**Aula Prática 2**

Lista com alguns comandos no Python (e seus pacotes) para alguns dos conteúdos vistos em aula. Alguns pacotes sugeridos:

```
import numpy as np
import cv2
import skimage
```

1. Transformada Hough:
   - `cv2.HoughLines`: implementa a transformada Hough, gera uma lista de parâmetros $(d_i, \theta_i)$ relacionados às retas detectadas.
   - `cv2.HoughLinesP`: implementa uma transformada Hough Probabilística, gera uma lista de pares de pontos relacionados aos *segmentos de retas* detectados.
2. Ajuste de curvas: ajuste por mínimos quadrados pode ser implementado facilmente com operações matriciais (`numpy.linalg`). Há também rotinas já prontas:
   - `cv2.fitLine`: implementa o ajuste de uma reta dado um conjunto de pontos, podendo-se usar uma função robusta de erro (como Huber).
3. Análise de formas e contornos: várias funções para manipulação de imagens binárias, tais como
   - `cv2.distanceTransform`: calcula a transformada Distância de uma imagem binária
   - `cv2.findContours`: acha os contornos em uma imagem binária
   - `cv2.arcLength`: acha o comprimento de uma curva (ou o perímetro de um contorno se a curva for fechada)
   - `cv2.contourArea`: acha a área compreendida no interior de um contorno fechado. A área também de um objeto binário também pode ser obtida facilmente pelo somatório dos pixels.
   - `cv2.approxPolyDP`: gera uma aproximação poligonal de um contorno (dado um limiar de tolerância).
   - `skimage.morphology.medial_axis`: calcula o *medial axis* de uma imagem binária.
4. Estéreo e SfM: há várias rotinas para retificação, casamento estéreo e SfM. Alguns exemplos abaixo:
   - `cv2.StereoBM`: classe que permite realizar vários algoritmos de *Block Matching* (BM) em pares estéreo retificados.
   - `cv2.stereoRectify`: realiza a retificação de um par de imagens a partir de suas matrizes de câmera. Ver também `cv2.stereoRectifyUncalibrated` para o caso de câmeras não calibradas (pontos em correspondência são necessários)
   - `cv2.findFundamentalMat`: implementa vários algoritmos para calcular a matriz fundamental entre duas vistas com base em correspondências de pontos.
   - `cv2.findEssentialMat`: implementa vários algoritmos para calcular a matriz essencial entre duas vistas com base em correspondências de pontos e os parâmetros intrínsecos.

- `cv2.calibrateCamera`: acha os parâmetros de câmera (intrínsecos, extrínsecos, distorção radial) a partir de várias vistas de um padrão de calibração conhecido.
- O software Bundler também fornece várias ferramentas para SfM, disponível em `https://www.cs.cornell.edu/~snavely/bundler/`

5. Extração e Casamento de Keypoints
   - `cv2.SIFT_create`, `cv2.ORB_create`, `cv2.BRISK_create`: classes que permitem a detecção de keypoints usando SIFT, ORB e BRISK, respectivamente.
   - `cv2.BFMatches`, `cv2.FlannBasedMatcher`: classes para o casamento de keypoints por força bruta (*Brute Force*) e por vizinhos próximos (*Approximate Nearest Neighbors*).
6. Detecção de Objetos
   - `cv2.CascadeClassifier`: detecção de objetos usando a conhecida técnica de Viola & Jones, baseada em Haar-like features e um classificador em cascata. Permite o treinamento de novas categorias.

Roteiro da aula prática: considere os Colab notebooks `.ipynb` em anexo:

1. `exemple_hough.ipynb` ilustra o uso da transformada Hough para achar linhas em uma imagem (focado na detecção dos eixos coordenados em um gráfico 2D anotado à mão em uma tela sensível ao toque).
2. `exemple_fundamental.py` ilustra o cálculo da matriz fundamental com base no casamento de pontos (obtido de forma automática com técnicas de *keypoint detection and matching*).

**Exercício 1.** Considere uma imagem contendo o desenho de um gráfico de uma função (feito à mão) junto com os eixos coordenados. O objetivo deste exercício é identificar os eixos coordenados e alinhar o gráfico na direção horizontal. Estude e execute o notebook `exemple_hough.ipynb` para a imagem `desenho.jpg` (fornecida), e avalie os resultados. Teste para outras imagens de "gráficos", e verifique se de fato os eixos seguem sendo detectados. Avalie os parâmetros de entrada da função `cv2.HoughLines` (como o *step* em $\theta, d$, e o limiar de detecção).

**Exercicio 2.** Este exercício ilustra o cálculo da matriz fundamental entre duas vistas da mesma cena, realizando o casamento de pontos em correspondência através de *keypoint matching*. Execute o notebook `exemple_fundamental.ipynb` para os pares estéro retificados *Tsukuba* e *Cones* do dataset Middlebury, fornecidos com o script, e verifique se as linhas epipolares estimadas se comportam como o esperado. Varie o parâmetro `num_match`, que controla a quantidade de pontos casados usados na estimativa da matriz fundamental. Você também pode testar outros descritores, como SIFT e BRISK, e usar pares de imagens gerados por você.