Robot Teknolojisine Giriş
BLM4830

Öğr. Grv. Furkan ÇAKMAK

Ders Tanıtım Formu ve Konular

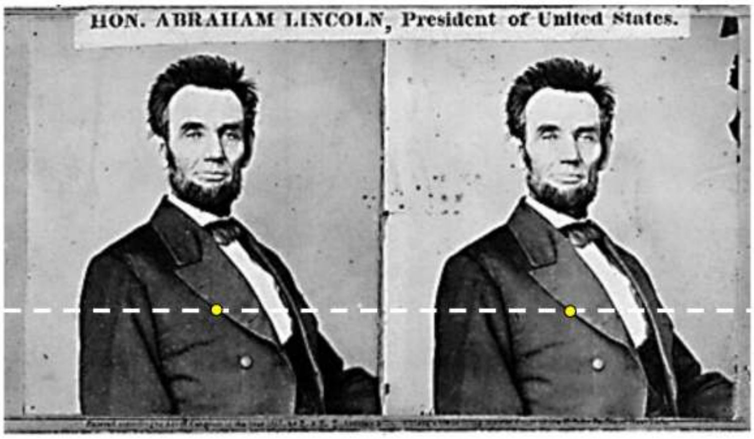
Hafta	Tarih	Konular
1	2.03.2022	Ders Tanıtımı, ROS ve Platform Tanıtımı, Robot Çeşitleri ve Robotik Konuları Başlangıcı
2	9.03.2022	Kinematik - Genel Tanımlar - Diferansiyel Sürüşlü Robot İçin Hesaplama Örnekleri
3	16.03.2022	Sensörler - Çeşitleri ve Çalışma Sistematikleri ve Uygulamaları
4	23.03.2022	Odometri ve Lokalizasyon Kavramları
5	30.03.2022	Haritalama Yöntemleri ve Uygulamaları
6	6.04.2022	Uygulama 1 (Laboratuvar)
7	13.04.2022	Navigasyon Yaklaşımları ve Uygulamaları
8	20.04.2022	Ara Sınav
9	27.04.2022	Keşif Yaklaşımları ve Uygulamaları
10	4.05.2022	Tatil - Ramazan Bayramı Arifesi
11	11.05.2022	Robot Üzerinden Görüntü İşleme Teknikleri
12	18.05.2022	Uygulama 1-2 (Laboratuvar)
13	25.05.2022	3B Haritalama Yöntemleri
14	1.06.2022	Proje Sunumları

Structure from stereo

- Recover the structure (depth of points) of the environment from two images taken by two distinct cameras, whose relative position and orientation is known
- Two major problems
  - Correspondence problem
  - 3D reconstruction
- Correspondence : which point in first image matches with which point in second image
- 3D construction : calculate depth from matching points

Structure from stereo – Horizantally perfectly aligned cameras - Matching

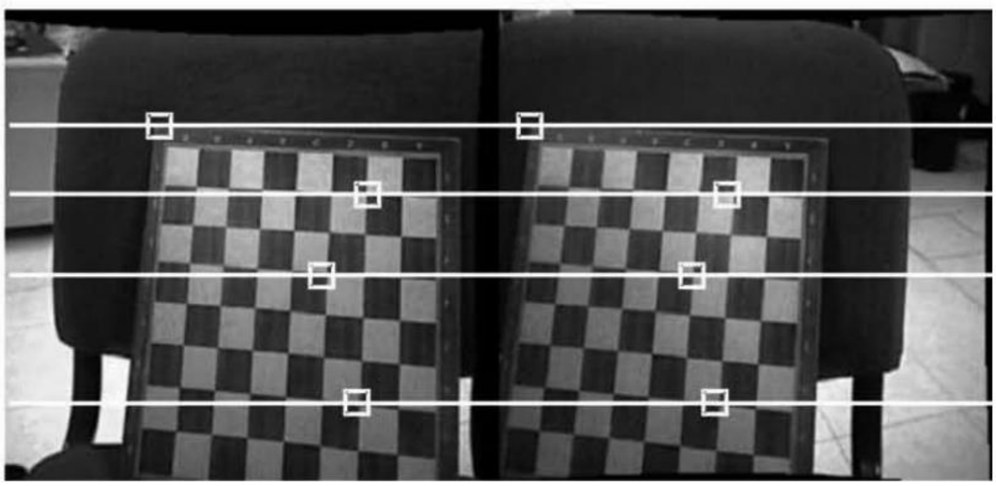
For each epipolar line
    For each pixel in the left image
        compare with every pixel on same epipolar line in right image
        pick pixel with minimum match cost



Structure from stereo – General case - Epipolar geometry

Structure from stereo – General case - Epipolar rectification

Original
Lens distortion
T&R
Horizantal match



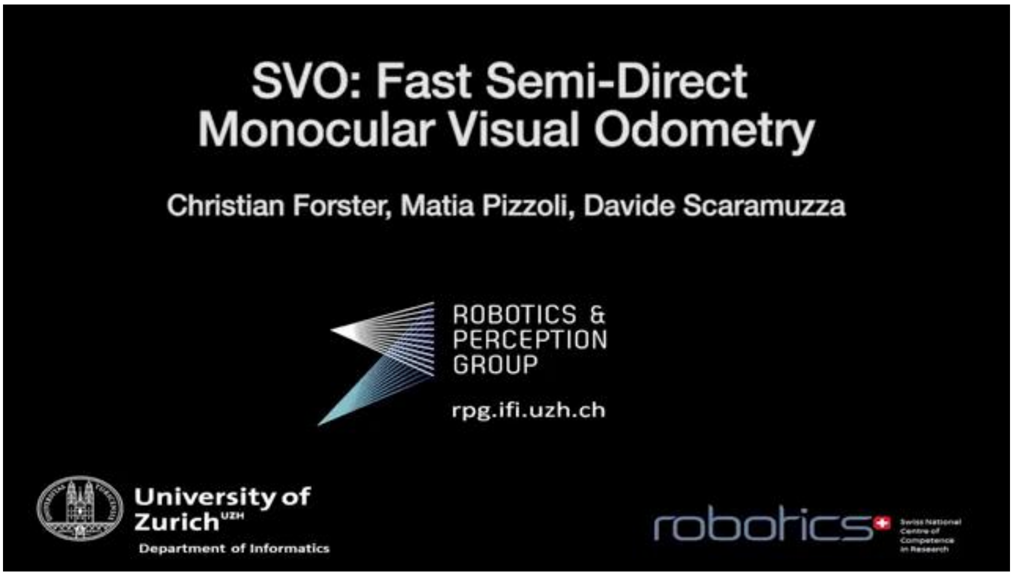
Structure from motion

- Process two images taken with the same camera at different times and from different unknown positions
- Both motion and the structure should be estimated

Structure from motion

Structure from motion - Visual odometry

https://www.youtube.com/watch?v=2YnIMfw6bJY



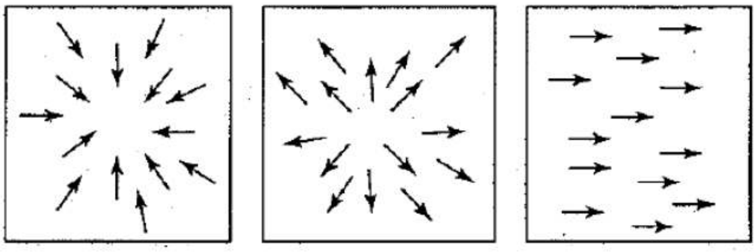
Motion field and Optical flow

- Motion field : If a point in the environment moves with velocity vo, this correspondes to a motion with velocity vi in image plane
- Optical flow : If light source moves the brightness pattern in image moves too

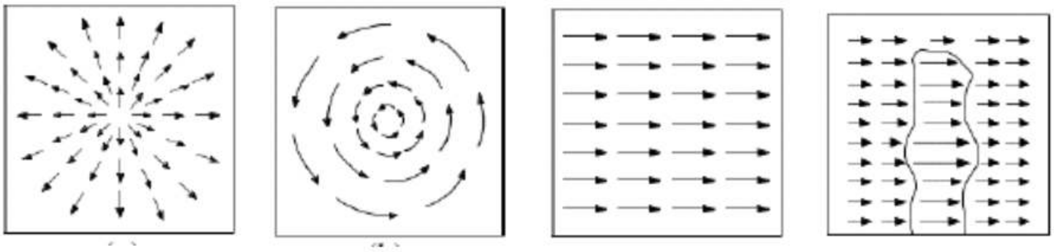
- Motion field : real World 3d motion
- Optical flow : Projection of the motion field onto the 2D image

- Estimate motion by optical flow

Motion Field



Zoom out **Zoom in** **Pan right to left**



Forward motion

Rotation

Horizontal translation

Closer objects appear to move faster!!

Color Tracking

- Color thresholding

$$R_{min} < r < R_{max} \ \ and \ \ G_{min} < g < G_{max} \ \ and \ \ B_{min} < b < B_{max}$$

- Color Segmentation
  - Adaptive thresholding
  - K-means clustering

  → Floor plane extraction

Image Filtering

- Frequency domain filtering
- Spatial Filtering

$$I'(x, y) = \sum^{a} \sum^{b} w(s, t) \,.\, I(x - s, y - t)$$

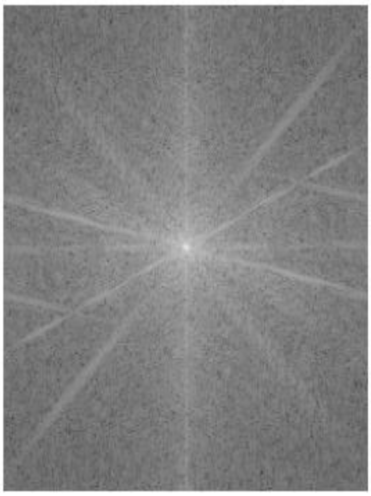
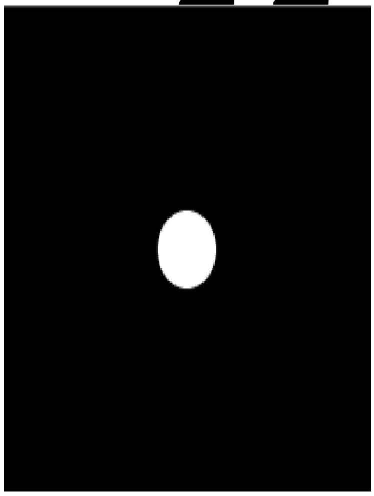
   

Image Filtering – Spatial Filtering - Smoothing Filters

- Median Filter : Non-linear filter

- Mean Filter : $w = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

- Gauss Filter : $G_\sigma(x, y) = \eta e^{-\frac{x^2+y^2}{2\sigma^2}}$ → $w = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

Image Filtering – Spatial Filtering – Edge Detection

- Roberts Filter : $r_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, r_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, |G| = \sqrt{r_1^2 + r_2^2}$
- Prewitt Filter : $p_1 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, p_2 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix},$
  $|G| = \sqrt{p_1^2 + p_2^2}, \theta = \operatorname{atan}\left(\frac{p_1}{p_2}\right)$
- Sobel Filter : $s_1 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, s_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix},$
  $|G| = \sqrt{s_1^2 + s_2^2}, \theta = \operatorname{atan}\left(\frac{s_1}{s_2}\right)$

Straight Edge Extraction – Hough Transform

- A line is defined as $y = mx + b$
- Initialize a 2D matrix A with axes carry possible values of m and b
- For every edge pixel (xp,yp) loop over all m and b vaules
- If yp=m.xp+b then A[m,b]++
- Largest value of A at m and b means a straight line with parameters m and b

Feature Extraction

- Two strategies for sersor input evaluation
- Raw, individual sensor data → low level features
- Extract information from one or more sensor readings → high level features

Feature Extraction

- Decision on appropriate features
- Target environment : Office environment→line features, Mars Rover→ visual odometry
- Available sensors : laser range finder→ line features, sonar sensor → single point ranging
- Computational power : visual features → costly
- Environment representation : continuous rep. → line features, occupancy grid rep. → ranging

Properties of ideal feature detectors

- Repeatability
- Distinctiveness
- Localization accuracy
- Quantity of features
- Invariance
- Computational efficiency
- Robustness

Corner Detectors

- Corner : intersection of one or more edges
- Corners are with high repeatability
- Moravec corner detection : patch similarity calculation with SSD (sum of squared differences) → locally maximal SSD value shows a corner
- Harris corner detection : first order Taylor expansion for image around pixel (u,v) with changing x and y

$$I(u+x, v+y) \cong I(u,v) + I_x(u,v)x + I_y(u,v)y$$

Harris Corner Detector

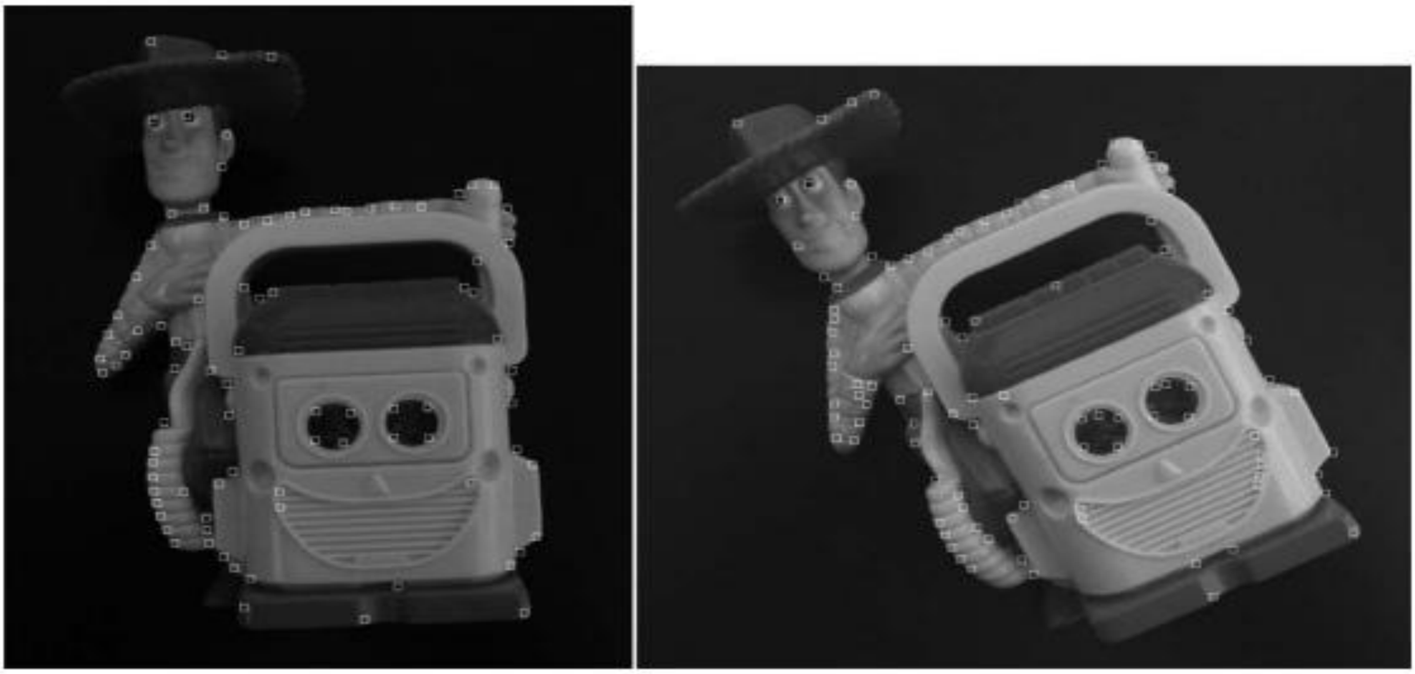
$$SSD(x,y) \cong \sum_u \sum_v \left[I_x(u,v)x + I_y(u,v)y \right]^2$$

$$= [x \quad y] \sum_u \sum_v \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$= [x \quad y] R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R \begin{bmatrix} x \\ y \end{bmatrix}$$

R: orientation, λ: change rate

Harris Corner Detector

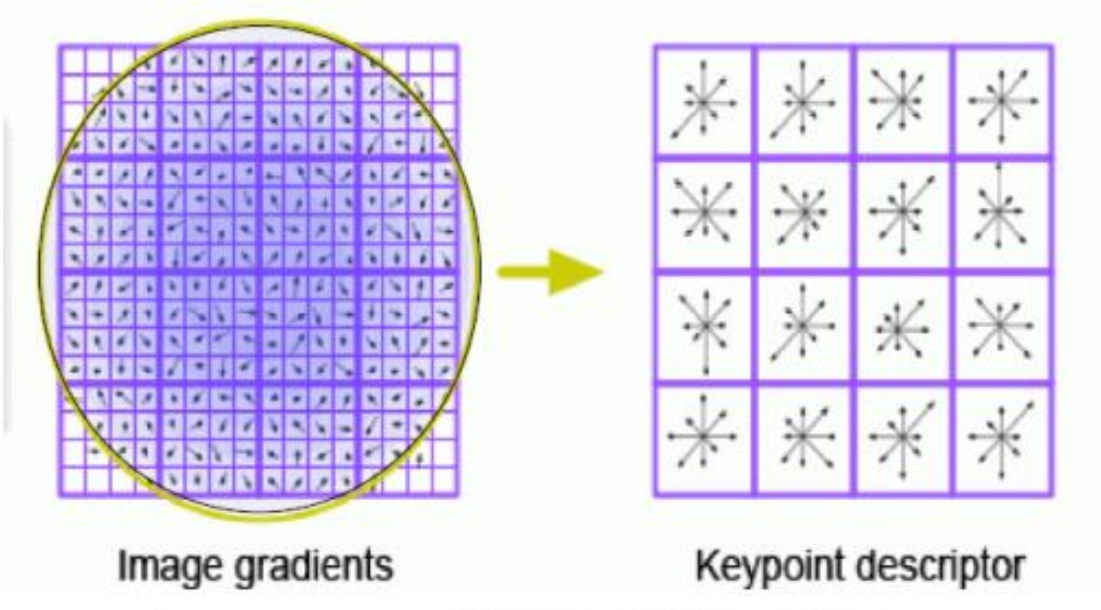


Blob Detectors

- Blob: an image pattern that differs from its immediate neighborhood in terms of intensty, color and texture
- Location accuracy is smaller than a corner
- Scale and shape accuracy is better defined compared to a corner

SIFT Features

Gaussian blurred images at different scales.
Difference of Gaussian images.
Keypoint selection as local maxima or minima of the DoG images
Orientation assignment
Keypoint descriptor



sensor_msgs/LaserScan

```
# Single scan from a planar laser range-finder
#
# If you have another ranging device with different behavior (e.g. a sonar
# array), please find or create a different message, since applications
# will make fairly laser-specific assumptions about this data

Header header           # timestamp in the header is the acquisition time of
                        # the first ray in the scan.
                        #
                        # in frame frame_id, angles are measured around
                        # the positive Z axis (counterclockwise, if Z is up)
                        # with zero angle being forward along the x axis

float32 angle_min        # start angle of the scan [rad]
float32 angle_max        # end angle of the scan [rad]
float32 angle_increment  # angular distance between measurements [rad]

float32 time_increment   # time between measurements [seconds] - if your scanner
                        # is moving, this will be used in interpolating position
                        # of 3d points
float32 scan_time        # time between scans [seconds]

float32 range_min        # minimum range value [m]
float32 range_max        # maximum range value [m]

float32[] ranges         # range data [m] (Note: values < range_min or > range_max should be discarded)
float32[] intensities    # intensity data [device-specific units].  If your
                        # device does not provide intensities, please leave
                        # the array empty.
```

sensor_msgs/Image   RGB or Depth

```
# This message contains an uncompressed image
# (0, 0) is at top-left corner of image
#

Header header       # Header timestamp should be acquisition time of image
                    # Header frame_id should be optical frame of camera
                    # origin of frame should be optical center of cameara
                    # +x should point to the right in the image
                    # +y should point down in the image
                    # +z should point into to plane of the image
                    # If the frame_id here and the frame_id of the CameraInfo
                    # message associated with the image conflict
                    # the behavior is undefined

uint32 height       # image height, that is, number of rows
uint32 width        # image width, that is, number of columns

# The legal values for encoding are in file src/image_encodings.cpp
# If you want to standardize a new string format, join
# ros-users@lists.sourceforge.net and send an email proposing a new encoding.

string encoding     # Encoding of pixels -- channel meaning, ordering, size
                    # taken from the list of strings in include/sensor_msgs/image_encodings.h

uint8 is_bigendian  # is this data bigendian?
uint32 step         # Full row length in bytes
uint8[] data        # actual matrix data, size is (step * rows)
```

sensor_msgs/PointCloud

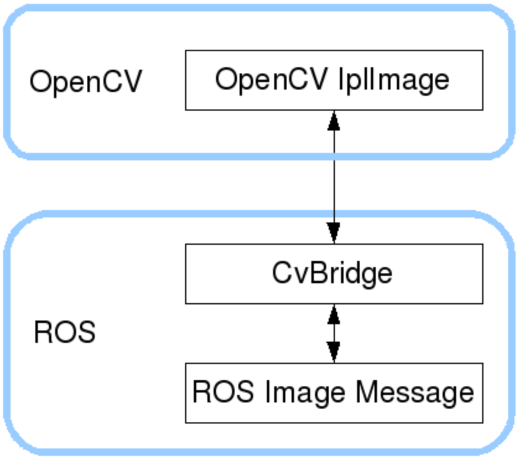
```
# This message holds a collection of 3d points, plus optional additional
# information about each point.

# Time of sensor data acquisition, coordinate frame ID.
Header header

# Array of 3d points. Each Point32 should be interpreted as a 3d point
# in the frame given in the header.
geometry_msgs/Point32[] points

# Each channel should have the same number of elements as points array,
# and the data in each channel should correspond 1:1 with each point.
# Channel names in common practice are listed in ChannelFloat32.msg.
ChannelFloat32[] channels
```

cv_bridge



Sabırla Dinlediğiniz İçin Teşekkürler