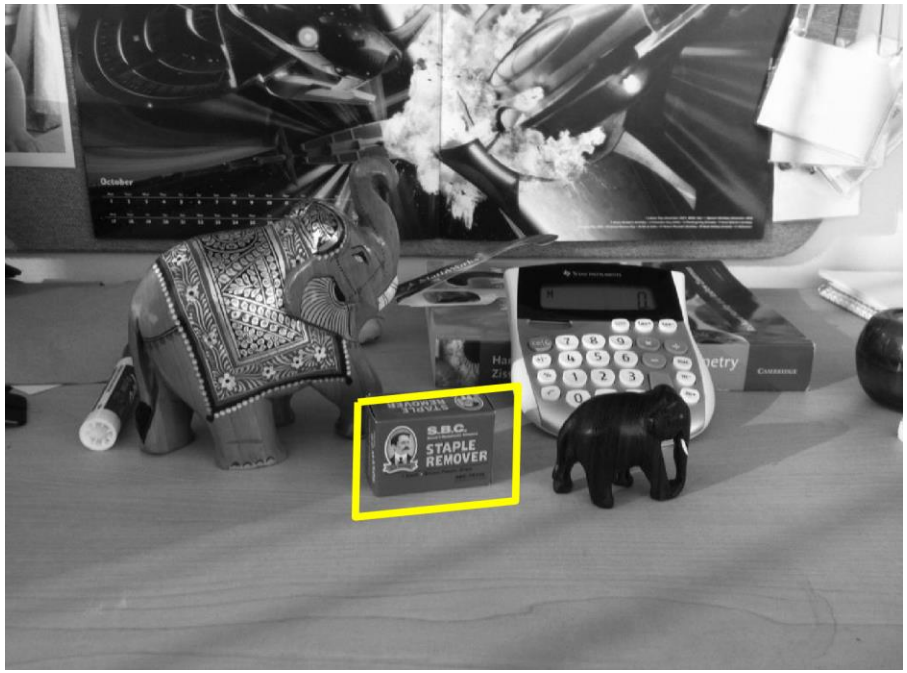
ROZ-Features

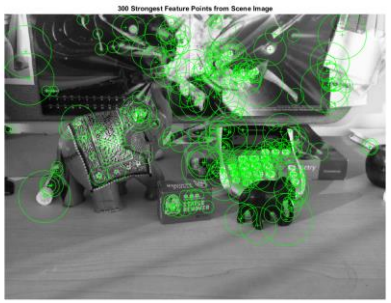
ÚTIA - ZOI
zoi.utia.cas.cz

Detekce objektů ve scéně



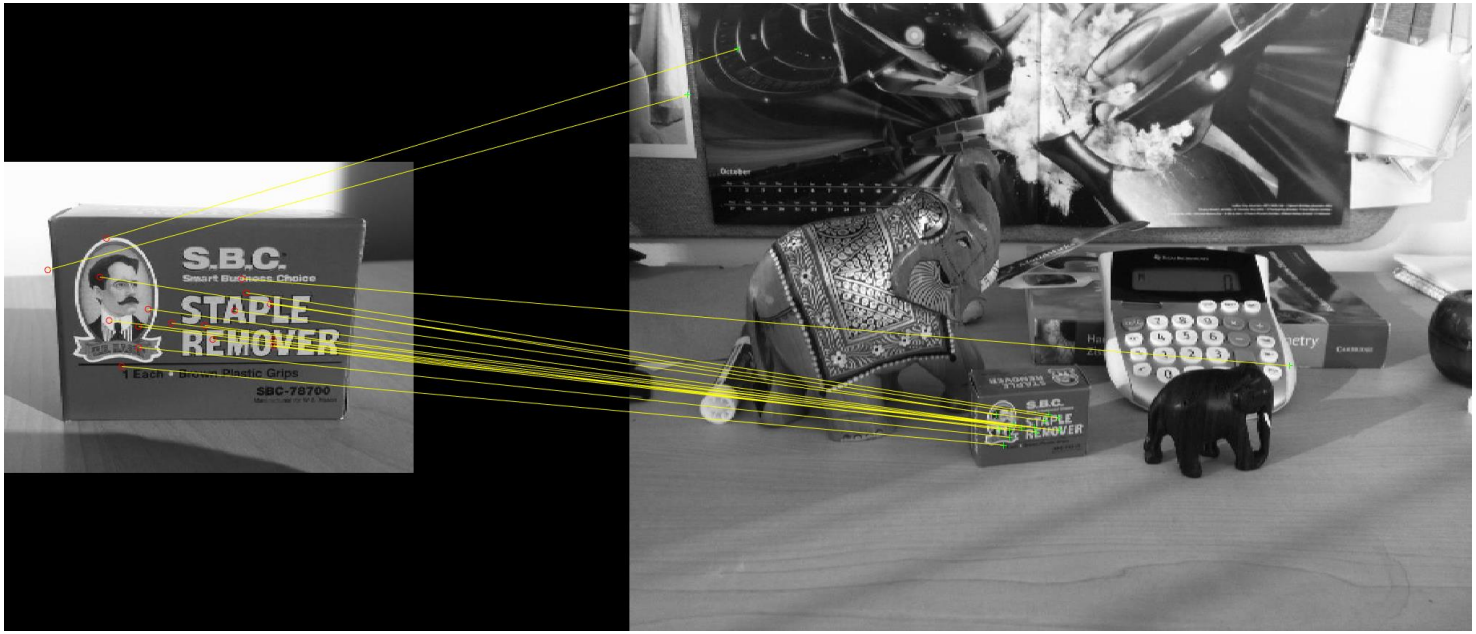
Detekce objektů ve scéně

```
% Načtení obrázků
boxImage = imread('boxImage.jpg');
sceneImage = imread('sceneImage.jpg');

% Detekce příznakú a zobrazení
boxPoints = detectSURFFeatures(boxImage);
scenePoints = detectSURFFeatures(sceneImage);

figure;
subplot(1,2,1)
imshow(boxImage);
title('100 Strongest Feature Points from Box Image');
hold on;
plot(selectStrongest(boxPoints, 100));

subplot(1,2,2)
imshow(sceneImage);
title('300 Strongest Feature Points from Scene Image');
hold on;
plot(selectStrongest(scenePoints, 300));
```



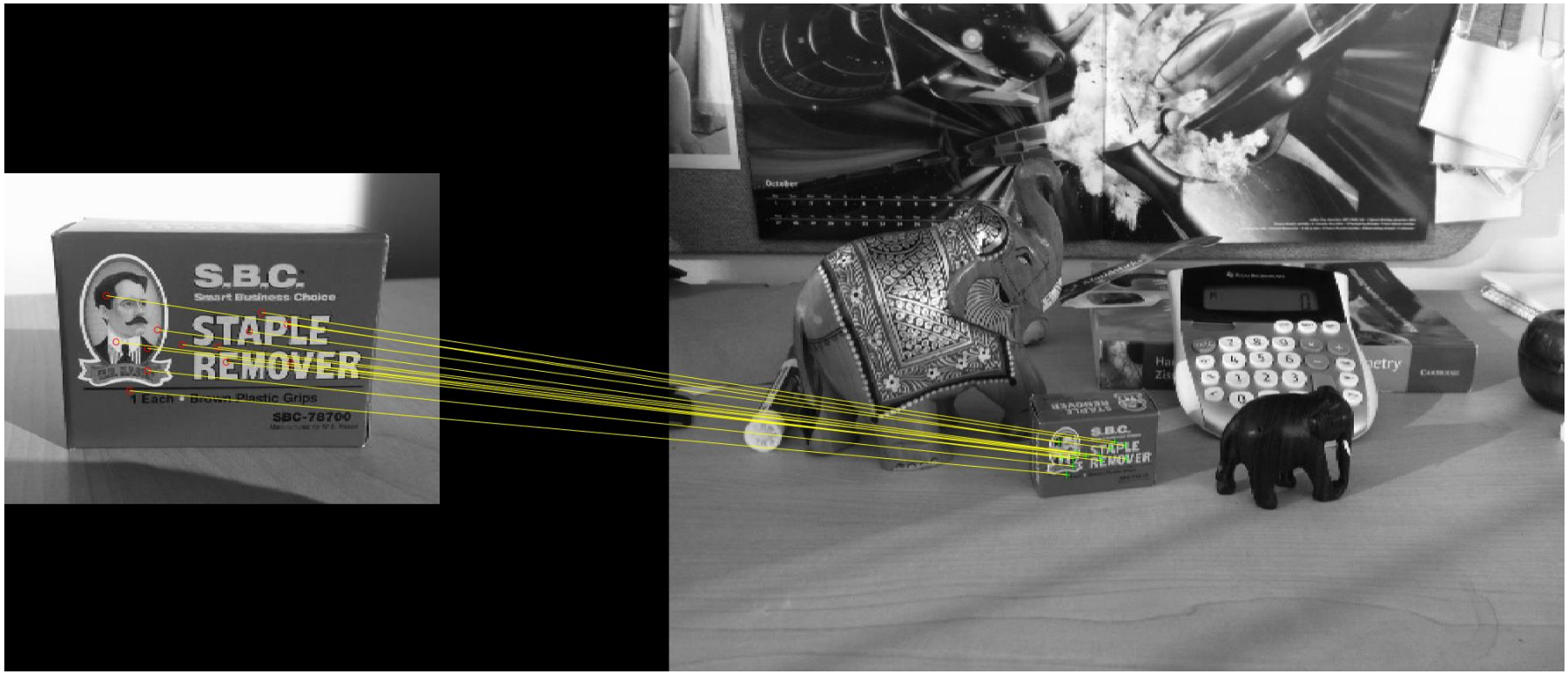
Detekce objektů ve scéně

```
% Extrakce deskriptorů
[boxFeatures, boxPoints] = extractFeatures(boxImage, boxPoints);
[sceneFeatures, scenePoints] = extractFeatures(sceneImage, scenePoints);

% Nalezení kandidátních bodů
boxPairs = matchFeatures(boxFeatures, sceneFeatures);
matchedBoxPoints = boxPoints(boxPairs(:, 1), :);
matchedScenePoints = scenePoints(boxPairs(:, 2), :);
figure;
showMatchedFeatures(boxImage, sceneImage, matchedBoxPoints, ...
    matchedScenePoints, 'montage');
```



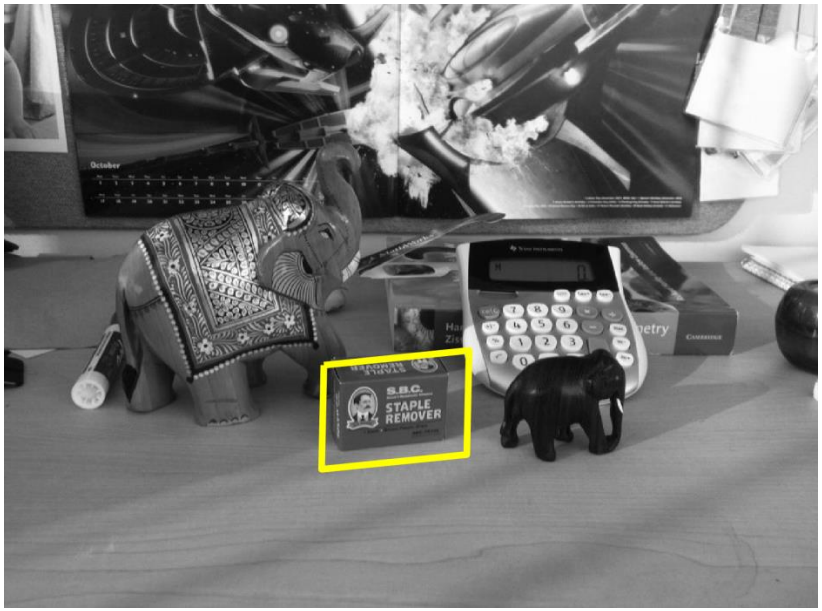
Detekce objektů ve scéně

```
% Vybrání bodů pomocí RANSAC + spočtení transformace
[tform,  inlierBoxPoints, inlierScenePoints] = ...
    estimateGeometricTransform(matchedBoxPoints, matchedScenePoints, 'affine');

figure;
showMatchedFeatures(boxImage, sceneImage, inlierBoxPoints, ...
    inlierScenePoints, 'montage');
```



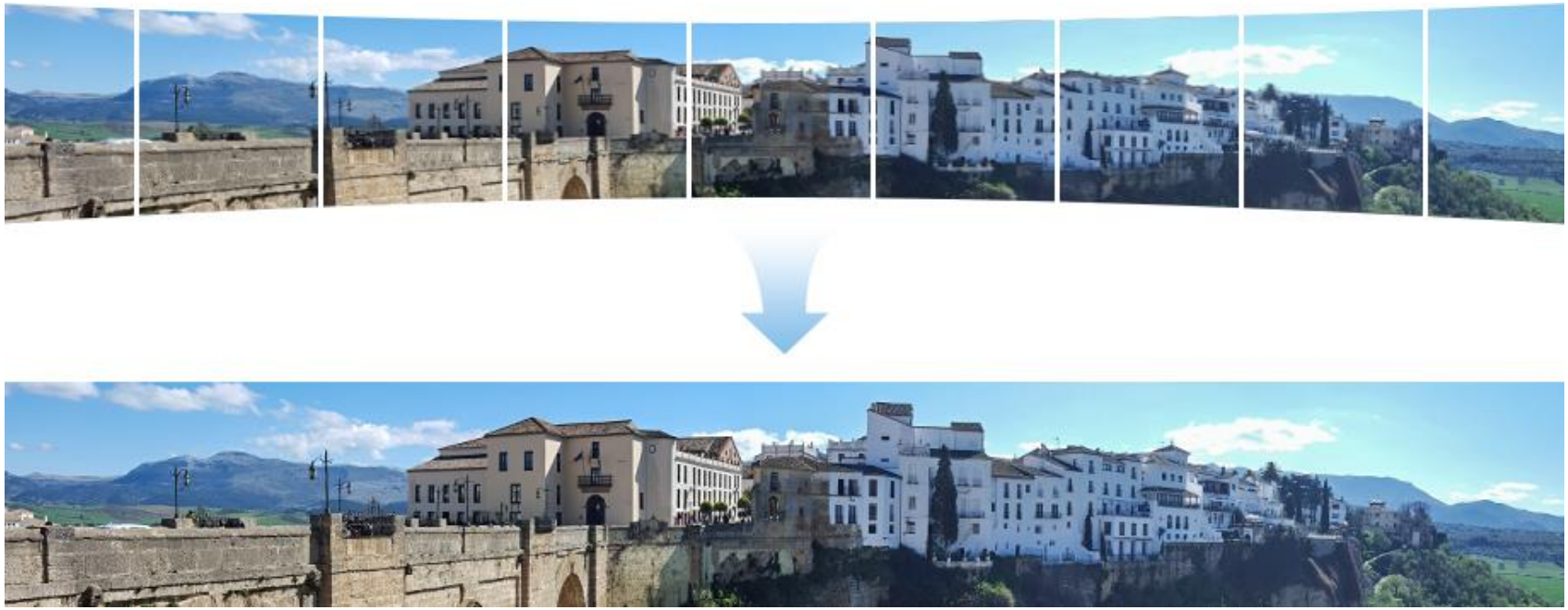
Detekce objektů ve scéně

```
% Transformace hranice objektu a překreslení do obrázku
boxPolygon = [1, 1;...                                % top-left
        size(boxImage, 2), 1;...                      % top-right
        size(boxImage, 2), size(boxImage, 1);... % bottom-right
        1, size(boxImage, 1);...                      % bottom-left
        1, 1];                        % top-left znovu
newBoxPolygon = transformPointsForward(tform, boxPolygon);
figure;
imshow(sceneImage);
hold on;
line(newBoxPolygon(:, 1), newBoxPolygon(:, 2), 'Color', 'y','LineWidth',10);
title('Detected Box');
```



Panorama

- Spojování panoramatických obrázků
  - na základě spočítaných příznaků



Načtení a zobrazení snímků

```
fo = 'img\';
```

- Dřívější přístup:

```
fi = dir(fullfile(fo, '*.jpg'));
numFi = length(fi);
for idx = 1:numFi
    I(:,:,:,idx) = imread([fo fi(idx).name]);
end
n = ceil(sqrt(numFi));
m = n - 1;
if m*n < numFi
    m = n;
end
for idx = 1:length(fi)
    subplot(m,n,idx)
    imshow(I(:,:,:,idx));
end
```

- Pomocí Datastore:

```
imgs = imageDatastore(fo);
montage(imgs.Files)
```

Registrace jednotlivých párů

- Vypočítat a porovnat příznaky mezi I_n a I_{n-1}
- Odhad geometrické transformace T_n , která odpovídá transformaci bodů I_n na I_{n-1}
- Vypočet finální transformace, která mapuje I_n na panoramatický snímek jako $T_n * T_{n-11} * \cdots * T_1$

Registrace jednotlivých párů

- Načtení prvního snímku

```
I = readimage(imgs, 1);
```

- Spočtení příznaků pro I_1

```
Ig = rgb2gray(I);
points = detectSURFFeatures(Ig);
[features, points] = extractFeatures(Ig, points);
```

- Inicializace všech transformací do matice identity.
  - Kamera je blízko scéně, proto použijeme projektivní transformaci. Pokud by byla scéna dále od kamery, tak by stačila afinní transformace.

```
numImgs = numel(imgs.Files);
tforms(numImgs) = projective2d(eye(3));
```

- Proměnná pro zapamatování velikosti obrázků

```
imgSize = zeros(numImages,2);
```

Registrace jednotlivých párů

Registrace jednotlivých párů

Iterace přes všechny páry

```
for idx = 2:numImgs
```

Body a příznaky obrázku I_{n-1}

```
   pointsPrev = points;
   featuresPrev = features;
```

Načtení I_n a převedení do šedotónu

```
   I = readimage(imgs, idx);
   Ig = rgb2gray(I);
```

Uložení velikosti obrázku I_n

```
   imgSize(idx,:) = size(Ig);
```

Detekce a extrakce SURF příznaků pro I_n

```
   points = detectSURFFeatures(Ig);
   [features, points] = extractFeatures(Ig, points);
```

Nalezení korespondencí mezi I_n a I_{n-1}

```
   indexPairs = matchFeatures(features, featuresPrev, 'Unique', true);
   matchedPoints = points(indexPairs(:,1), :);
   matchedPointsPrev = pointsPrev(indexPairs(:,2), :);
```

Odhad transformace mezi I_n a I_{n-1}

```
   tforms(idx) = estimateGeometricTransform(matchedPoints, matchedPointsPrev,...
       'projective', 'Confidence', 99.9, 'MaxNumTrials', 2000);
```

Spočtení $T_n * T_{n-1} * \cdots * T_1$

```
   tforms(idx).T = tforms(idx).T * tforms(idx-1).T;
end
```

Přepočítání transformací

- Všechny T_i byly napočítány relativně k prvnímu obrázku

- Je potřeba to přepočítat ke středovému obrázku
  - Stačí invertovat transformaci středového obrázku a aplikovat tuto transformaci na všechny ostatní transformace

- Jak nalézt obraz, který je zhruba ve středu scény?
  - Pomocí metody **`outputLimits`**

```
for i = 1:numel(tforms)
  [xlim(i,:),ylim(i,:)]=outputLimits(tforms(i),[1 imgSize(i,2)],[1 imgSize(i,1)]);
end
```

Přepočítání transformací

- Inicializace obrázku kam budeme mapovat jednotlivé body ze všech zpracovávaných pohledů:

```
avgXLim = mean(xlim, 2);
[~, idx] = sort(avgXLim);
centerIdx = floor((numel(tforms)+1)/2);
centerImgIdx = idx(centerIdx);
Tinv = invert(tforms(centerImgIdx));
```

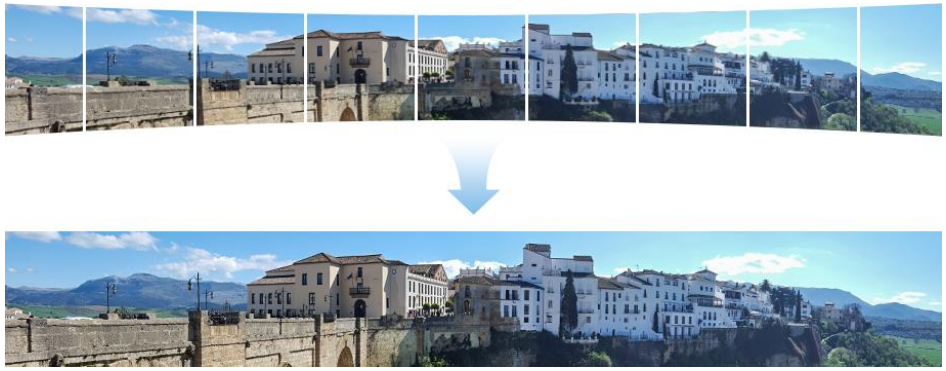
- Nyní použijeme inverzní transformaci středového obrazu na všechny ostatní:

```
for i = 1:numel(tforms)
    tforms(i).T = tforms(i).T * Tinv.T;
end
```

Inicializace výstupního obrázku

- Spočítáme průměrnou limitu ze všech transformací a díky ní najdeme obrázek, který je ve středu.
  - Používáme pouze limity X, protože předpokládáme, že scéna je horizontální

```
for i = 1:numel(tforms)
  [xlim(i,:),ylim(i,:)]=outputLimits(tforms(i),[1 imgSize(i,2)],[1 imgSize(i,1)]);
end

maxImageSize = max(imgSize);
```



Nalezení minima a maxima výstupních limitů

```
xMin = min([1; xlim(:)]);
xMax = max([maxImageSize(2); xlim(:)]);

yMin = min([1; ylim(:)]);
yMax = max([maxImageSize(1); ylim(:)]);
```

Spočtení rozměrů finálního obrázku

```
width  = round(xMax - xMin);
height = round(yMax - yMin);
```

Inicializace odpovídající 3-kanálového obrázku

```
panorama = zeros([height width 3], 'like', I);
```

Finální výpočet Panoramatu

- Funkce k vytvoření finálního obrázku:
  - **`imwarp`** k mapování obrázků do panoramatu
  - **`vision.AlphaBlender`** k překrytí obrázků dohromady
  - **`imref2d`** k vytvoření referenčního 2D prostorového objektu definujícího velikost panoramatu

```
blender = vision.AlphaBlender('Operation', 'Binary mask', ...
    'MaskSource', 'Input port');

xLimits = [xMin xMax];
yLimits = [yMin yMax];
panoramaView = imref2d([height width], xLimits, yLimits);
```

Finální výpočet Panoramatu

- Mapování všech snímků do finálního obrázku:

```
for i = 1:numImgs
   I = readimage(imgs, i);
```

Transformace obrázku do panoramatu

```
   warpedImg = imwarp(I, tforms(i), 'OutputView', panoramaView);
```



Vygenerování binární masky

```
   mask = imwarp(true(size(I,1),size(I,2)), tforms(i),...
            'OutputView', panoramaView);
```



Překrytí panoramatu transformovaným obrázkem

```
   panorama = step(blender, panorama, warpedImg, mask);
end
```



Finální výpočet Panoramatu

- Zobrazení výstupu:

```
fo = 'img\';
imshow(panorama)
```

Další datová sada

- Zobrazení výstupu:

```
fo = 'img\';
imgs = imageDatastore(fo);
```









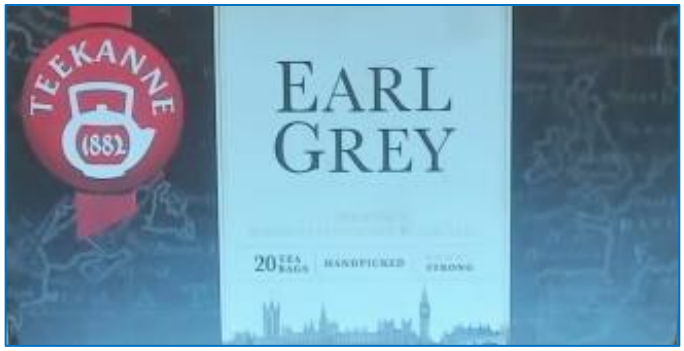

Finální výpočet Panoramatu

- Zobrazení výstupu:

```
figure;
imgs = imageDatastore(fo);
```

Detekce objektů ve scéně - video

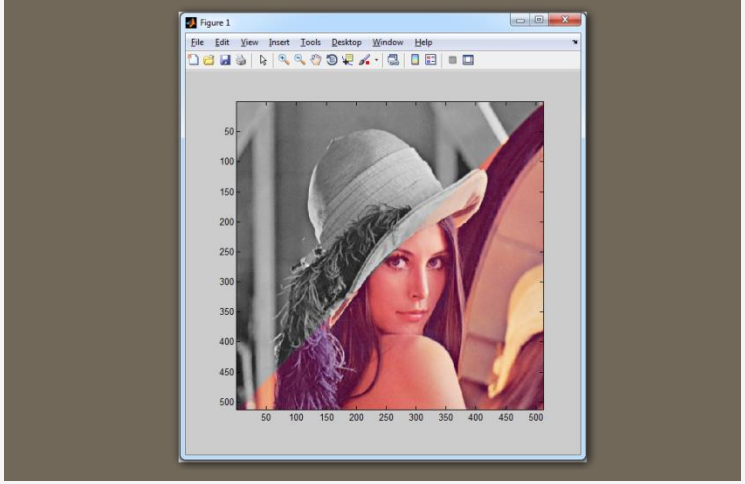
- Pomocí napočítaných SURF
- Invariantní k TRS



```
% Volba kamery
cam = webcam(2);
boxImage = rgb2gray(imread('caj.png'));
boxPolygon = [1,1;size(boxImage,2),1;size(boxImage,2),size(boxImage,1);1,size(boxImage,1);1,1];
% detekce bodu v referencnim snimku
boxPoints = detectSURFFeatures(boxImage);
[boxFeatures, boxPoints] = extractFeatures(boxImage, boxPoints);
for i = 1:1000
    temp = snapshot(cam);
    sceneImage = rgb2gray(temp);
% detekce bodu ve snimku z kamery
    scenePoints = detectSURFFeatures(sceneImage);
    [sceneFeatures, scenePoints] = extractFeatures(sceneImage, scenePoints);
    boxPairs = matchFeatures(boxFeatures, sceneFeatures);
    matchedBoxPoints = boxPoints(boxPairs(:, 1), :);
    matchedScenePoints = scenePoints(boxPairs(:, 2), :);
    imshow(sceneImage);
    if( matchedBoxPoints.Count < 3 || matchedScenePoints.Count < 3 )
    newBoxPolygon = boxPolygon;
    else
        try
            [tform, inlierBoxPoints, inlierIPoints] = ...
                estimateGeometricTransform(matchedBoxPoints, matchedScenePoints, 'affine');
            newBoxPolygon = transformPointsForward(tform, boxPolygon);
            dl = line(newBoxPolygon(:, 1), newBoxPolygon(:, 2), 'Color', 'y','LineWidth',10);
        catch
            disp('Nepodarilo se naji inliners! \n')
        end
    end
    pause(0.1);
end
clear('cam');
```

Detekce objektů ve scéně - video

- Pomocí napočítaných SURF
- Invariantní k TRS



Děkuji za pozornost