ROZ1 – CVIČENÍ VI.

Geometrická registrace (matching) obrazů

Registraci obrazu (*Image Registration*)

- Více snímků téže scény
- Odpovídající pixely v těchto snímcích musí mít stejné souřadnice
- Pokud je nemají >> chybná detekce např. změn v obraze

- **Kategorie registrace obrazu:**
  - Different viewpoints – multiview (více pohledový)
  - Different times – multitemporal (více časový)
  - Different modalities - multimodal (multimodální)
  - Scene to model registration (Scéna k modelu registrace)

- Model geometrického zkreslení:

$$g = T_G(f)$$

Registraci obrazu - **postup**

- **Control point selection (kandidáti na řídících body)**
- **Control point matching (párování řídících bodů):**
- **Transform model estimation**
- **Image resampling and transformation**

Registraci obrazu - **postup**

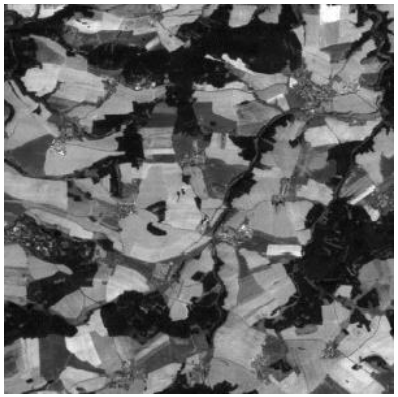
- **Control point selection (kandidáti na řídících body)**
  - zvlášť na referenčním obrázku a zvlášť na registrovaném obrázku
  - musí jít dobře automaticky detekovat
  - musí být stabilní
  - musí jich být dostatečný počet
  - měli by být rozmístěny pokud možno po celém snímku
  - musí být invariantní k transformaci – z tohoto hlediska nejvíce vyhovují právě ty těžiště uzavřených oblastí
  - Jsou to většinou rohy, těžiště uzavřených oblastí nebo extrémní křivosti křivek.
- **Control point matching (vybrání řídících bodů):**
  - mnoho technik, jak toto provádět
  - zde je hlavní teoretický problém

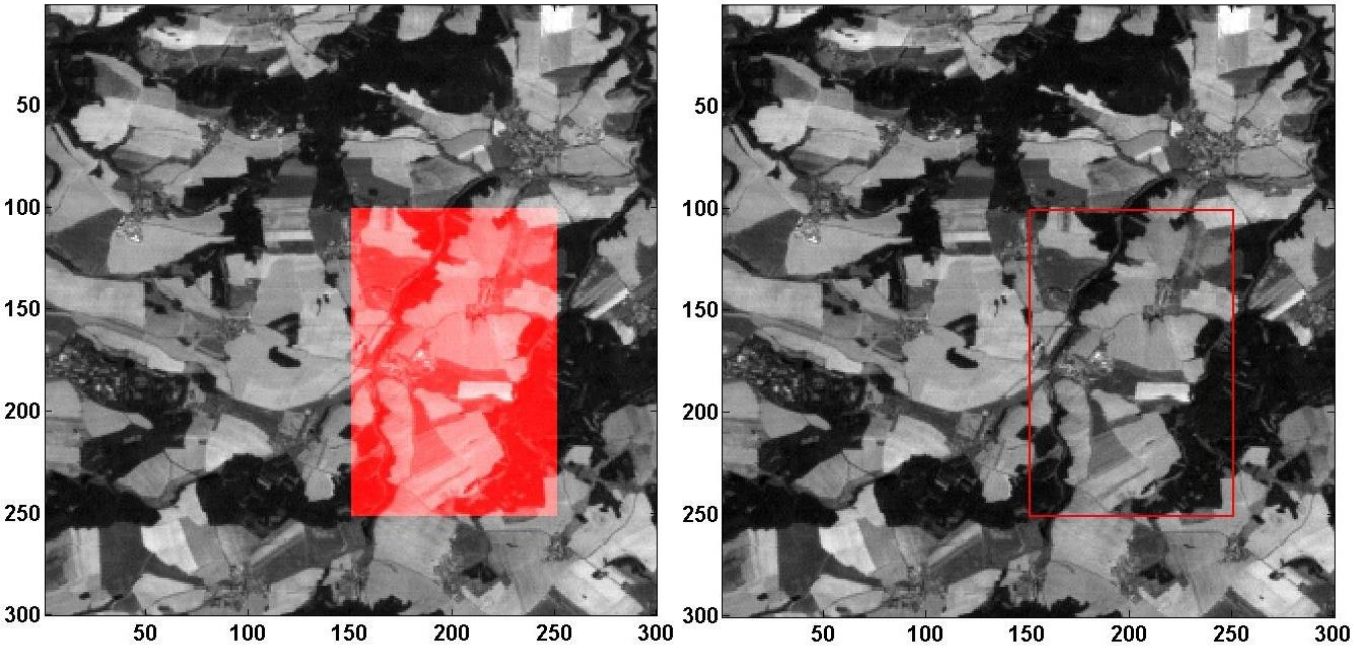
Control point matching - Metody

- **signálově závislé**
  - Obrazová korelace
  - Jiná míra podobnosti než korelace
  - Pyramidální reprezentace
  - Fázová korelace
- **signálově nezávislé (Příznakové metody)**
  - Kombinatorické (grafové)
  - TRS (translation, rotation, scaling)

Registrace pomocí korelace – úloha I.

- Napište funkci na registraci pomocí obyčejné korelace:
  - `function [] = regCorr(R, T)`
    - R … referenční obrázek (referencni.pgm)
    - T … template – výřez z obrázku R (templat.pgm)
    - Výstup bude:

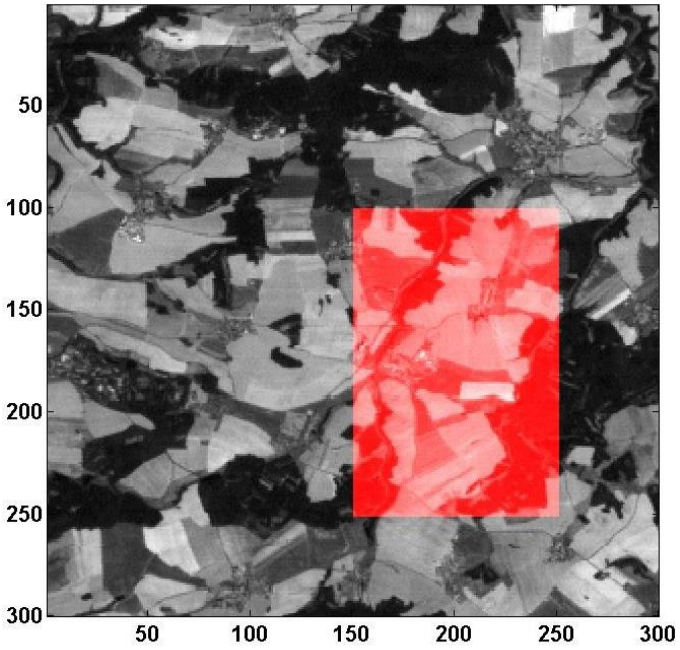




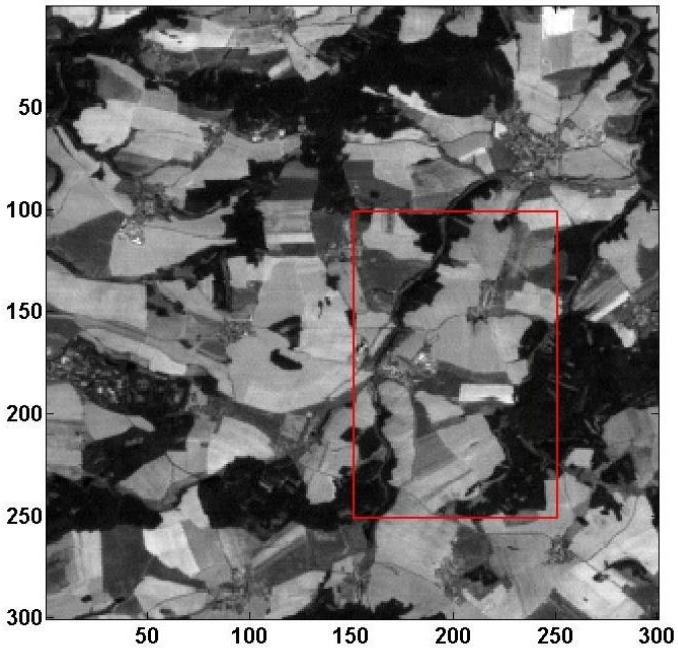
  - `filter2(), hold on, hold off, line(), uint8()`

ÚLOHA I. – ŘEŠENÍ A) VYBARVENÍ

```
function regCorr(R,T)
F = filter2(T,R,'valid');
[Y, X] = find(F==max(F(:)));
S=size(T);

H=zeros(size(R,1),size(R,2),3);
H(:,:,1)=R;
H(:,:,2)=R;
H(:,:,3)=R;

H(Y:Y+S(1),X:X+S(2),1)=255;
zobr(uint8(H));
end
```



ÚLOHA I. – ŘEŠENÍ B) RÁMEČEK




```
function regCorr(R,T)
F = filter2(T,R,'valid');
[Y, X] = find(F==max(F(:)));
S=size(T);

zobr(R);
hold on
line((X:X+S(2)),ones(S(2)+1,1)*Y,'Color','red','LineWidth',2);
line((X:X+S(2)),ones(S(2)+1,1)*Y+S(1),'Color','red','LineWidth',2);
line(ones(S(1)+1,1)*X,(Y:Y+S(1)),'Color','red','LineWidth',2);
line(ones(S(1)+1,1)*Y+S(1),(Y:Y+S(1)),'Color','red','LineWidth',2);
hold off
end
```

Fázová korelace

- Vysoká výpočetní rychlost
- Modifikací algoritmu lze dosáhnout dobré robustnosti vůči šumu
- Omezena na hledání geometrických transformací TRS
  - složitější transformace detekovat nelze
- Základem této metody je FST *(Fourier Shift Theorem)*:

$$\mathcal{F}_x[f(x - x_0)](k) = e^{-2\pi i k x_0} F(k)$$

Fázová korelace

- $\mathcal{F}[f(x,y)] = F(u,v)$ $\quad\quad$ $\mathcal{F}[w(x,y)] = W(u,v)$
- $f(x,y) = w(x-a, y-b)$
- *FST:* $F(u,v) = W(u,v) \cdot e^{-2\pi i k(ua+vb)}$
- **Cross–power spektrum**:

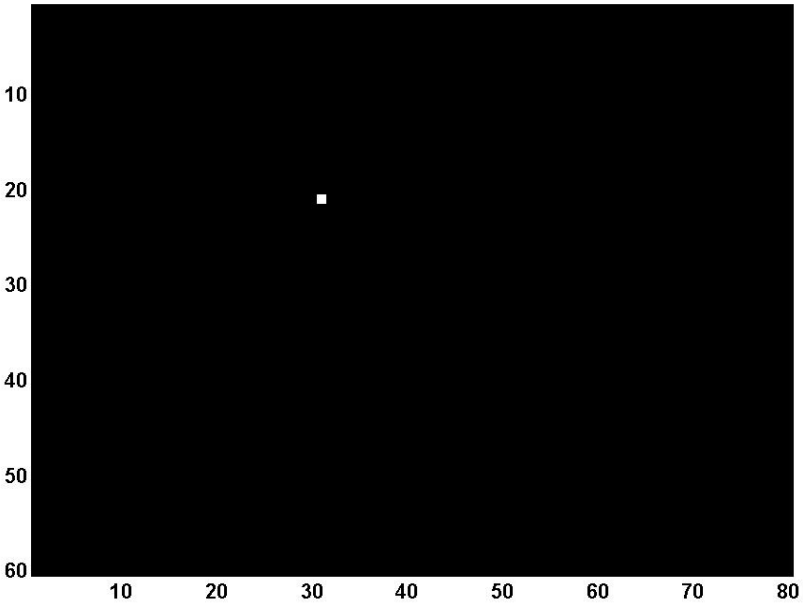
$$\frac{W \cdot F^*}{|W \cdot F|} = e^{-2\pi i(ua+vb)}$$

  - vyplývá z předpokladu, že obrázky jsou stejné jen posunuté (FST)
- *F… Fourier originálního obrázku*
- *F^*… komplexně sdružený*
- *W… Fourier okénka w (nebo posunutého obrázku)*
- *a,b… neznámé parametry posunu*
- Provede se IFT:

$$IFT\left(e^{-2\pi i(ua+vb)}\right) = \delta(x-a, y-b)$$

Registrace pomocí Fázové Kor. – úloha II.

- Napište funkci, na fázovou korelaci:
  - `function [X,Y] = phaCorr(R, T)`
    - R … referenční obrázek (rez.pgm)
    - T … template – posunutý R (rez_t.pgm)
    - Výstup bude: velikost posunu v ***x*** a **y**



Výsledný obrázek delta funkce je zvětšený…

ÚLOHA II. – ŘEŠENÍ

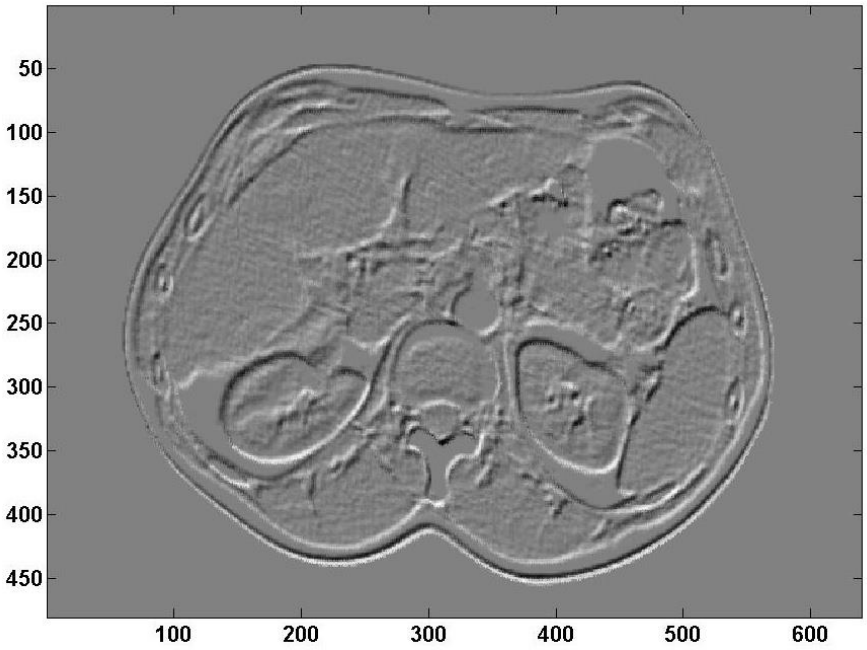
```
function [Y, X] = phaCorr(R,T)
FR=fft2(R);
FT=fft2(T,size(R,1),size(R,2));
Z=abs(ifft2((FR.*conj(FT)./(abs(FR).*abs(FT)))));

zobr(Z);

[Y, X] = find(Z==max(Z(:)));
X=X-1
Y=Y-1
End
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

>> phaCorr(RT,R)
X =

    30
Y =

    20
```

Příkazy:

- `[X,Y,B]=ginput(N)`
- `subplot(S1,S2,N)`
- `num2str(N)`
- `text(X,Y,'text','color','red');`
- `cart2pol(X,Y)`

Pomocí ručně vybraných bodů nalezněte úhel otočení. – úloha III.

- `function [X,Y] = rucniRot(R, T)`
  - R … referenční obrázek (rez.pgm)
  - T … template – středově rotovaný obrázek R (rez_r.pgm)
  - Výstup bude: úhel otočení
  - `cart2pol()`



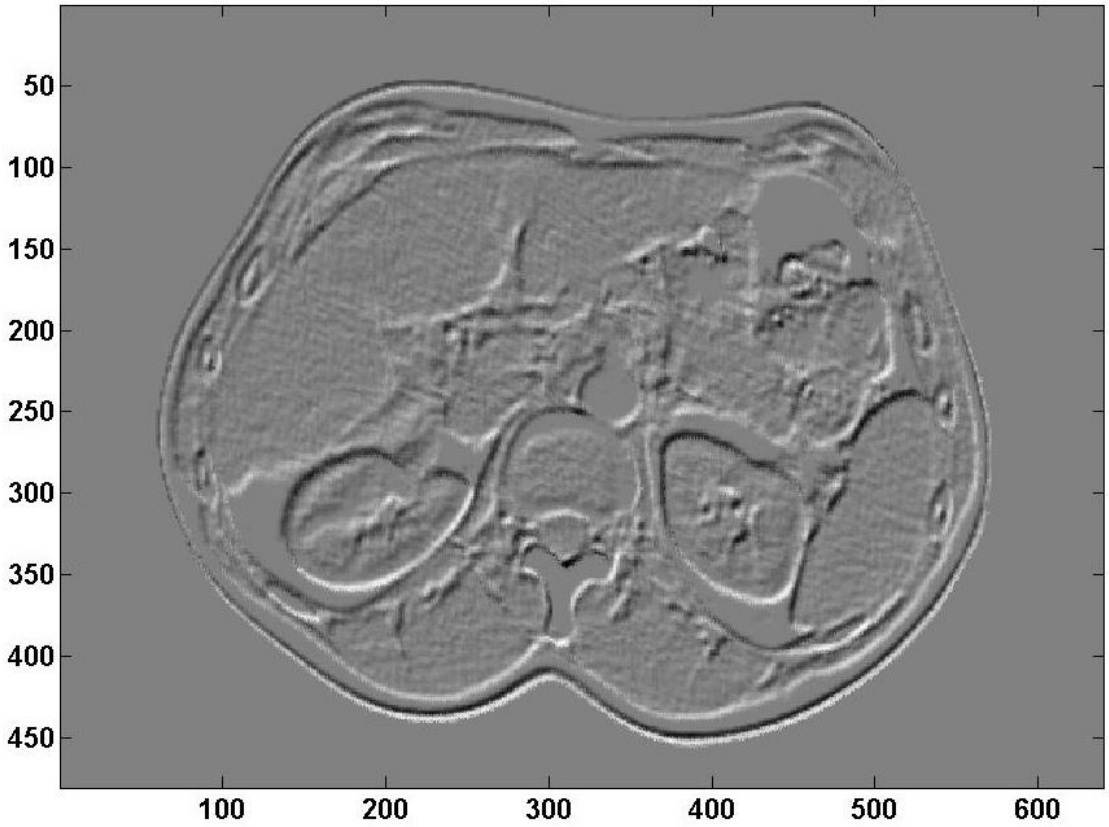
Zobrazení rozdílů po zpětném otočení podle nalezeného úhlu

ÚLOHA III. – ŘEŠENÍ

```
function R=rucniRot(I1,I2)
close all;
zobr(I1);
p1 = ginput(3);
for i=1:3
    text(p1(i,1),p1(i,2),['\times' num2str(i)],'Color','red');
end
zobr(I2);
p2 = ginput(3);
for i=1:3
    text(p2(i,1),p2(i,2),['\times' num2str(i)],'Color','red');
end
% prevedeni na souradnice
p1(:,1)=p1(:,1)-(size(I1,2)/2);
p1(:,2)=-(p1(:,2)-(size(I1,1)/2));
p2(:,1)=p2(:,1)-(size(I2,2)/2);
p2(:,2)=-(p2(:,2)-(size(I2,1)/2));
%transformace do polarnich
[t1,r1] = cart2pol(p1(:,1),p1(:,2));
[t2,r2] = cart2pol(p2(:,1),p2(:,2));
rotace = (t2-t1)*180/pi;
J=rotace<0;
rotace(J)=360+rotace(J);
R=sum(rotace)/3;
end
```

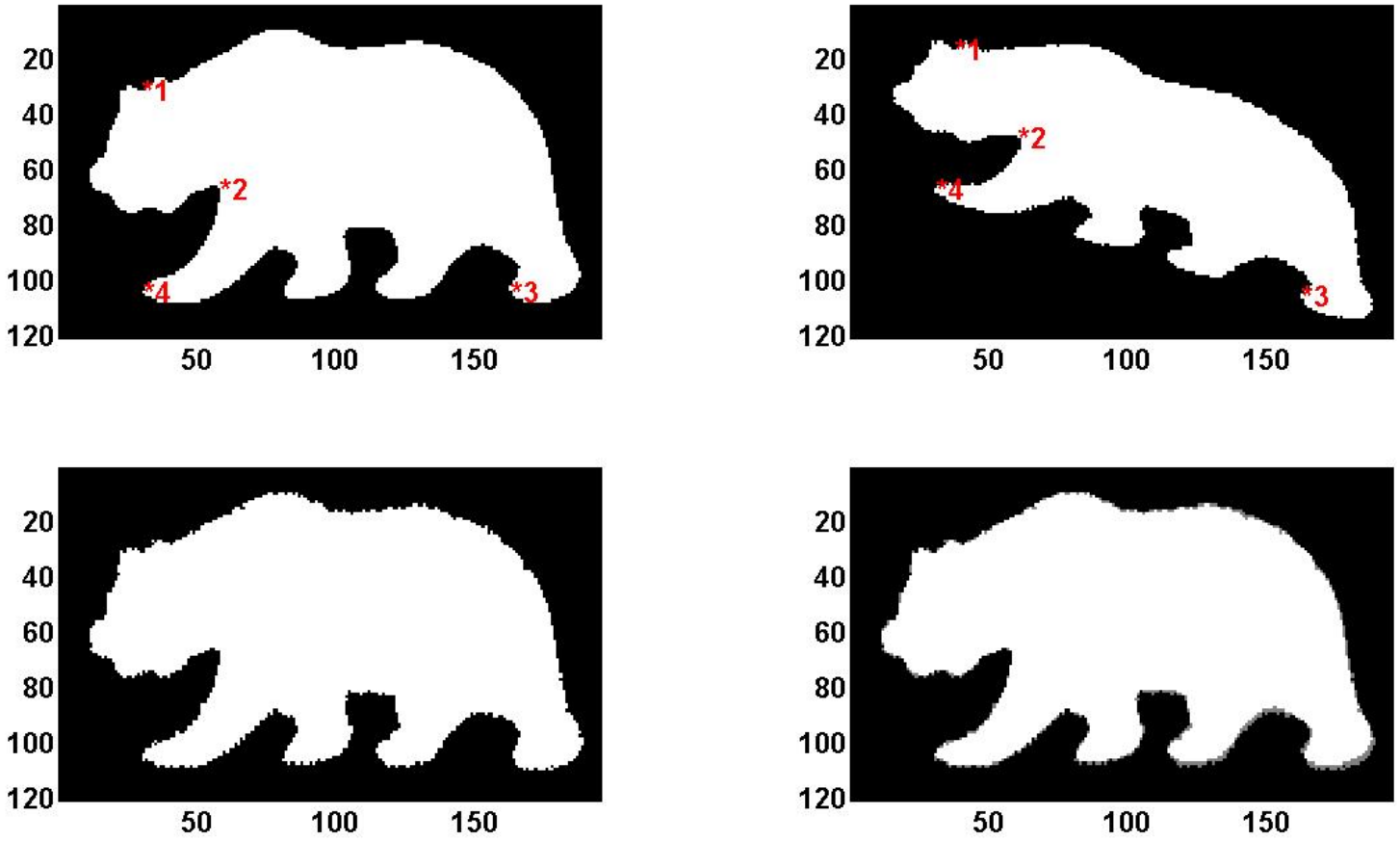
ÚLOHA III. – ŘEŠENÍ

```
rot=rucniRot(R,RR)

rot =

   29.9888
```



Afinní transformace – Úloha iv.

- Výsledný program pro registraci:



Metoda nejmenších čtverců (LMS)

- Minimalizujeme kriteriální funkci:

$$F(x) = \|\boldsymbol{Ax} - \boldsymbol{y}\|^2$$

- Matice $\boldsymbol{A^T A}$ je positivně definitní >> pro nalezení minima stačí funkci derivovat podle $\boldsymbol{x}$ a výsledek porovnat s nulou:

$$\frac{\boldsymbol{\partial F}}{\boldsymbol{\partial x}} = \boldsymbol{2A^T Ax - 2A^T y = 0}$$

$$\boldsymbol{x = (A^T A)^{-1} A^T y}$$

Afinní transformace

$$y_1 = a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}$$
$$y_2 = a_{2,1}x_1 + a_{2,2}x_2 + a_{2,3}$$

- Pro Euklidův prostor to zahrnuje:
  - posunutí, otáčení, změnu měřítka, zkosení, zrcadlení, a jejich skládání
- Důležitá vlastnost AT:
  - převádí přímky na přímky (nebo bod) a obecněji afinní podprostory na afinní podprostory

Afinní transformace řešená LMS

Pro přehlednost přeindexujeme:

$$v_1 = t_{1,1}u_1 + t_{1,2}u_2 + p_1$$
$$v_2 = t_{2,1}u_1 + t_{2,2}u_2 + p_2$$
$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$

$$\boldsymbol{y} = [v_1 \quad v_2], \boldsymbol{A} = [u_1 \quad u_2 \quad 1], \boldsymbol{x} = \begin{bmatrix} t_{1,1} & t_{2,1} \\ t_{1,2} & t_{2,2} \\ p_1 & p_2 \end{bmatrix}$$

kde:

- $\boldsymbol{T} = \begin{bmatrix} t_{1,1} & t_{2,1} \\ t_{1,2} & t_{2,2} \end{bmatrix}$…koeficienty transformace (bez posunu)
- $\boldsymbol{P} = [p_1 \quad p_2]$…koeficienty posunu

$$\boldsymbol{x} = (\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{y}$$

Určete Transformaci Méďů – úloha IV.

- Ze 3 bodů vzoru a odpovídajících 3 bodů obrazu spočitat transformaci
- Nejprve naprogramujte funkci pro určení koeficientů AT:

```
function [T, P] = urciTrans(U, V)
% - určí transformaci pro V = T*U + P
% pomoci metody nejmenších čtverců
```

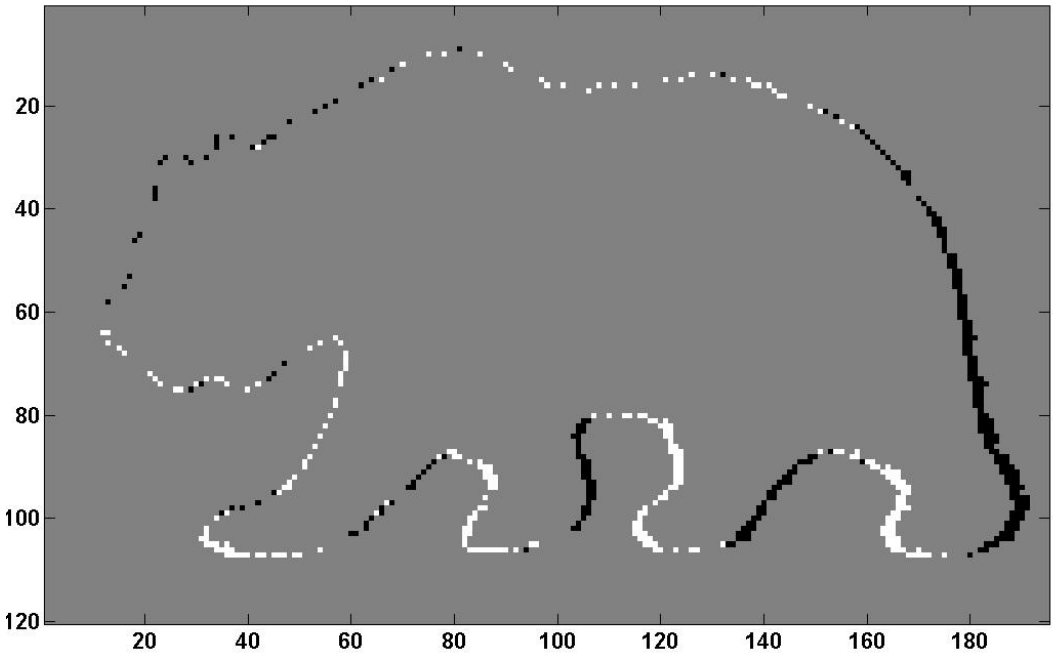
$$[x = (A^T A)^{-1} A^T y]$$

ÚLOHA IV. – ŘEŠENÍ

```
function [T, P] = urciTrans(U, V)
% V = T*U + P

A = [U, ones(size(U,1),1)];
Y = V;
X = ((inv(A'*A))*A')*Y;

T=X(1:2,1:2);
P=X(3,:);

end
```

Určete Transformaci Méďů – úloha IV.

- Spočítejte transformační koeficienty pro obrázky *MedaTran.pgm* a *Meda.pgm*
- Koeficienty dosaďte do `afinTran(Img, Tran, Pos)`
- Zobrazte rozdíly mezi *Meda.pgm* a zpětně transformovaným *MedaTran.pgm*



ÚLOHA IV. – ŘEŠENÍ

```
M=double(imread('Meda.pgm'));
MT=double(imread('MedaTran.pgm'));
zobr(MT);
[U1, U2]=ginput(3);
zobr(M);
[V1, V2]=ginput(3);
U = [U2, U1];
V = [V2, V1];
[T,P] = urciTrans(U,V);
zobr(M-afinTran(MT, T, P));
```

ROZ1 – CVIČENÍ

Konec cvičení pro tento rok.

Doporučujeme si zapsat –

- ROZ2 v ZS
- Speciální funkce a transformace v obraze (SFTO): momentové invarianty a wavelety