie interpolowanego sygnału, wprowadzając *ringing effect*. Ustalamy wartość $a \in \mathbb{N}_+$ (najczęściej $a = 2 \vee a = 3$).
Kernel dla jednego wymiaru:

$$L(x; a) = \begin{cases} 1 & x = 0, \\ \frac{a \sin(\pi x) \sin(\frac{\pi x}{a})}{(\pi x)^2} & -a \leq x \leq a \wedge x \neq 0 \\ 0 & \text{w p.p.} \end{cases}$$

Dla obrazów: $L(x, y; a) = L(x; a)L(y; a)$

# Reprezentacja piramidowa – MIP map

- przygotowujemy i zapamiętujemy serię obrazów o różnych rozdzielczościach (zwyczajowo, każda następna reprezentacja ma dwa razy krótsze boki)
- dzięki zwiększeniu kosztu pamięciowego zyskujemy czas i antialiasing
- stosowany dla tekstur 3D oraz przy metodzie słownikowej

# Klasyfikacja przekształceń

euklidesowe:

$$\begin{bmatrix} \cos\phi & -\sin\phi & t_x \\ \sin\phi & \cos\phi & t_y \\ 0 & 0 & 1 \end{bmatrix},$$

zachowują: długości, kąty, odcinki równoległe, linie proste

podobieństwo:

$$\begin{bmatrix} s\cdot\cos\phi & -s\cdot\sin\phi & t_x \\ s\cdot\sin\phi & s\cdot\cos\phi & t_y \\ 0 & 0 & 1 \end{bmatrix},$$

zachowują: kąty, odcinki równoległe, linie proste

# Klasyfikacja przekształceń

afiniczne:

$$\begin{bmatrix} A & B & C \\ D & E & F \\ 0 & 0 & 1 \end{bmatrix},$$

zachowują: odcinki równoległe, linie proste

projekcyjne:

$$\begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix},$$

zachowują: linie proste

# Klasyfikacja przekształceń – przykład











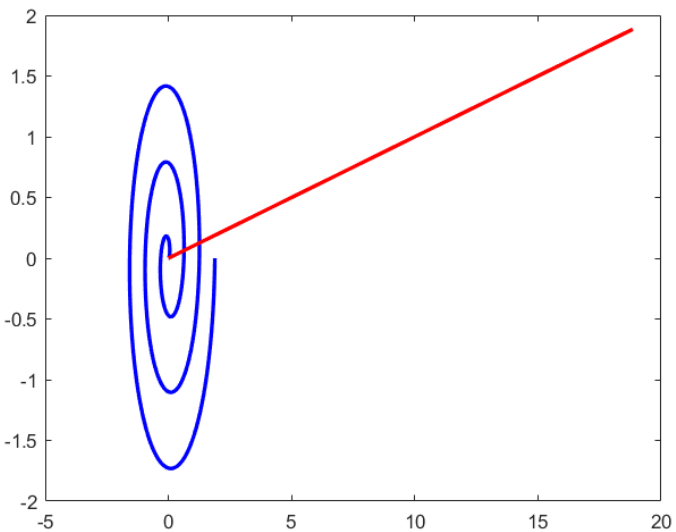


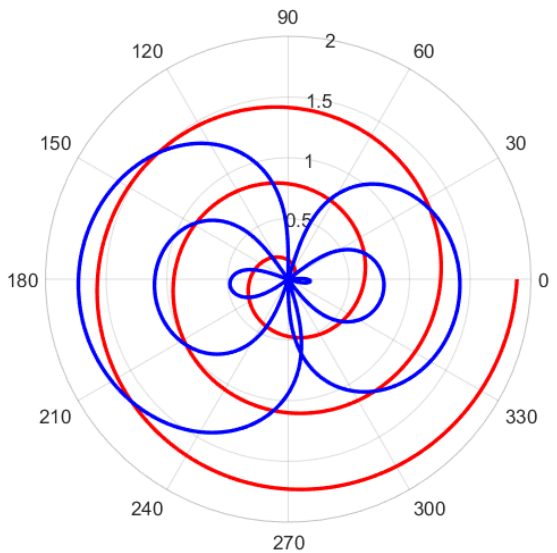
# Współrzędne polarne

- dwuwymiarowa reprezentacja punktów przez $r$–promień oraz $\theta$ – kąt
- zmiana współrzędnych:

$$r = \sqrt{x^2 + y^2},\ \theta = \mathrm{tg}^{-1}(\frac{y}{x})$$

$$x = r\cos\theta,\ y = r\sin\theta$$

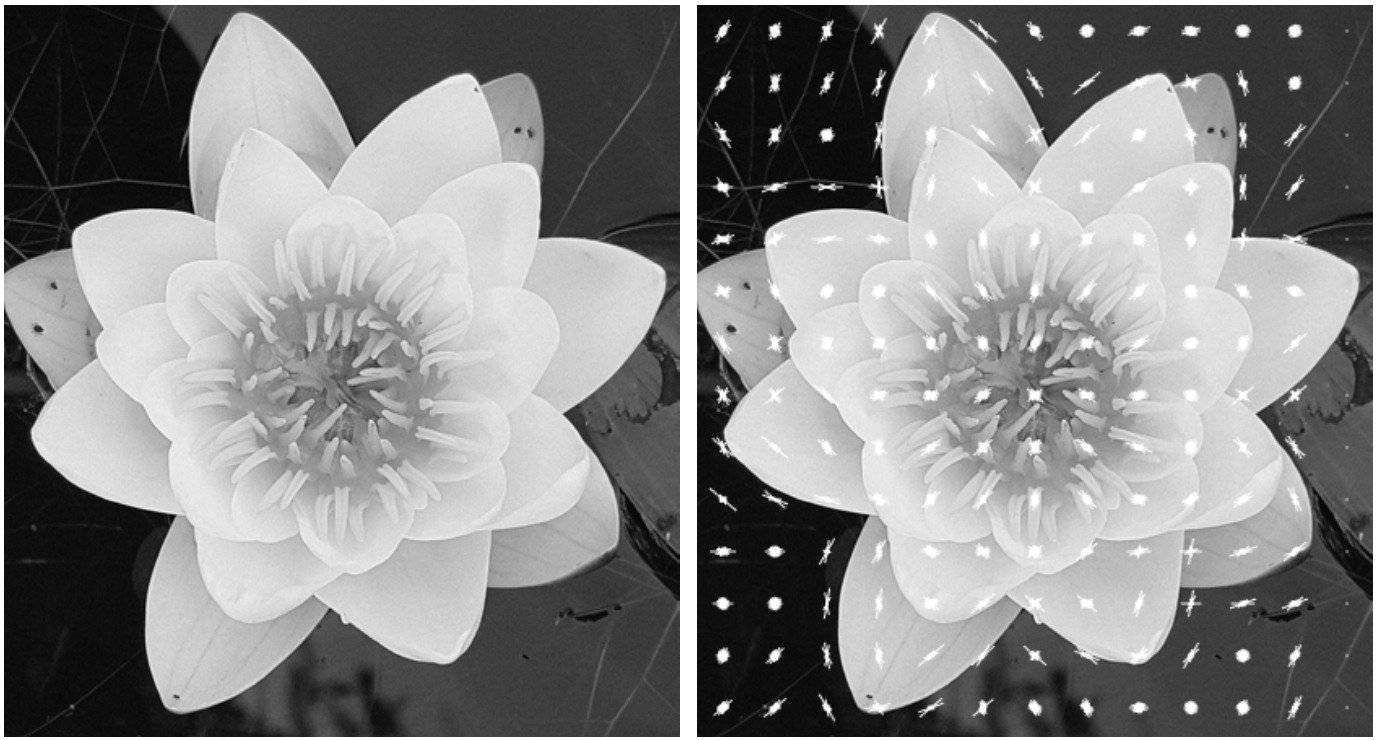


# Histogram zorientowanych gradientów

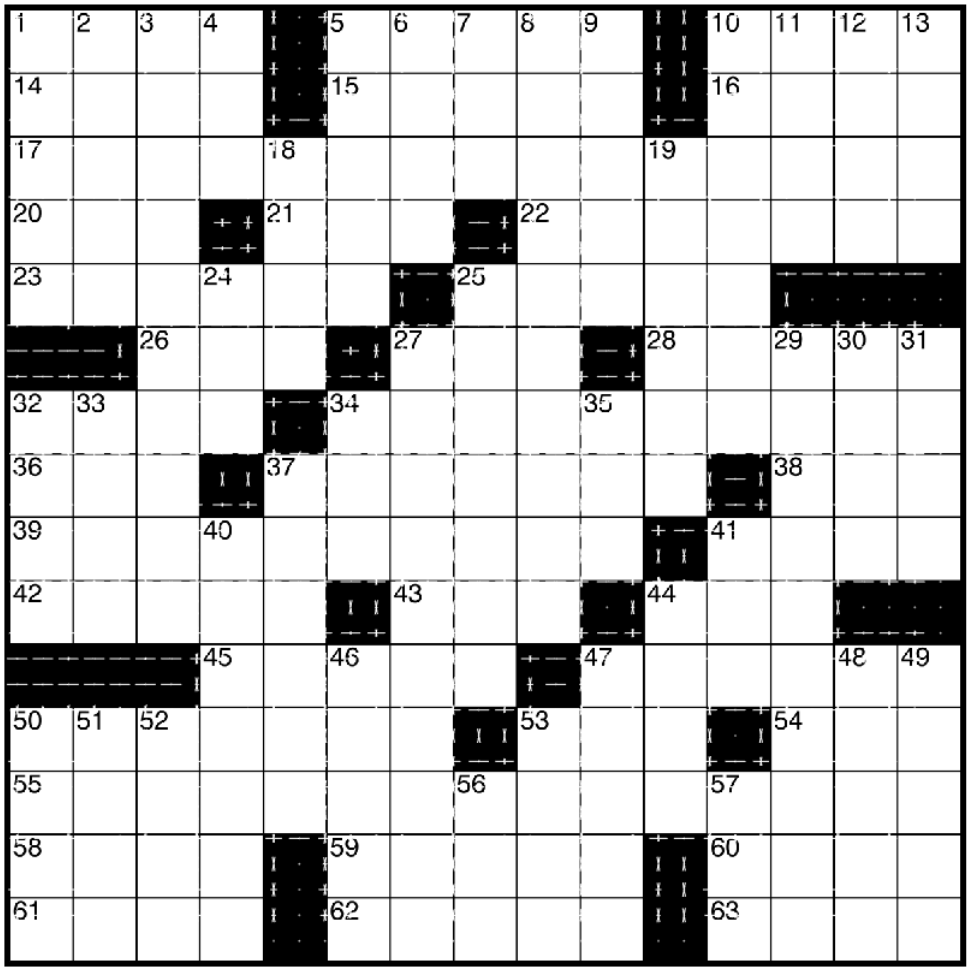
- szukamy deskryptora obrazu pomagającego znaleźć kształt obiektu bez skomplikowanego preprocessingu
- wyliczamy gradienty $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ oraz $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ – porównujemy intensywność piksela z sąsiadami
- grupujemy piksele w zachodzące na siebie komórki, w każdej komórce ustawiamy kierunek gradientu na najczęściej w niej występujący
- sprawdzamy dwa główne kierunki gradientu, które są ze sobą porównywane pod kątem dominacji – równoległy do osi oraz okrężny

# HOG – przykład



```
[fv,hv]=extractHOGFeatures(I);
[fv2,hv2]=extractHOGFeatures(I,'CellSize', [64 64]);
```

# Jednak nie zawsze jest super



# Idea

- chcemy wykrywać proste kształty: linie, okręgi, elipsy, ...
- kształty mogą być niedokładne – brakujące piksele, różnica intensywności (grayscale), zniekształcone (podobieństwo).
- chcemy reprezentować punkty należące do szukanego kształtu za pomocą wybranego punktu (ew. punktów) – im więcej punktów należących do kształtu, tym większa wartość w wybranym akumulatorze
- po zakończeniu operacji, porównujemy wartości akumulatorów z ustalonymi progami

# HoughLines

Prosta $y = ax + b$, może być jednoznacznie reprezentowana przez parę (a więc punkt) $(a, b)$

- dla każdego napotkanego punktu na prostej, zwiększamy wartość akumulatora w punkcie $(a, b)$
- po zakończonej operacji, dla każdej pary $(a, b)$ sprawdzamy wartość akumulatora, jeśli przekracza ustalony próg – na oryginalnym obrazie istnieje prosta $y = ax + b$

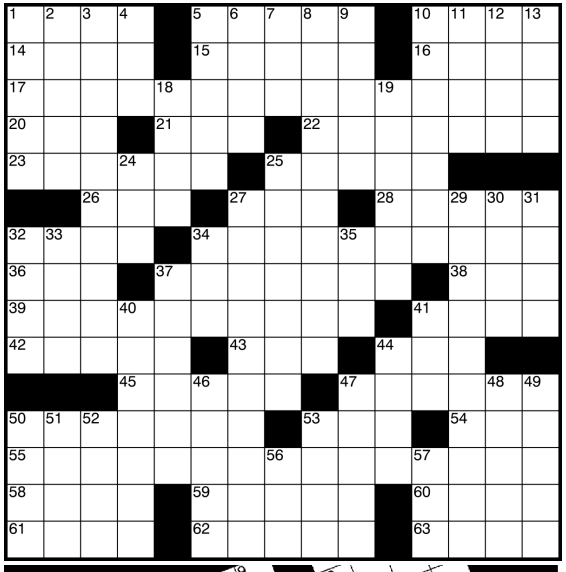
# HoughLines

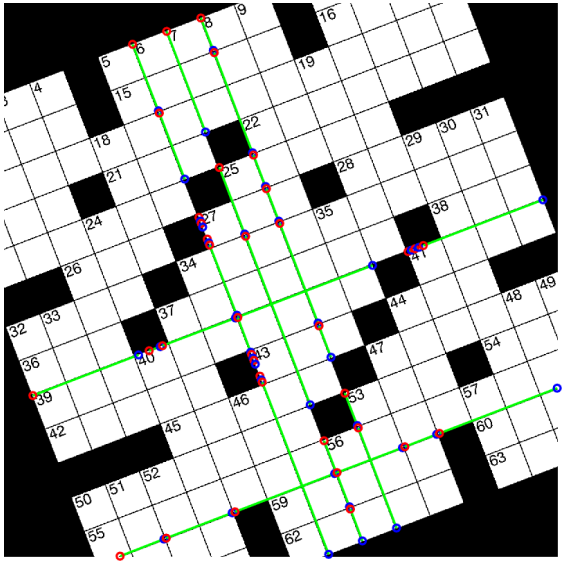
Prosta $y = ax + b$, może być jednoznacznie reprezentowana przez parę (a więc punkt) $(a, b)$

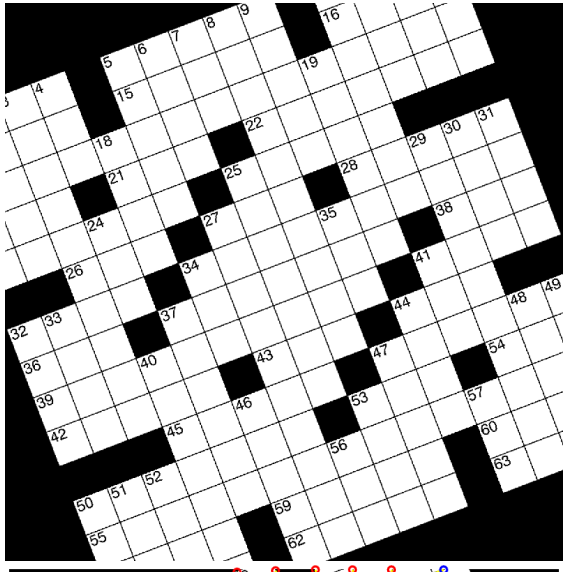
- dla każdego napotkanego punktu na prostej, zwiększamy wartość akumulatora w punkcie $(a, b)$
- po zakończonej operacji, dla każdej pary $(a, b)$ sprawdzamy wartość akumulatora, jeśli przekracza ustalony próg – na oryginalnym obrazie istnieje prosta $y = ax + b$

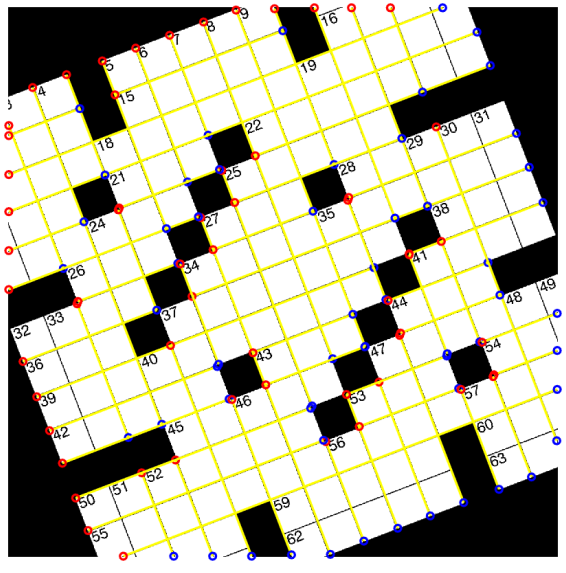
. . . ale pojawia się problem z liniami pionowymi.
Przyjmijmy zatem inną reprezentację: $r = x \cos\theta + y \sin\theta$. Możemy teraz reprezentować linię w oryginalnej przestrzeni przez parę $(\theta, r)$ i powtórzyć wcześniejszy pomysł.
**Uwaga:** Alternatywnie zamiast reprezentacji polarnej $(\theta, r)$ możemy parametryzować linię jako punkt na sferze jednostkowej.

# HoughLines – przykład

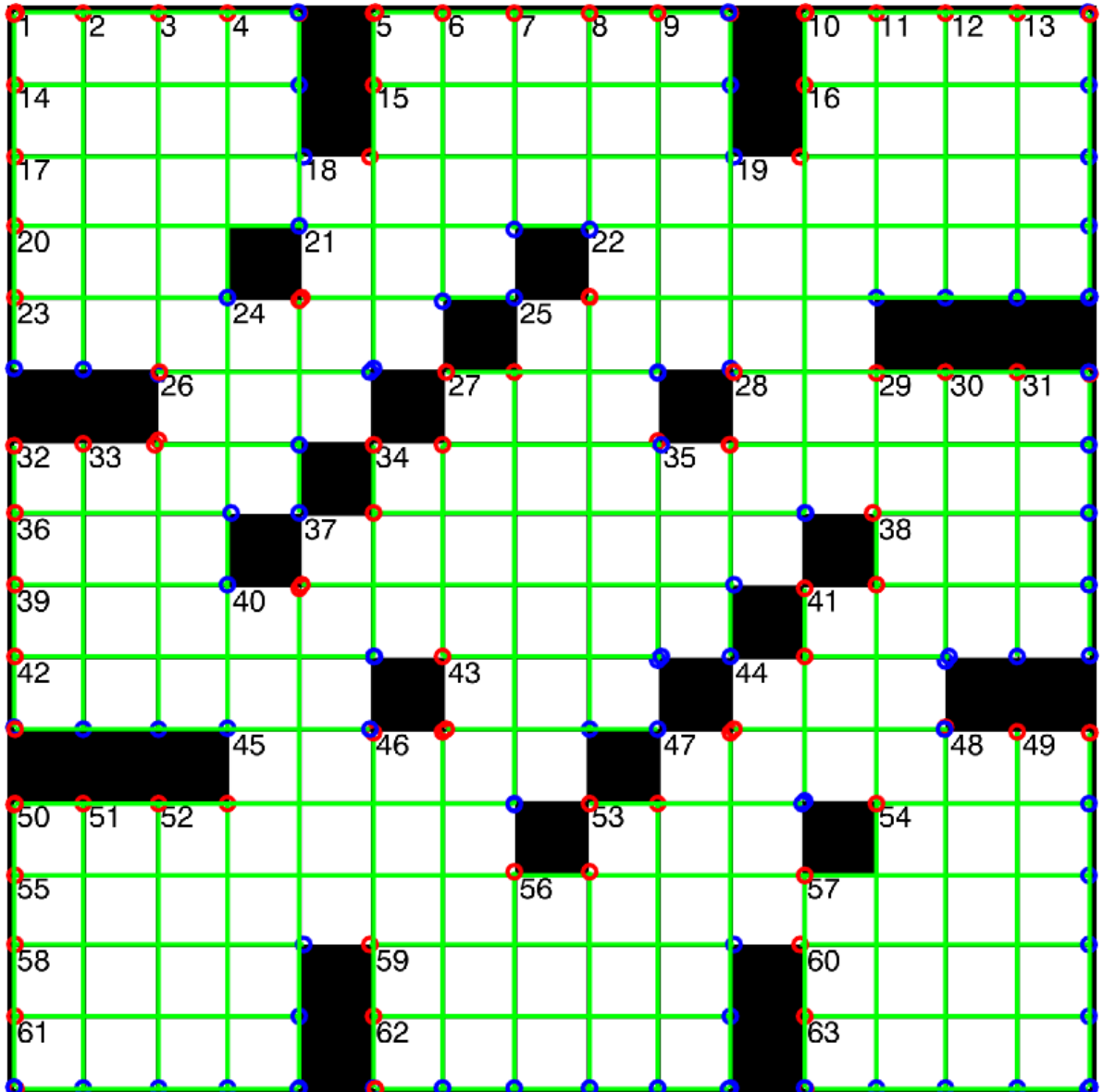








# HoughLines – przykład



# CHT

- zamiast linii możemy wykrywać okręgi (wyszukiwać okręgu lub szukać okręgów o ustalonym promieniu)
- okrąg określony jest równaniem $(x-a)^2+(y-b)^2=r^2$, zatem analogicznie do wykrywania linii, możemy reprezentować go trójką $(a,b,r)$
- punkty okręgu będą leżeć na powierzchni stożka
- wyszukujemy okręgów w nowej przestrzeni, dla każdego punktu przyjmujemy wartość akumulatora równa liczbie okręgów (rzutów stożka na 2D) przecinających się w danym punkcie
- ustalmy wartość progową, punkty które przekroczyły próg są środkami okręgów na oryginalnym obrazie

# CHT – przykłady



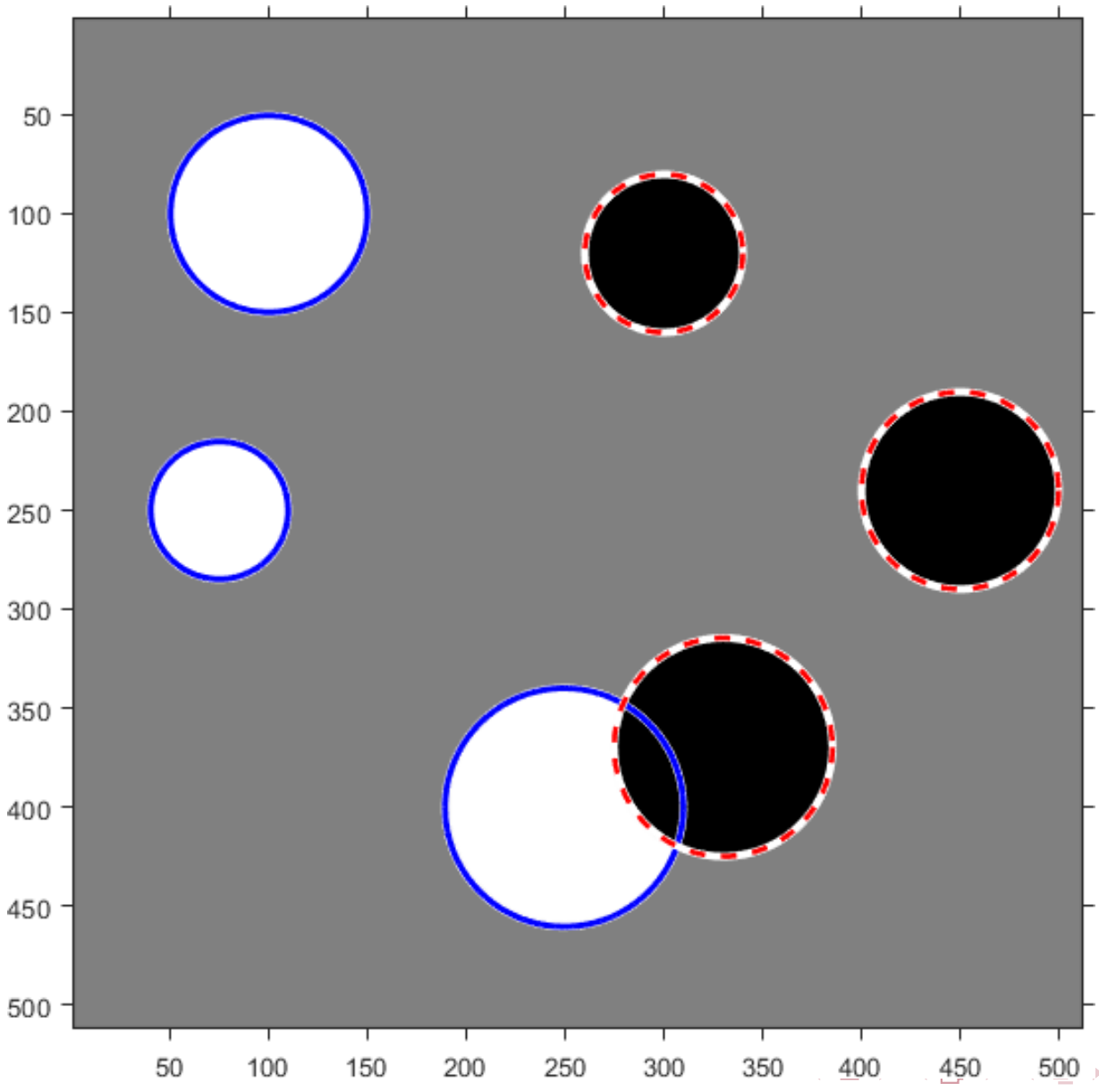
# CHT – przykłady



# CHT – przykłady



# CHT – przykłady



# Transformata Hougha – krzywe

- problem z wykrywaniem kół lub prostych, gdy są zniekształcone
- co z innymi obiektami (krzywe, elipsy, hiperbole) ?
- idea: szukaj wszystkich krzywych, głosuj za pomocą akumulatorów, wybierz najlepszy wynik
- problem: redundancja zliczania (jeden punkt może należeć do kilku elips), większa dokładność wymaga większej rozdzielczości przestrzeni Hougha
- rozwiązanie: randomizacja

# RHT – elipsy

- Reprezentacja elipsy:
  $a(x-p)^2 + 2b(x-p)(y-q) + c(y-q)^2 = 1, ac > b^2$
- Jednak wystarczy znać 3 punkty i styczne w tych punktach
- Losujemy 3 punkty $p_1, p_2, p_3$, znajdujemy proste łączące wybrane punkty, wyznaczamy styczne
- Wyznaczamy punkty przecięcia stycznych $P$ i $Q$
- Wyznaczamy odcinki między $P$ i środkiem prostej łączącej dwa pierwsze punkty oraz między $Q$ i środkiem prostej łączącej $p_2$i $p_3$.
- Punkt przecięcia odcinków będzie środkiem elipsy, podstawiamy go oraz punkty $p_1, p_2, p_3$ do równania elipsy

Gdzie losowość: wybór punktów, pamiętanie akumulatorów dla kilku najlepszych kandydatów.

# Uogólniona transformata Hougha

Nawet z pomocą randomizacji, jesteśmy ograniczeni do wyszukiwania obiektów, które posiadają znane równanie parametryczne.
Rozwiązanie: wspomaganie się słownikiem (Ballard).

- wybieramy reprezentację obiektu (zestaw krawędzi, własności gradientu, reprezentacja piramidowa)
- wybieramy punkt odniesienia ('środek') szukanego obiektu, np. środek masy lub środek geometryczny

# Uogólniona transformata Hougha

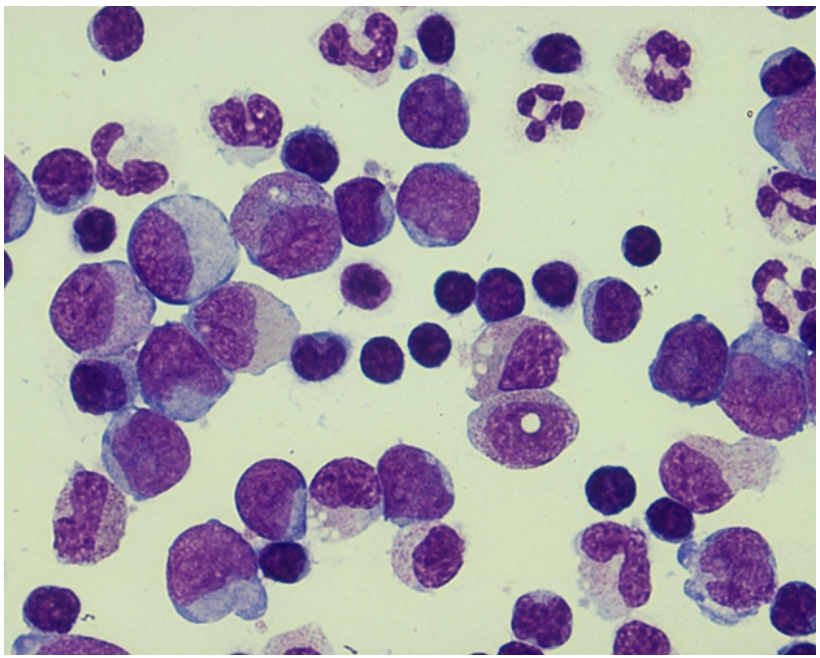
- Budujemy R-Tabelę: dla każdego punktu na brzegu obiektu obliczamy gradient oraz $r$ – odległość od środka.
  Zapisujemy w tabeli wartość $r$ jako funkcję gradientu $\phi$. Uwaga: dla jednego kierunku gradientu możemy mieć wiele $r$ (w celu rozróżnienia pamiętamy RDF lub kąt między wektorem a gradientem).
  *Przykład wyboru parametrów*:
  $r = \sqrt{(x_c - x_b)^2 + (y_c - y_b)^2}$, $\beta = \mathrm{tg}^{-1} \frac{y_b - y_c}{x_b - x_c}$
- R-Tabela w pełni opisuje kształt, co więcej, jest odwracalna, więc pomoże nam zlokalizować wystąpienia obiektu (środek i kąty)
- Czyścimy tablę akumulatorów $A$ (indeksujemy ją po punktach środkowych), dla każdego piksela krawędzi $(x, y)$ szukamy wpisu w R-Tabeli, który jest najbliższy kierunkowi gradientu w $(x, y)$:
  dla $(r, \beta)$ z tabeli wyliczamy $x_c = x + r\cos\beta$, $y_c = y + r\sin\beta$ i inkrementujemy wartość $A(x_c, y_c)$.
- Po przejściu wszystkich pikseli, wybieramy 'najlepszy środek'.

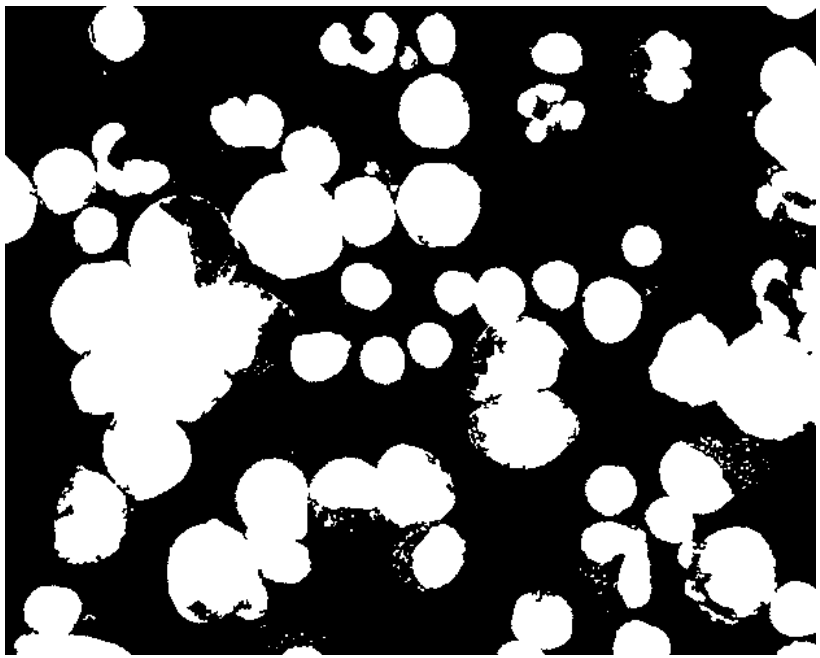
# Uogólniona transformata Hougha

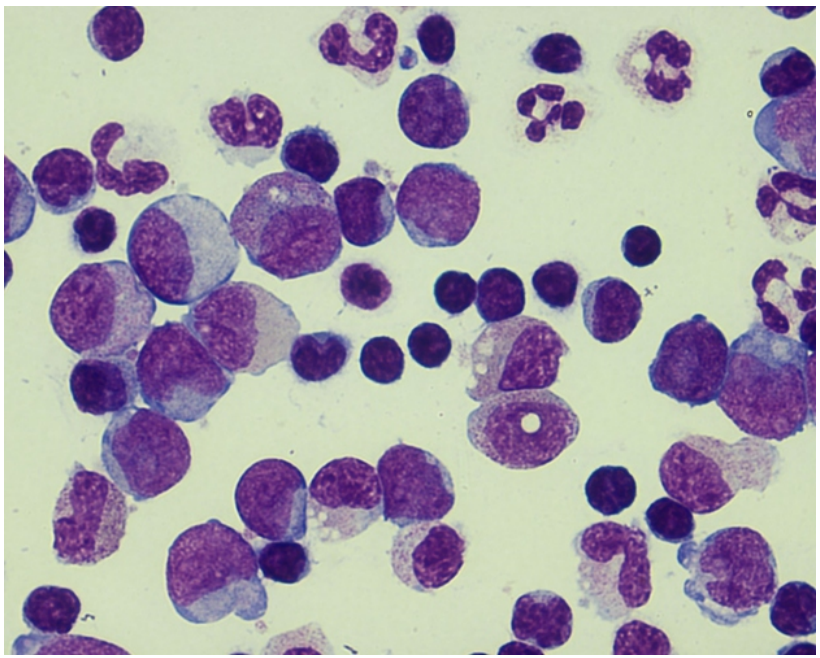
- Wada: wrażliwość na obroty i reskalowanie
- Rozwiązanie: $x_c = x + sr\cos(\beta + \alpha)$, $y_c = y + sr\sin(\beta + \alpha)$, mamy akumulator 4 wymiarowy $A(x_c, y_c, s, \alpha)$

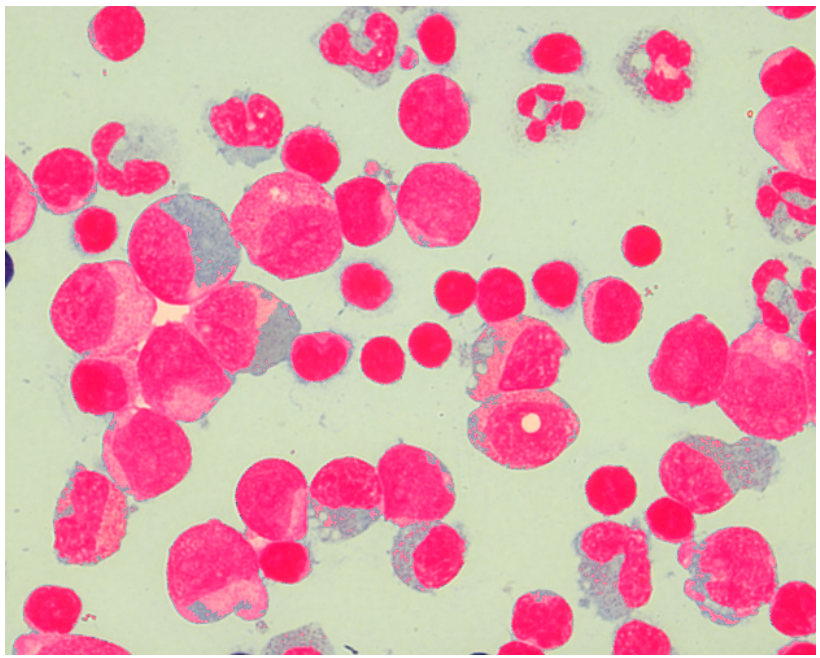


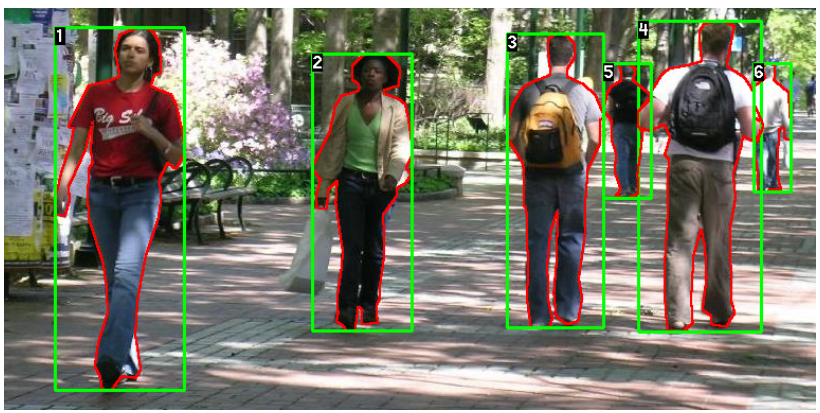


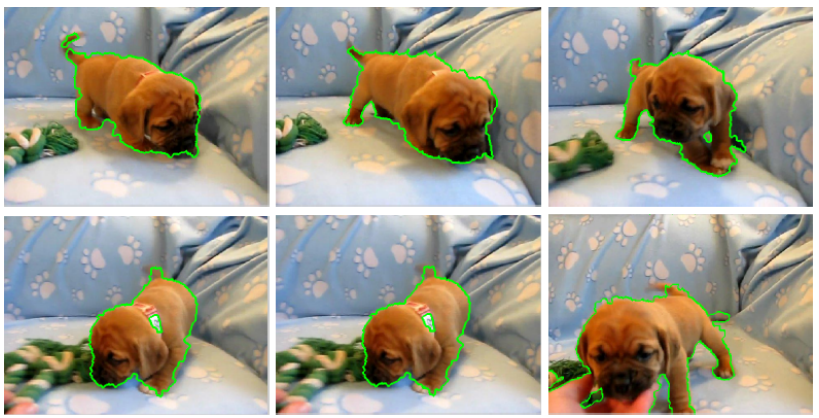


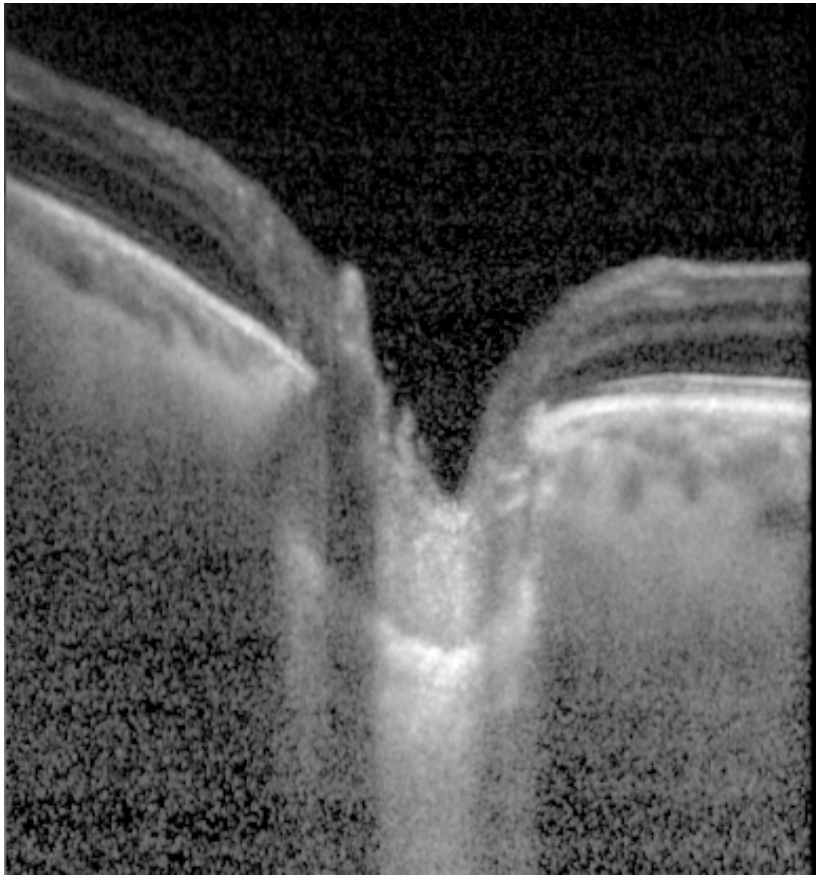


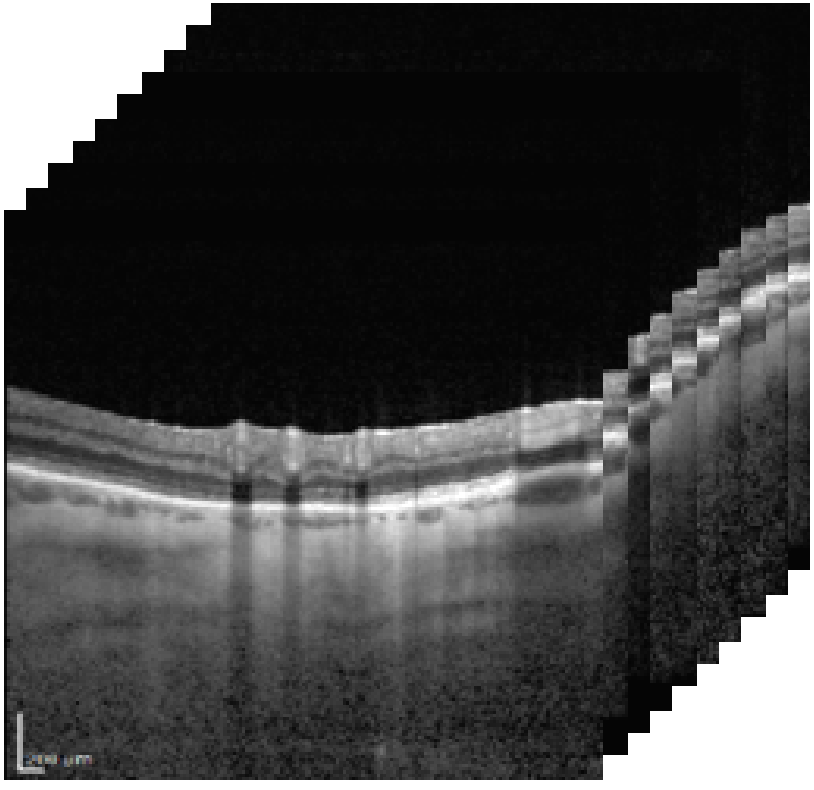


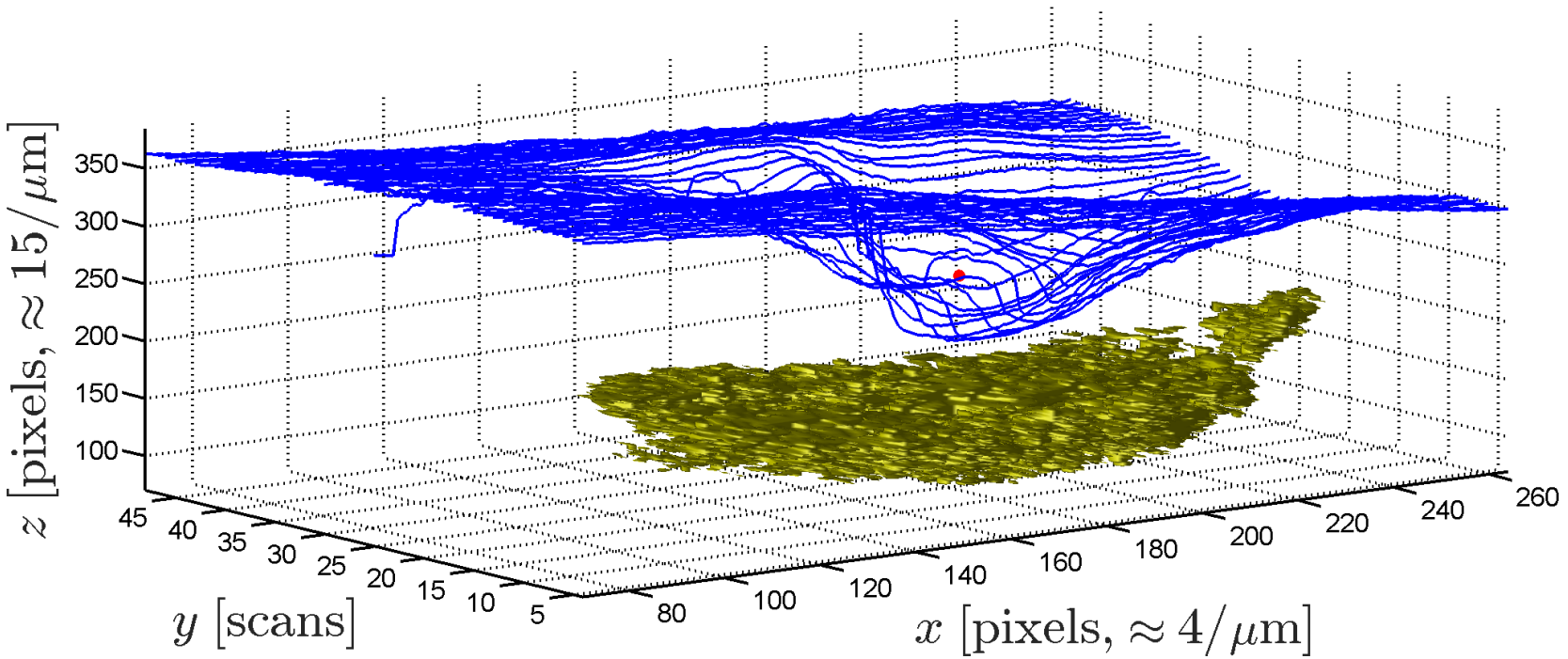



1
2
3
4
5
6







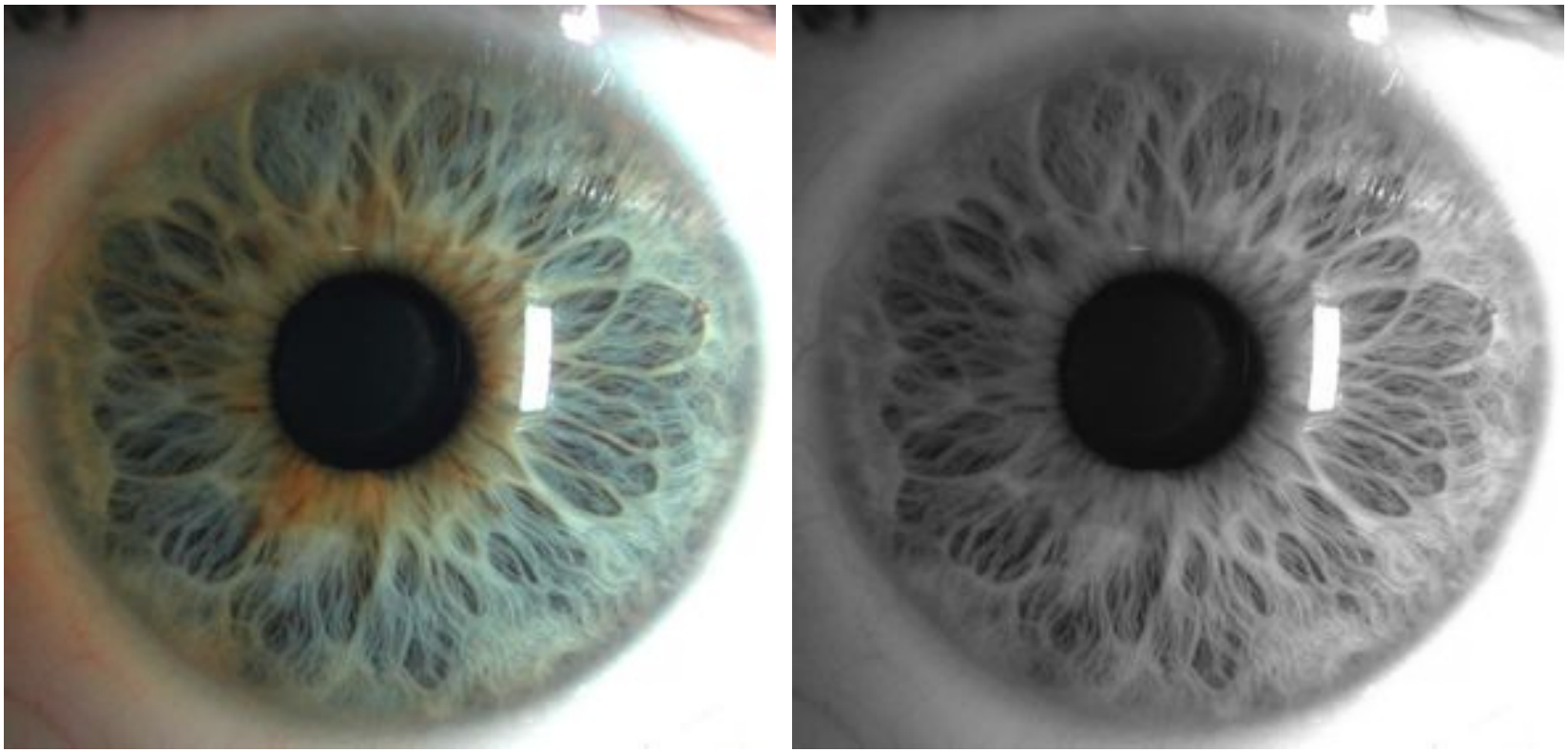

$z$ [pixels, $\approx 15/\mu$m]
350
300
250
200
150
100
45
40
35
30
25
20
15
10
5
$y$ [scans]
80
100
120
140
160
180
200
220
240
260
$x$ [pixels, $\approx 4/\mu$m]

- wyodrębnienie charakterystycznych obiektów z obrazu
- wyszukanie konkretnego obiektu vs znalezienie obiektu i jego klasyfikacja
- wykrywanie anomalii
- pomiary biomedyczne i biometryczne
- śledzenie ruchu
- computer vision w sterowaniu pojazdami

# Metody poznane do tej pory

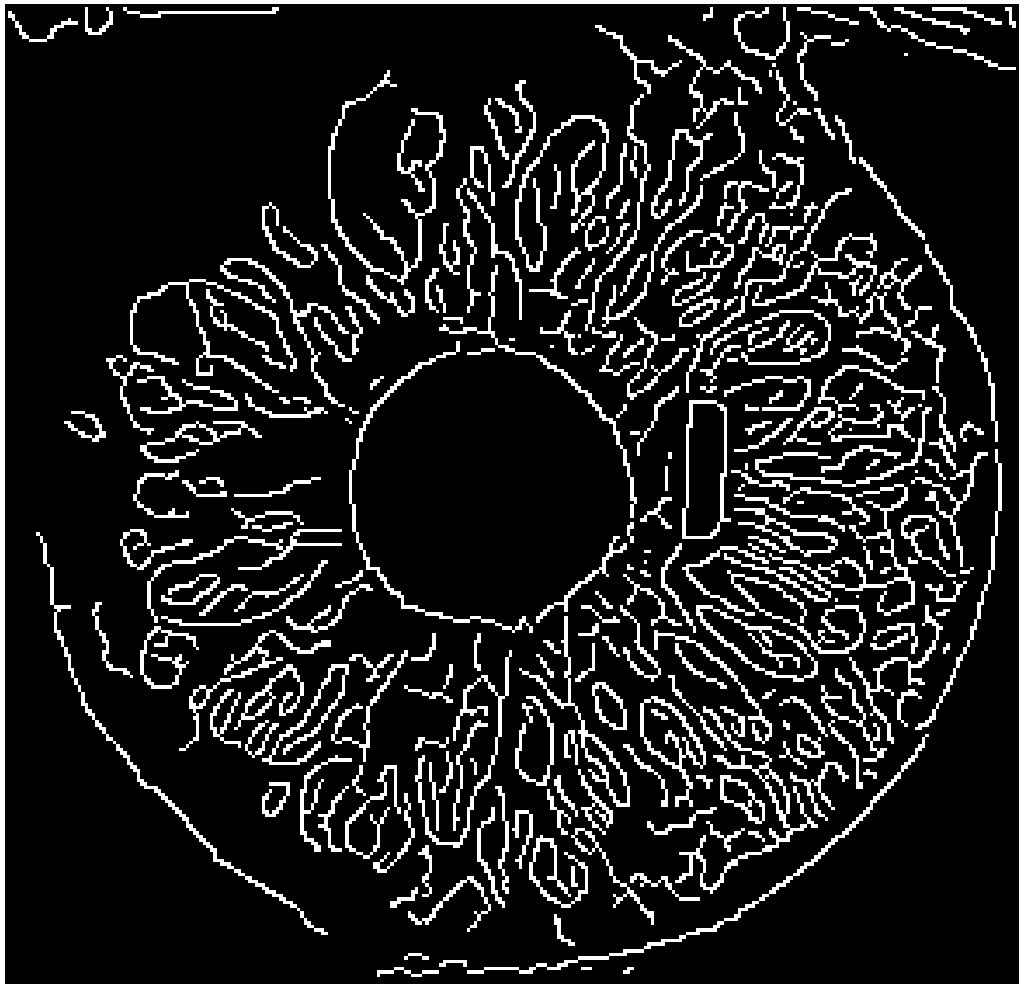
- wyznaczanie krawędzi (np. Canny)
- konturowanie morfologiczne
- transformata Hougha
- progowanie Otsu

# Przykłady



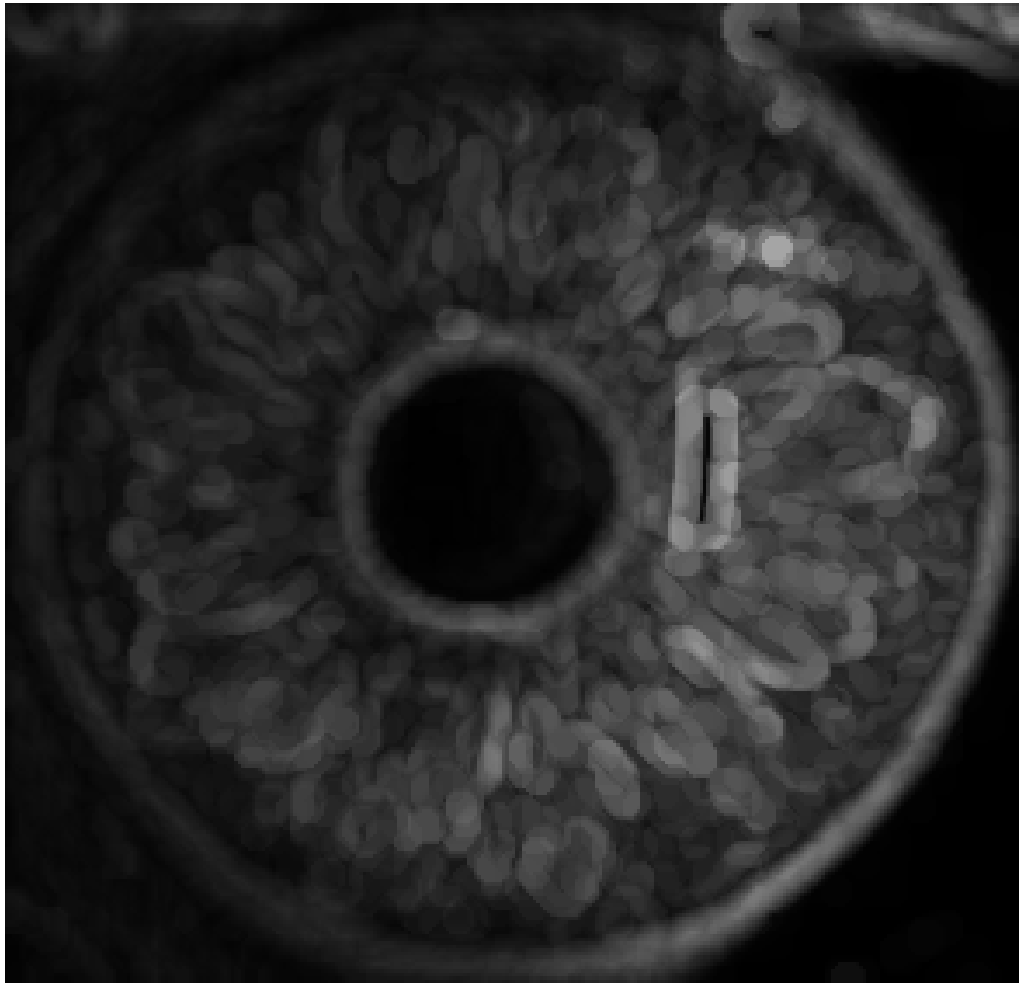
Zdjęcie i skala szarości

# Przykłady



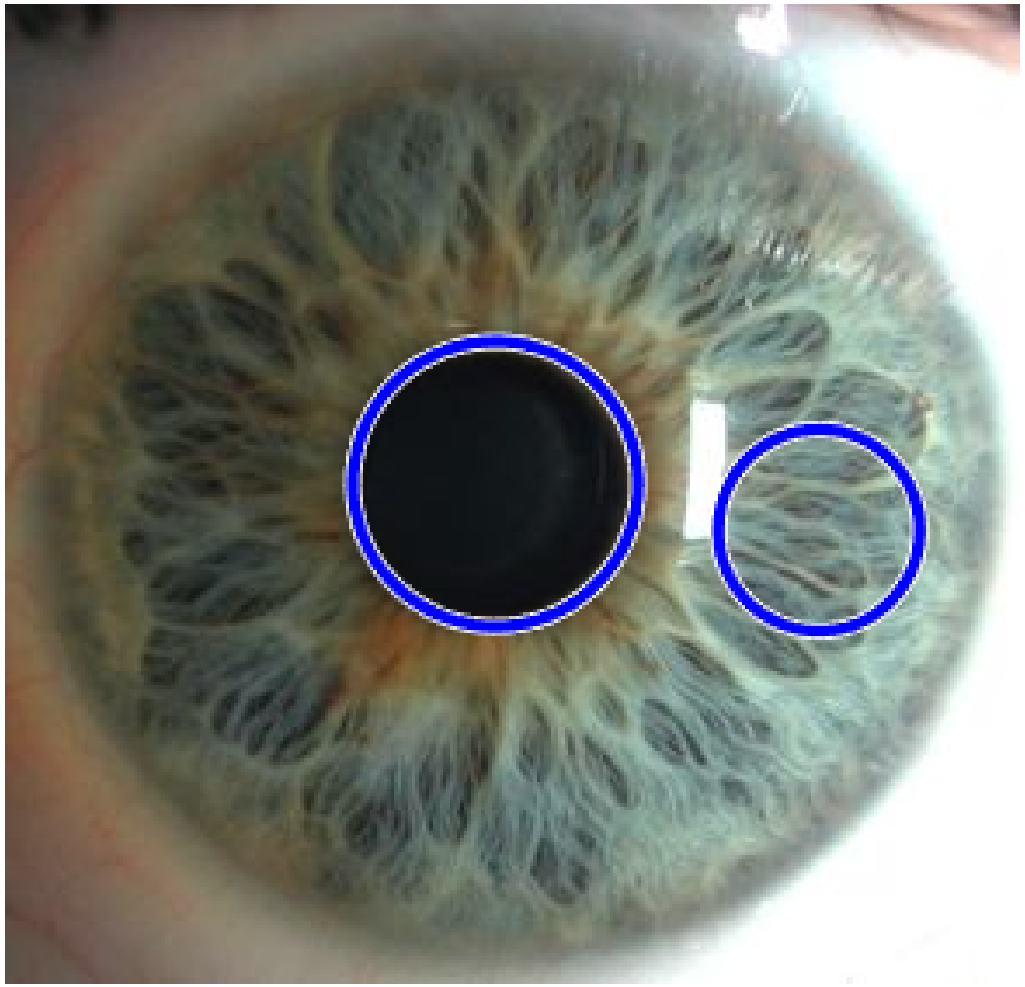
Krawędzie

# Przykłady



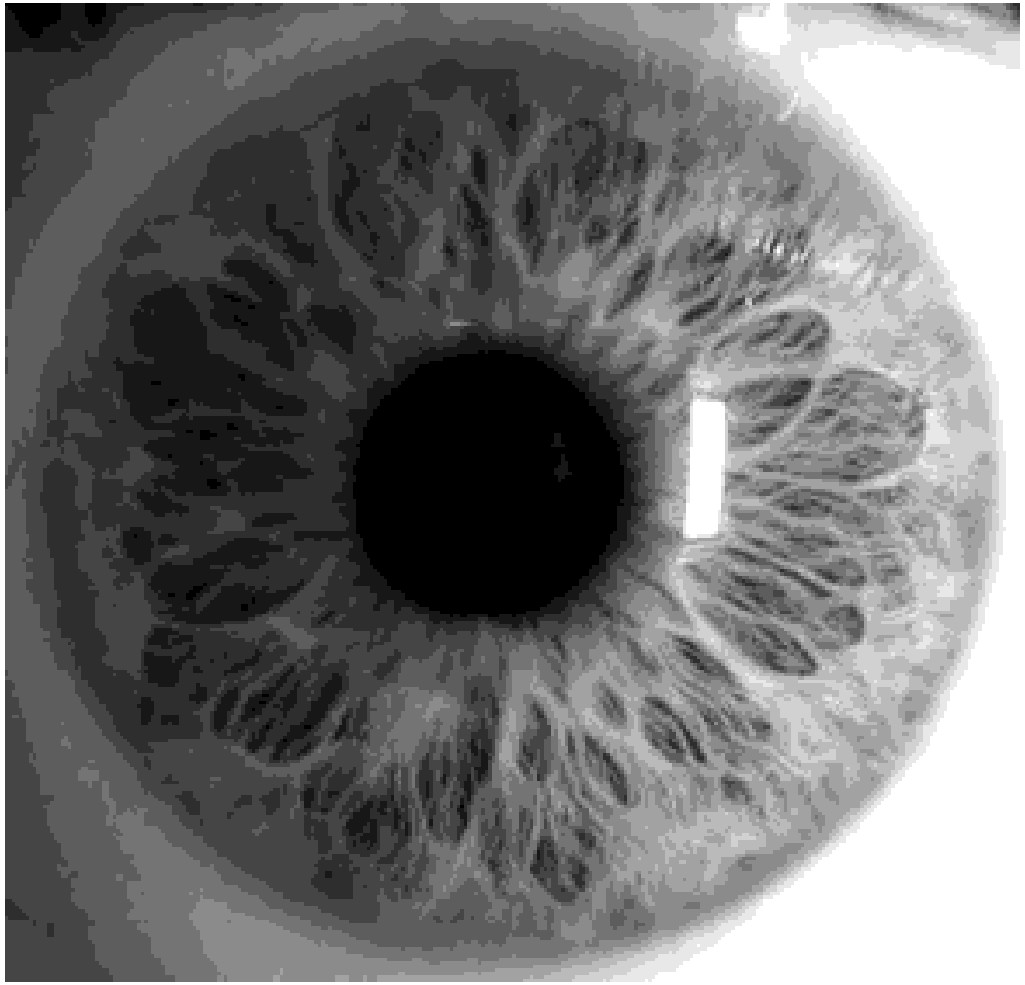
Kontury morfologiczne

# Przykłady



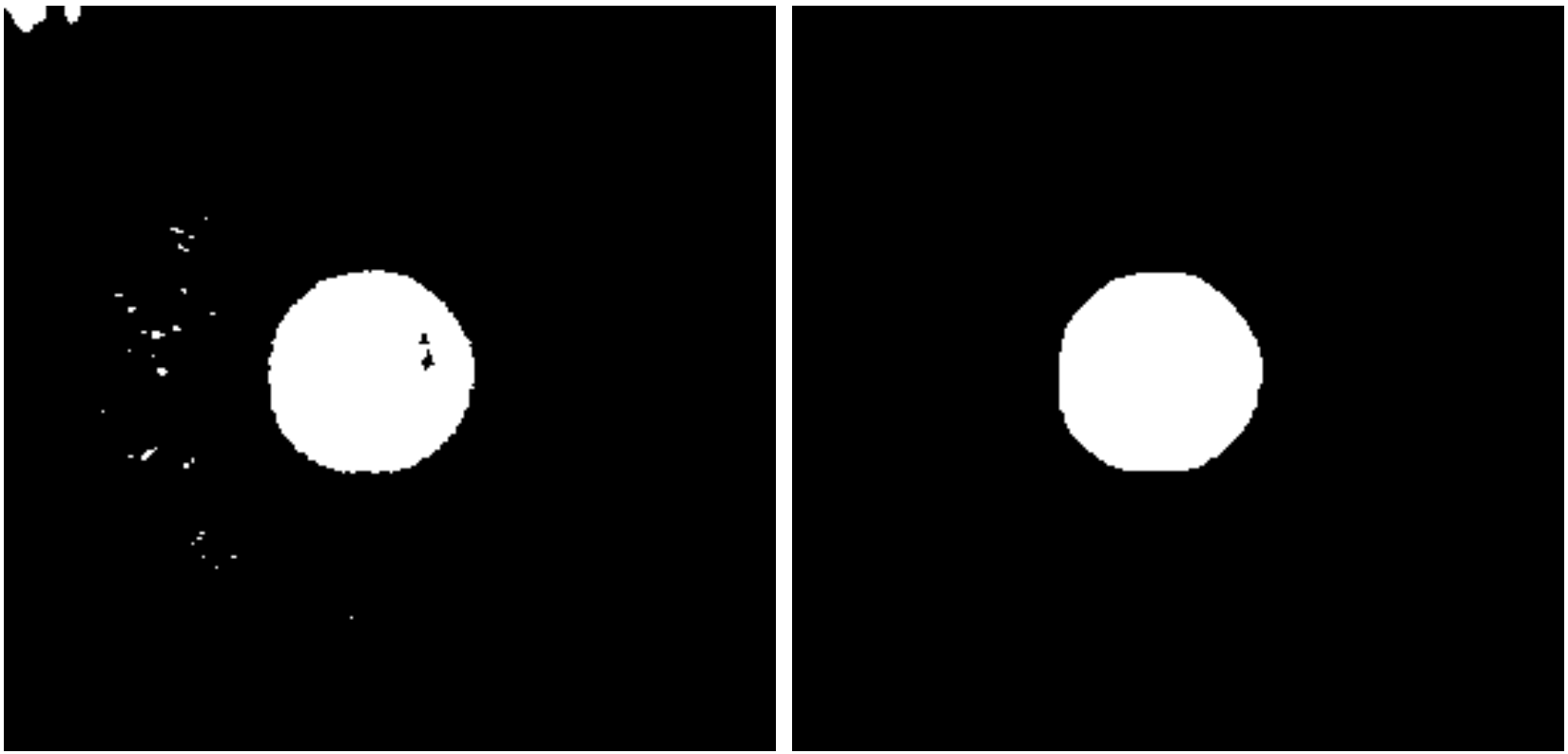
Hough

# Przykłady



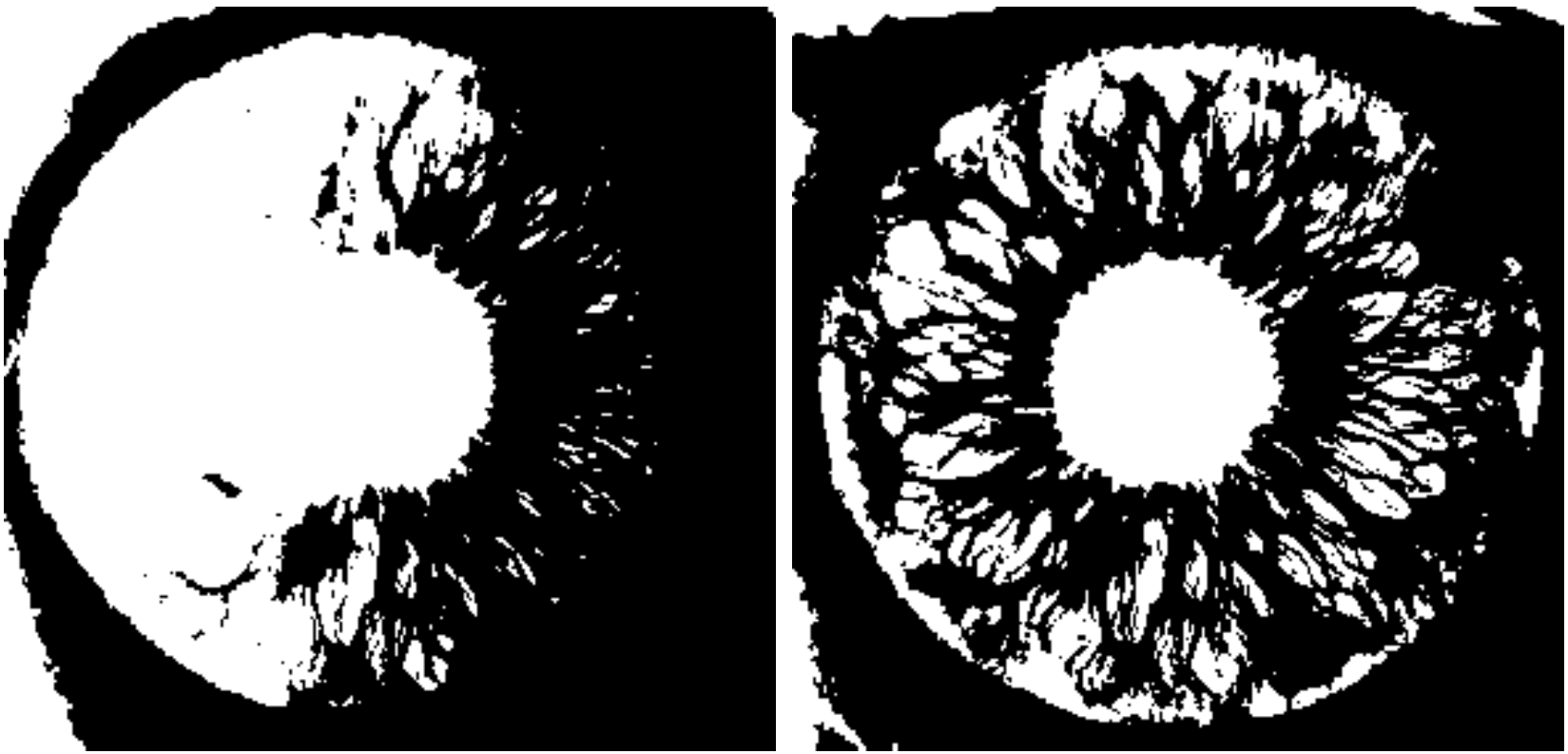
Otsu

# Przykłady



Otsu – źrenica

# Przykłady



Otsu – tęczówka?

# Retina – Retinex

- Problem: nierówno oświetlone zdjęcie (nieliniowa zmiana jasności)
- imitacja przetwarzania obrazu przez ludzkie oko
- podwójna reprezentacja obrazu: drobnokomórkowa (*parvocellular*) – statyczne detale, wielkokomórkowa (*magnocellular*) – informacje dot. dynamizmu
- celem algorytmu jest estymacja funkcji odbicia $r(x, y)$

J. Hérault, B. Durette: *Modeling Visual Perception for Image Processing*,
J.-M. Morel, A. B. Petro, C. Sbert: *A PDE formalization of retinex theory*

# Retina – Retinex

- Wykonujemy losowo błądzenie z ustalonych pikseli (wybieramy $N$ pikseli źródłowych, z każdego wyprowadzamy $J$ ścieżek o długości $K_j$)
  Wyliczamy

$$r(p) = \frac{1}{J} \sum_{j=1}^{J} \sum_{k=1}^{K_j} \lambda(I(p_k^{(j)}) - I(p_{k+1}^{(j)})) \ ,$$

dla $\lambda = \mathbb{1}_{|x|>\tau}|x|$ i ustalonego progu $\tau$.
Rozwiązanie równania różniczkowego:

$$\begin{cases} \frac{\partial r}{\partial N} = 0 & \text{dla } p \text{ na krawędzi obrazu} \\ -\Delta r(p) = F(p) & \text{w p.p.} \end{cases}$$

$F(p) = \lambda(I(p) - I(p_{(+1,0)})) + \lambda(I(p) - I(p_{(-1,0)})) + \lambda(I(p) - I(p_{(0,+1)})) + \lambda(I(p) - I(p_{(0,-1)}))$.
Rozwiązujemy w dziedzinie częstotliwości,

$$I = \mathcal{F}^{-1} \left\{ \frac{\mathcal{F}\{F(p)\}}{2\left(\cos\left(2\pi \frac{r}{H}\right) + \cos\left(2\pi \frac{s}{W}\right) - 2\right)} \right\}$$

# Retina – Retinex

Operacje:

filtrowanie górnoprzepustowe – wyrównuje spektrum $\frac{1}{f}$

nieliniowy filtr częstotliwościowy – stopniowanie przetwarzania od zgrubnego do szczegółowego

multipleksing – zwielokrotnianie długości fali (przesunięcie kolorów), ze względu na dekodowanie kolorów w korze, siatkówka odczytuje wyłącznie luminancję

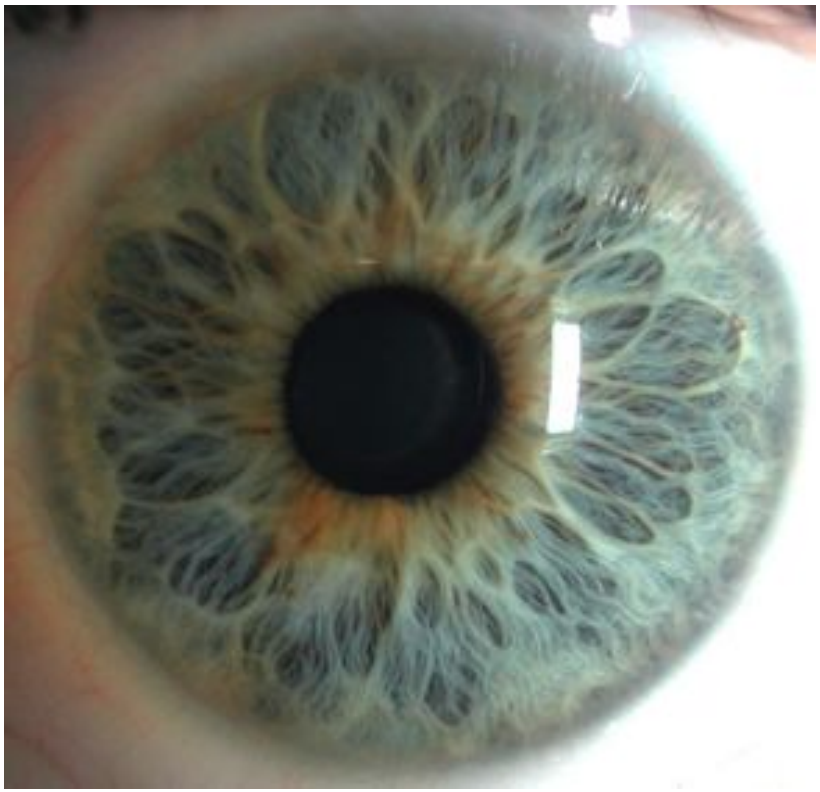
nieregularne próbkowanie – antialiasing

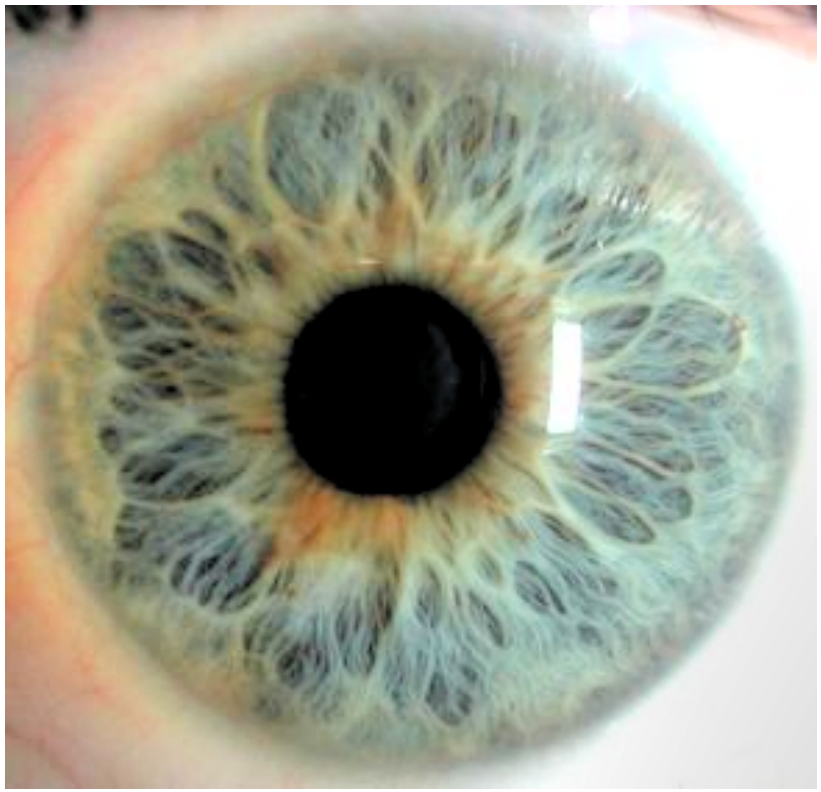
próbkowanie zależne od położenia – odzwierciedlenie nierównomiernie rozłożonych receptorów – rozpoznanie kontekstu, estymacja gradientu, uniezależnienie się od przybliżeń

lokalne wyrównanie histogramu – kompresja fotoreceptorów

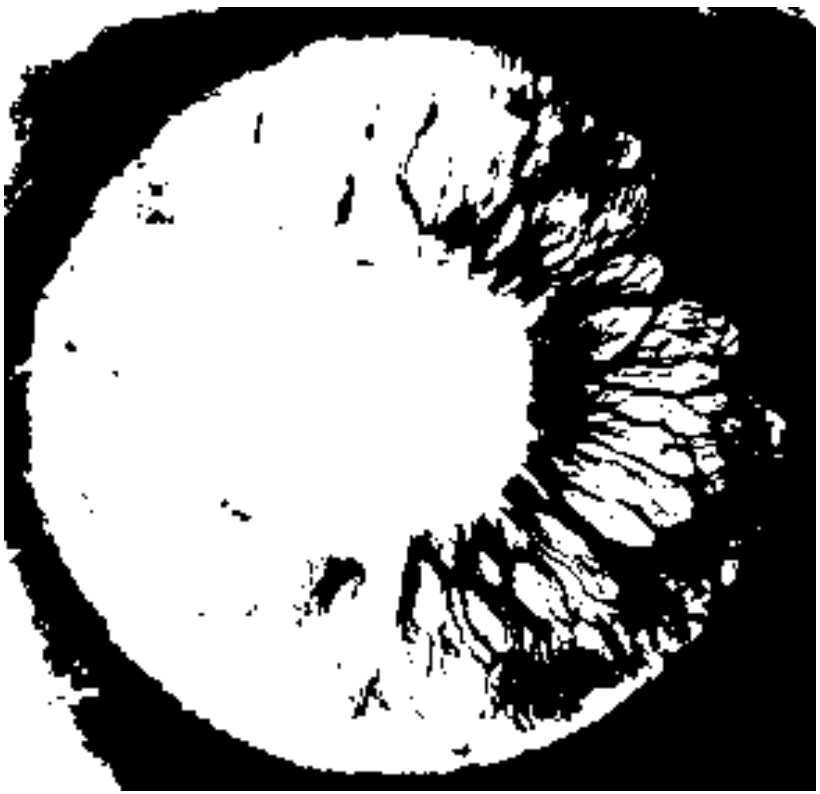
lokalne wyrównanie kontrastu – kompresja ganglionów

# Retina – Retinex



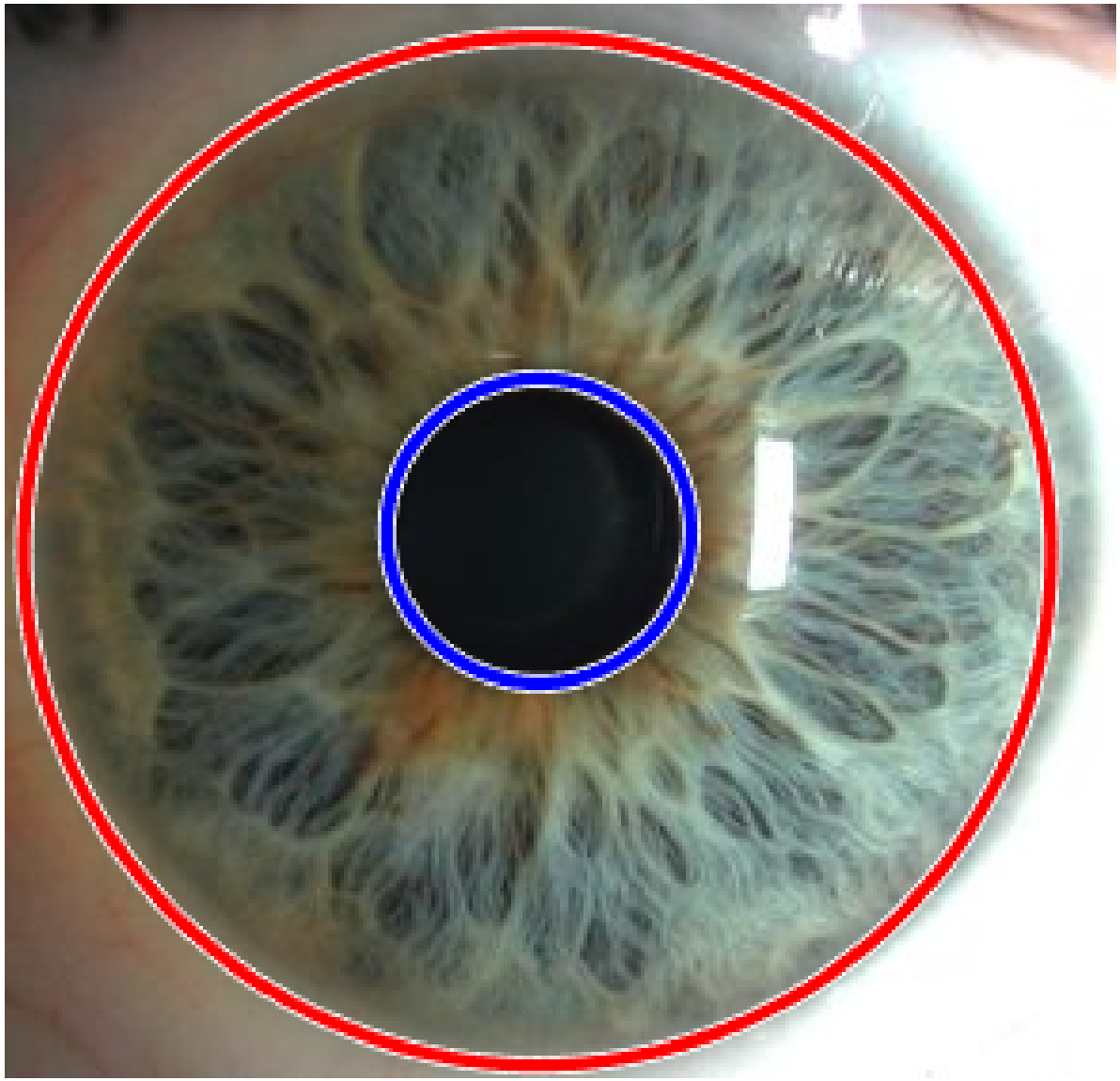


# Retina – Retinex

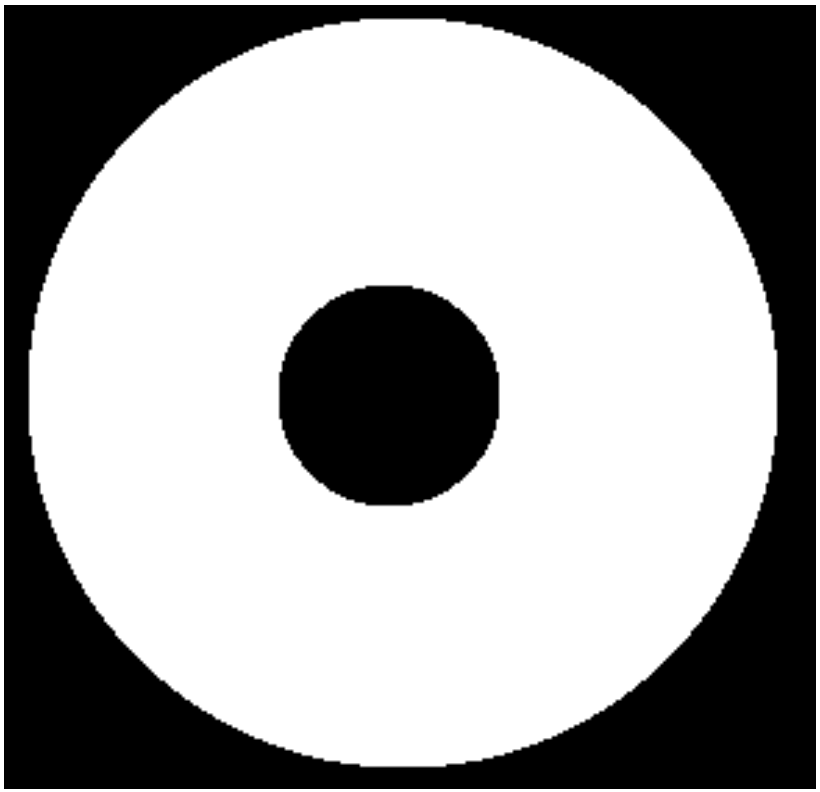


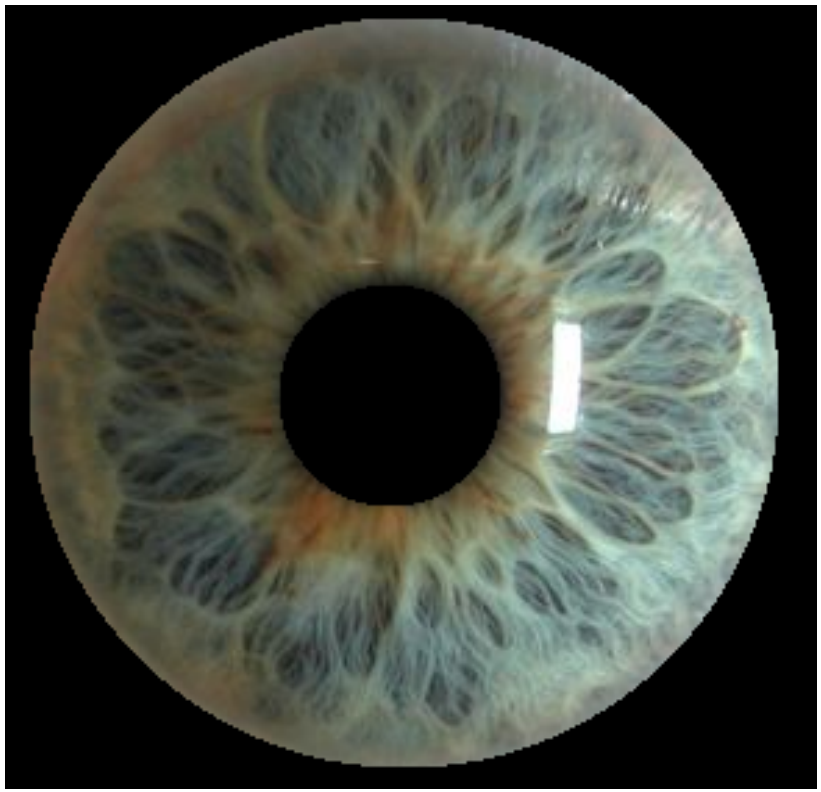


# Retina – Retinex



# Retina – Retinex





# Klastryzacja na podstawie intensywności

- k-means
- Mean-shift
- adaptatywny multiOtsu
- watershed
- segmentacja grafowa
- region growing

# Region growing

- Idea: zaczynamy w ustalonym regionie, badamy podobieństwo sąsiednich pikseli, w przypadku wystarczającego podobieństwa dodajemy je do regionu
- Przykład dla intensywności:
  $S, \Delta, \delta$ – parametry wejściowe
  Do{
  $S' \leftarrow S$
  $\forall_{x \in N(S')} \text{if } \frac{1}{\delta}\text{mean}(S') < x < \delta\text{mean}(S') : \ S' \leftarrow S' \cup \{x\};$
  }While$(\|S - S'\| > \Delta)$,

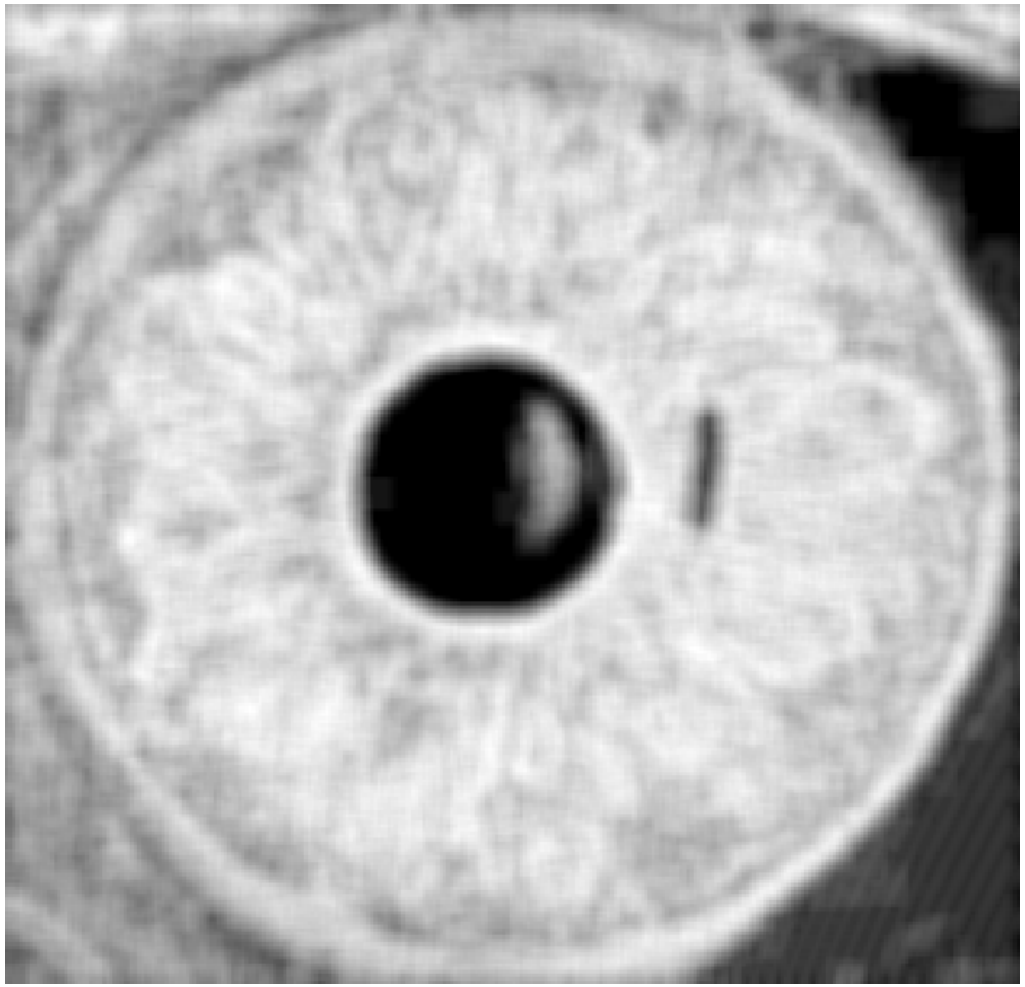
# Tekstura, zmienność

- filtr entropijny $H = -\sum p \log p$
- GLCM

$$M_{\Delta x,\Delta y}(k,o) = \sum_{x,y} \begin{cases} 1 & I(x,y) = k \wedge I(x+\Delta x, y+\Delta y) = o \\ 0 & \text{w p.p.} \end{cases}$$

  - kontrast $\sum_{k,o} |k-o|^2 p(k,o)$
  - korelacja $\sum_{k,o} \frac{(k-\mu k)(o-\mu o)p(k,o)}{\sigma_k \sigma_o}$
  - energia $\sum_{k,o} p(k,o)^2$
  - homogeniczność $\sum_{k,o} \frac{p(k,o)}{1+|k-o|}$
- filtr Gabora

# Tekstura, zmienność

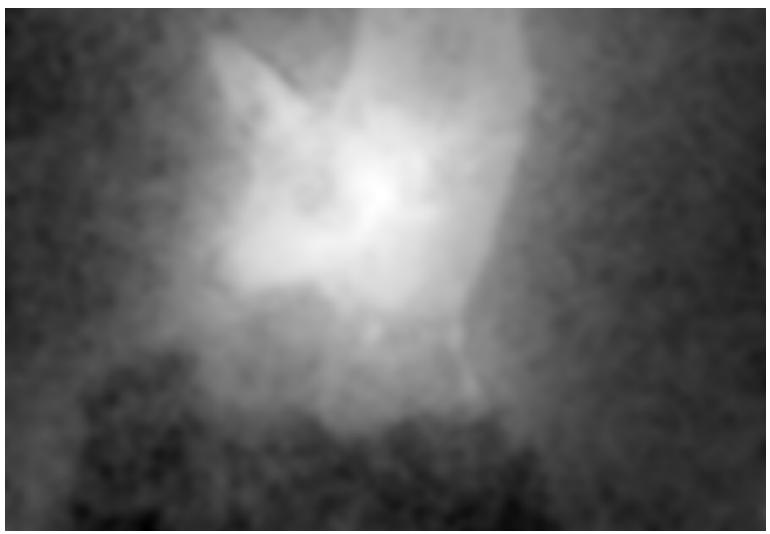


# Falki Gabora

- $f(x) = e^{-\frac{(x-x_0)^2}{a^2}} e^{-iA(x-x_0)}$
- minimalizują wariancję w dziedzinie przestrzennej i częstotliwościowej
- nie są ortogonalne
- FT falki Gabora, jest falką Gabora

# Falki Gabora

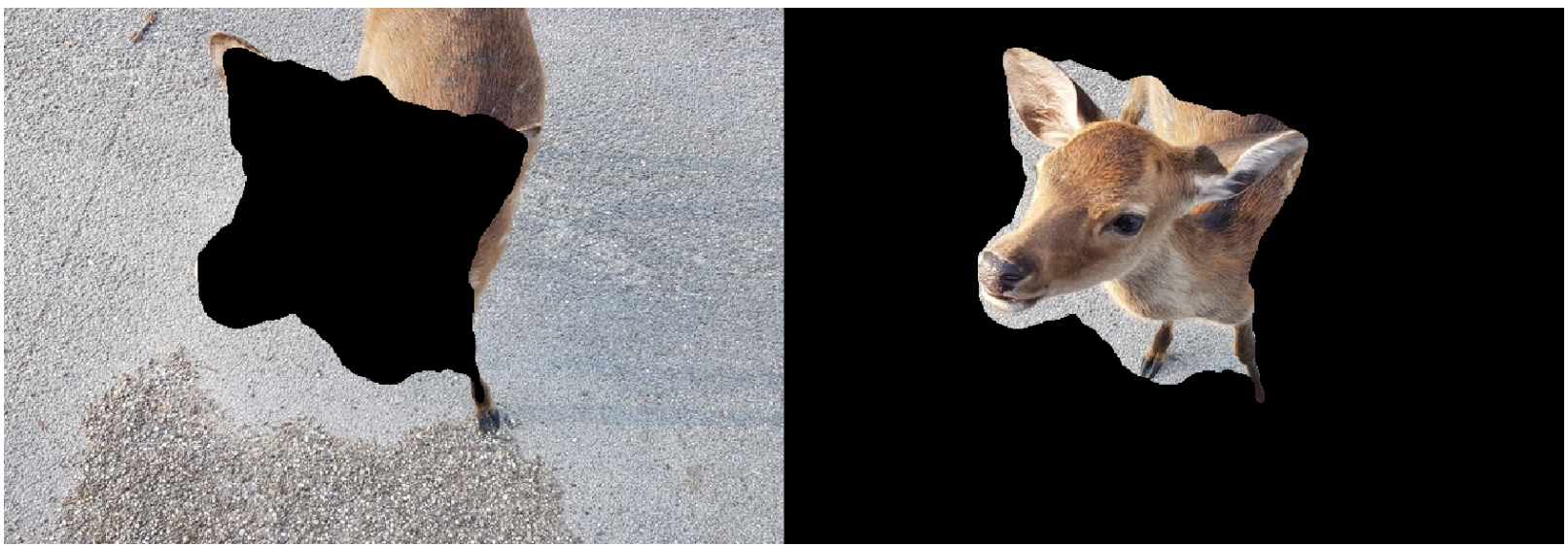




# Falki Gabora



# Falki Gabora



# Falki Gabora





# Snakes – idea

- Adaptacja praktyki z sygnałów 1D – dla grupy punktów (próbek), chcemy znaleźć krzywą 'dostatecznie bliską' próbkom o ustalonych własnościach (np. gładkości).
- Standardową miarą odległości jest: $\mathrm{dist}(M, \hat{M}) = \sum_i (M(x_i) - \hat{M}(x_i))^2$
- Gładkość definiujemy przez równowagę energii $E = aE_1 + bE_2$, dla $a + b = 1$ oraz $E_1 = \int \left( \frac{dM(x)}{dx} \right)^2 dx$, $E_2 = \int \left( \frac{d^2M(x)}{d^2x} \right)^2 dx$
- Minimalizujemy globalną energię: $\mathrm{dist} + \lambda E$
- $\mathrm{dist}$ możemy interpretować jako sumę energii zewnętrznych $E_{\mathrm{dist}} = \lambda_t e_t + \lambda_k e_k + \lambda_d e_d$

# Snakes – gradient

- otrzymaliśmy dość złożony model do minimalizacji
- w celu efektywnych obliczeń często jest upraszczany – energia wewnętrzna redukowana jest tylko do pierwszego rzędu
- energię zewnętrzną upraszczamy do czynnika krawędziowego – $E_{\text{dist}} = \lambda_k e_k = -\lambda_k \int \left\| (G_\sigma * \nabla^2 I) M(s) \right\|^2 ds$
- problem upraszcza się do $\min_{f;\alpha,\lambda} \left( \alpha \int \left\| \dot{f}(s) \right\|^2 ds - \lambda \int \left\| \nabla I(f(s)) \right\|^2 ds \right)$ ... który wciąż jest skomplikowany – rozwiązanie numeryczne na zasadzie programowania dynamicznego

# Snakes – Active Contours Without Edges

- inne popularne uproszczenia idei snake'ów również korzystają z krawędzi – gradient maksymalizuje się na krawędziach, dzięki czemu ewolucja zatrzymuje się
- model bezkrawędziowy – T.F. Chan, L.A. Vese: *Active Contours Without Edges*
  dwie intensywności: $F(C) = F_1(C) + F_2(C) =$
  $\int$ wewnątrz $C |I(x,y) - c_1|^2 \, dxdy +$
  $\int$ na zewnątrz $C |I(x,y) - c_2|^2 \, dxdy$, dla stałych $c_1, c_2$ będących średnimi intensywności po obu stronach krzywej $C$. Zadanie minimalizacyjne $\inf_C m\mathrm{len}(C) + F(C)$

# Snakes – Active Contours Without Edges

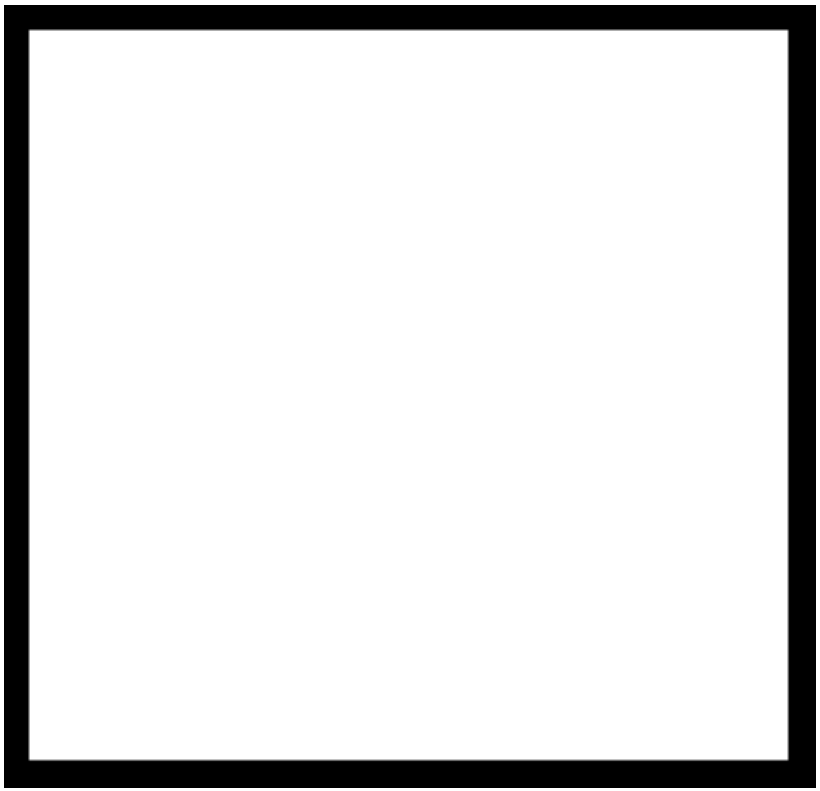
Estymacja $C$ w rejonie $\Omega$:

$$\begin{cases} \text{wnętrze}(C) = w = \{(x, y) \in \Omega; \phi(x, y) > 0\} \\ \text{zewnątrz}(C) = \omega \setminus \overline{w} = \{(x, y) \in \Omega; \phi(x, y) < 0\} \\ C = \partial w = \{(x, y) \in \Omega; \phi(x, y) = 0\} \end{cases}$$

$$\begin{aligned} F(c_1, c_2, \phi) = & \quad m \int_\Omega \delta(\phi(x, y)) \left|\nabla \phi(x, y)\right| + \\ & \lambda_1 \int_\Omega \left|I(x, y) - c_1\right|^2 H(\phi(x, y)) dxdy + \\ & \lambda_2 \int_\Omega \left|I(x, y) - c_2\right|^2 (1 - H(\phi(x, y))) dxdy \end{aligned}$$

Wyznaczamy $c_1(\phi), c_2(\phi)$, dodajemy iteracje, rozwiązujemy równanie różniczkowe $\frac{\partial \phi}{\partial t}$ znajdując minimum.

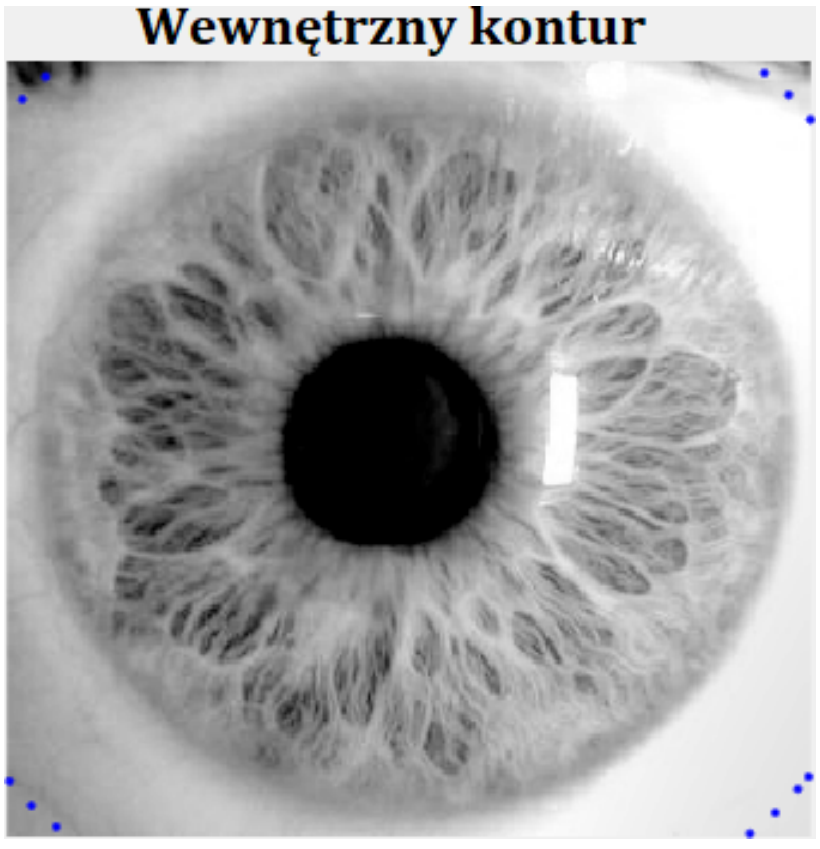
# AC – przykłady





Przebieg działania

# AC – przykłady






# AC – przykłady

