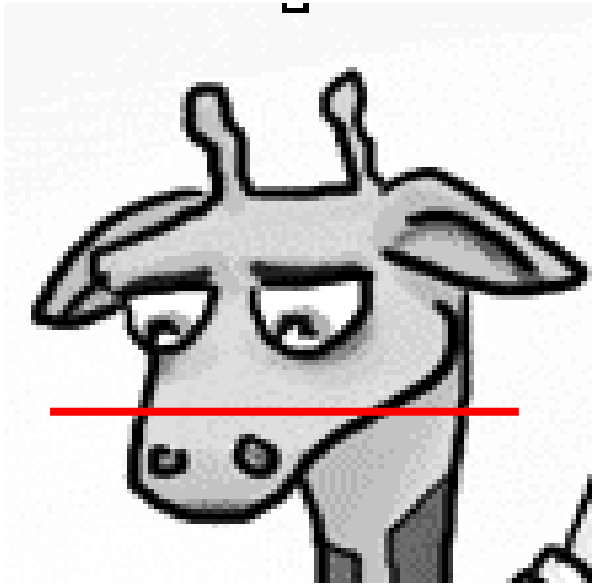
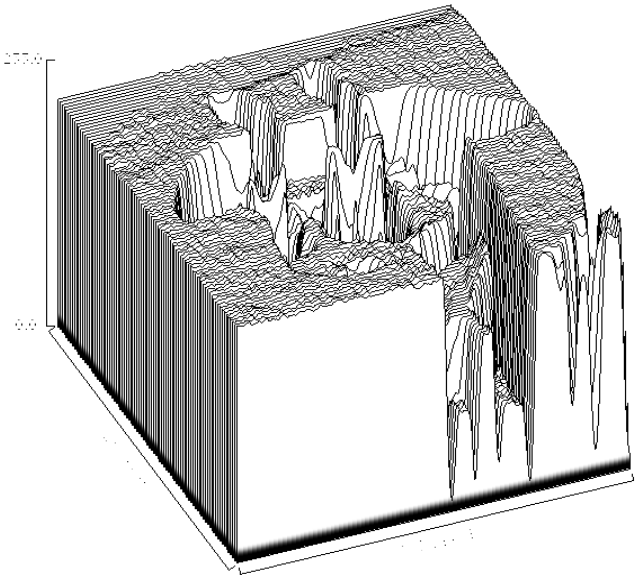
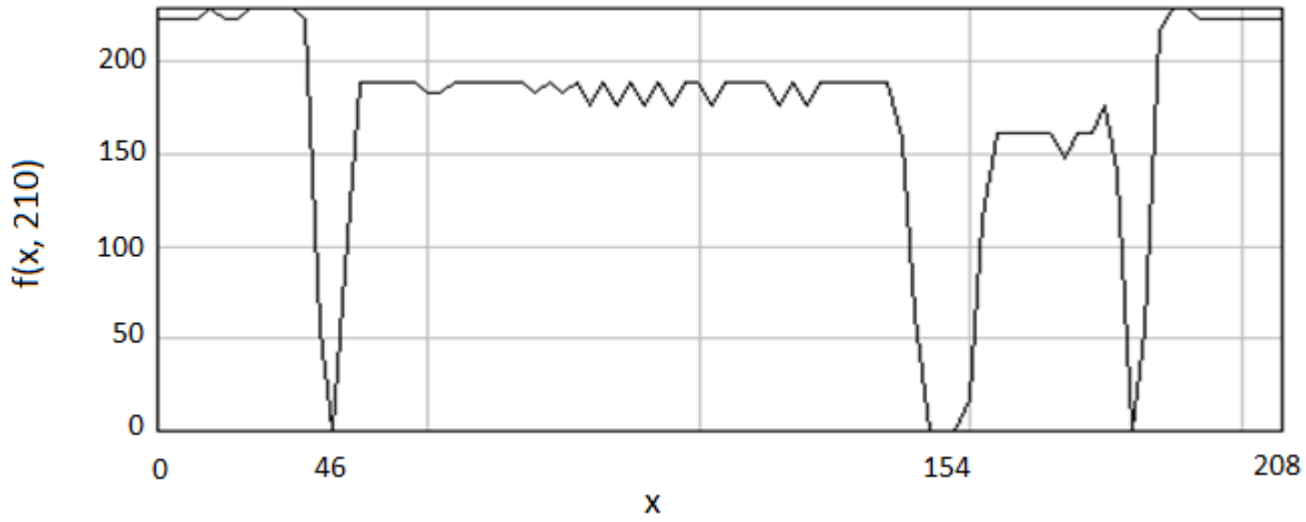
# OpenCV
# Élkeresés

# Élek



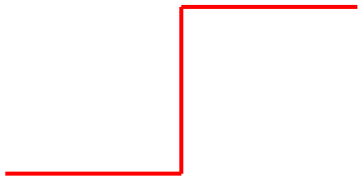




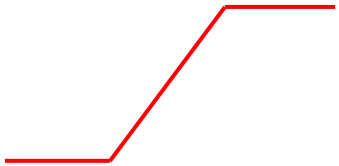
# Élek

- Él: a **képfüggvény** hirtelen változása
- Típusai

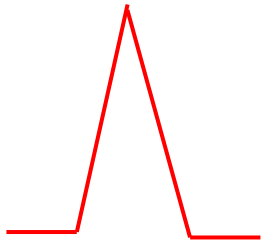
Lépcsős



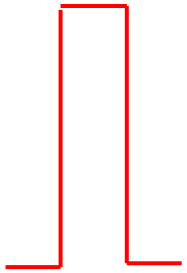
Rámpaszerű



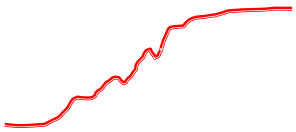
Tetőszerű



Vonalszerű él



- Felvételeken zajos formában jelennek meg



# Gradiens módszerek

- Adott pontbeli *x* ill. *y* irányú változás jellemezhető egy
  - *f* folytonos képfüggvény esetén *x* és *y* szerinti parciális deriváltakkal

$$\frac{\partial f(x,y)}{\partial x} \qquad \frac{\partial f(x,y)}{\partial y}$$

  - *I* diszkrét képfüggvény esetén közelíthető

$$\frac{I(x,y)-I(x_0,y)}{x-x_0} \qquad \frac{I(x,y)-I(x,y_0)}{y-y_0}$$

  - Gradiens erőssége:

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

  - Gradiens iránya:

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

# Feladat: Roberts gradiens

- Töltsön be egy képet (szürkeárnyalatosként)
- Hozzon létre két azonos méretű fekete-fehér képet (CV_16S ->short)
- Határozza meg az alábbi képet: (*I* az eredeti kép)

$$G_1(x,y) = I(x,y) - I(x+1,y+1)$$
$$G_2(x,y) = I(x+1,y) - I(x,y+1)$$

- A gradiens erősség közelítése:

$$G = |G_1| + |G_2|$$

- Konvertálja bájtra:

```
convertScaleAbs(G, dest);
```

- Jelenítse meg a gradiens erősség képet (G)
- Küszböljön a zajok (hamis élek) elnyomása érdekében
- Jelenítse meg az élképet

*Megj.: 2-2 pontra épít => a kiugró értékek erősen hatnak*

# Gradiens módszerek: Sobel

- Illesszük a kernelt egy *(x,y)* képpontra
- Számoljuk ki a kernel pontjai és a kép kernel alá eső pontjainak szorzatösszegét.
- A szorzatösszeget mentsük az eredménykép *(x,y)* koordinátájára.
- Ismételjük, ameddig van még feldolgozatlan képpont.

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −2 | 0 | 2 |
| −1 | 0 | 1 |

$$(-1)\cdot 10 + (-2)\cdot 10 + (-1)\cdot 13 + 1\cdot 10 + 2\cdot 14 + 1\cdot 10$$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 10 | 11 | 14 | 10 | 11 | 14 | 10 | 11 | 14 |
| 13 | 10 | 10 | 13 | 10 | 10 | 13 | 10 | 10 |
| 10 | 10 | 11 | 50 | 10 | 11 | 10 | 10 | 11 |
| 10 | 10 | 50 | 50 | 50 | 10 | 10 | 10 | 10 |
| 10 | 10 | 50 | 50 | 50 | 11 | 10 | 10 | 11 |
| 10 | 10 | 10 | 50 | 10 | 10 | 10 | 10 | 10 |
| 10 | 10 | 11 | 10 | 10 | 11 | 10 | 10 | 11 |
| 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 8 | -2 | -6 | 8 | -2 | -6 | 8 | 0 |
| 0 | 5 | 1 | -6 | 5 | 1 | -6 | 5 | 0 |
| 0 | -1 | 45 | -4 | -41 | 5 | -4 | -1 | 0 |
| 0 | 39 | 123 | -2 | -121 | -37 | -2 | -1 | 0 |
| 0 | 121 | 160 | -1 | -158 | -120 | -2 | 2 | 0 |
| 0 | 120 | 160 | 0 | -158 | -120 | -2 | 2 | 0 |
| 0 | 41 | 120 | -1 | -118 | -40 | -2 | 2 | 0 |
| 0 | 2 | 40 | -2 | -38 | 0 | -2 | 2 | 0 |
| 0 | 1 | 0 | -1 | 1 | 0 | -1 | 1 | 0 |

# Feladat: Gradiens nagyságának meghatározása

- Töltsön be egy képet (szürkeárnyalatosként)
- Sobel operátorral határozza meg az X és Y irányú gradienst:

```
Mat dx, dy;
Sobel(src, dx, CV_16S, 1, 0);
Sobel(src, dy, CV_16S, 0, 1);
```

- Közelítse a gradiens erősséget:

$$G = |dx| + |dy|$$

- Konvertálja bájtra:

```
convertScaleAbs(G, dest);
```

- Élek meghatározása: küszöböléssel

# Feladat: Gradiens nagyságának meghatározása

- Töltsön be egy képet (szürkeárnyalatosként)
- Sobel operátorral határozza meg az X és Y irányú gradienst:

```
Mat dx, dy;
Sobel(src, dx, CV_32F, 1, 0); //CV_32F – gyökvonás miatt
Sobel(src, dy, CV_32F, 0, 1);
```

- Számítsa ki a gradiens nagyságát

  pontonkénti négyzetre emelés: dx.mul(dx)

  pontonkénti gyökvonás: cv::sqrt( src, dest )

$$G(p) = \sqrt{dx(p)^2 + dy(p)^2}$$

- Konvertálja bájtra:

```
convertScaleAbs(G, dest);
```

- Élek meghatározása: küszöböléssel

# Éldetektálás: Laplace

- *f* folytonos képfüggvény esetén *x* és *y* szerinti másodrendű parciális deriváltakkal

$$\frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

- *I* diszkrét képfüggvény esetén a közelítés:

$$\frac{I(x+h,y)+I(x-h,y)-2f(x,y)}{h^2} + \frac{I(x,y+h)+I(x,y-h)-2f(x,y)}{h^2}$$

- Maszkkal (erősséged ad, irányt nem)

Négyszomszédság szerint

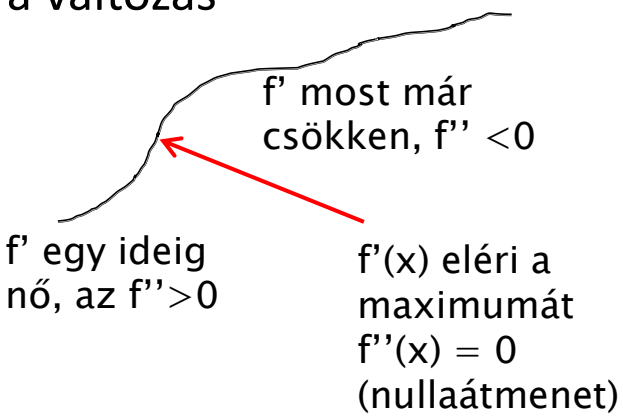
| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | −4 | 1 |
| 0 | 1 | 0 |

Nyolc szomszédság szerint

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | −8 | 1 |
| 1 | 1 | 1 |

# Éldetektálás: Laplace

- 1. derivált (f') ~ milyen gyorsan változik a függvényérték egy pont körül
  érintő meredeksége
- 2. derivált(f'') ~ gyorsul vagy lassul-e a változás
  érintő meredekségek eltérése




- Probléma:
  rámpaszerű élnél az f'' kétszer reagál
- Megoldás: nulla átmenetek figyelése
  elméletileg jó, de sok hamis élt ad → zajcsökkentés fontos!

# Opcionális feladat: Laplace operátor kipróbálása

- Töltse be egy képet (szürkeárnyalatosként)
- Alkalmazzon rá Gauss szűrést

```
void GaussianBlur(src, dest, kernelSize, sigma, sigma);
```

- Jelenítse meg a Laplacian függvények által adott képet:

```
void Laplacian(InputArray src, OutputArray dest,
                               int ddepth, int kernelSize=1);
```

`ddepth: CV_16S`

- Valósítson meg egy nullaátmenet figyelő függvényt: (short)
  - A kép értéke legyen 255, ha a Laplace eredménye 0, vagy két szomszédos pont előjele ellentétes.
  - Minden más esetben a pont értéke legyen nulla.

- Megjelenítés: konvertálja bájtra:

```
void convertScaleAbs(InputArray dest, OutputArray dest2);
```

# Éldetektálás: Canny éldetektáló

- Simítás (Gauss)
- Gradiens számítás
  - OpenCV alatt Sobel gradiens
- Nem maximum értékek elnyomása
  - lokálisan nézve, él irányra kb. merőlegesen
- Hiszterézis küszöbölés
  - th1 < th2 két küszöbérték
  - Egy pontot biztos élpont, ha a gradiens erősség a th2 felett van
  - Egy pontot élpont, ha a th2>=gradiens erősség > th1. és van csatlakozó biztos élpont
  - Más esetben a pontot nem tekintjük élpontnak.

```
cv::Canny(InputArray src, OutputArray dest, int th1, int th2
                          [,int sobelKernelMeret] [,bool L2grad = false]);
```

  - input: egy csatornás, 8 bites kép
  - output: egy csatornás, 8 bites kép (fekete-fehér)
  - L2grad: pontosabb (, és lassabb) gradiens erősség számítás

# Hough Circle

- Canny éldetektálóra épül (Sobel gradiens)
- Azokat a köröket adja vissza, amelyek elegendő számú előtérponton mennek át.

```
void cv::HoughCircles(InputArray image,  //szürkeskálás kép
    OutputArray circles,  //vector<Vec3f>
    int method,  //HOUGH_GRADIENT
    double dp,   //felbontás inverze (1-eredeti, 2-felezett kép,)
    double minDist,  //körök közötti távolság
    double param1 = 100,  //Canny Th (felső)
    double param2 = 100,  //Hány ponton menjen át minimum
    int minRadius = 0,
    int maxRadius = 0
);
```

# Feladat: Korongdetektálás

https://arato.inf.unideb.hu/szeghalmy.szilvia/kepfeld/img/go2.png

- Szürkeskálában olvassa fel a képet
- Végezzen kördetektálást a HoughCircles segítségével
  - felezze a felbontást (d = 2)
  - a minR és maxR legyen a korong mérete alapján beállítva
  - várja el, hogy legalább 50 képponton átmenjen a kör a detektáláshoz
  - küszöbnek állítson be kb. 80-as értéket.
- Járja be a kapott kör vektort:

```
for(auto c : circles){
    //középpont (x,y) koordinátája: c[0], c[1],  sugár: c[2]
```

- A középpont világosságkódja alapján határozza meg mely korongok mely játékoshoz tartoznak.
- Különböző színnel rajzolja körbe a két játékos korongjait:

```
circle(InputOutputArray img, Point cp, double radius,
                        Scalar color, int thickness=1);
```

# Hough Circle: HOUGH_GRADIENT_ALT

- Canny éldetektálóra épül (Scharr gradiens)
- Azokat a köröket adja vissza, amelyek elegendő számú előtérponton mennek át.

```
void cv::HoughCircles(InputArray image,  //szürkeskálás kép
    OutputArray circles,  //vector<Vec3f>
    int method,  //HOUGH_GRADIENT_ALT
    double dp,   //felbontás inverze (1-eredeti, 2-felezett kép,)
    double minDist,  //körök közötti távolság
    double param1 = 100,  //Canny Th (felső)
    double param2 = 0.9,  //kör tökéletessége
    int minRadius = 0,
    int maxRadius = 0
);
```


99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.
127 little bugs in the code...