Perception for Autonomous Systems

Lecture 1 - Image Processing (01/02/2021)

Introduction Questions

1. What is "Perception"?

2. What is an "Autonomous System"?

3. Why Autonomous Systems need Perception?

Outline/Content:

- What is Image Processing?
- Image
- Color
- Linear Filtering
- Non-linear Filters / Thresholding
- Morphology
- Connrected Components Analysis
- Summary

> Topics and Reading Sources:
>
> - Color
> - Book A, Chapter 2.3.2
> - Linear Filtering
> - Book A, Chapters 3.1, 3.2
> - Non-linear Filters, Morphology, Thresholding and Connected Components Analysis
> - Book A, Chapter 3.3

What is Image Processing?

Image processing are operations we perform on an image to change or enhance it and make it suitable for further analysis. So that useful information can be highlighted or get extracted from it.

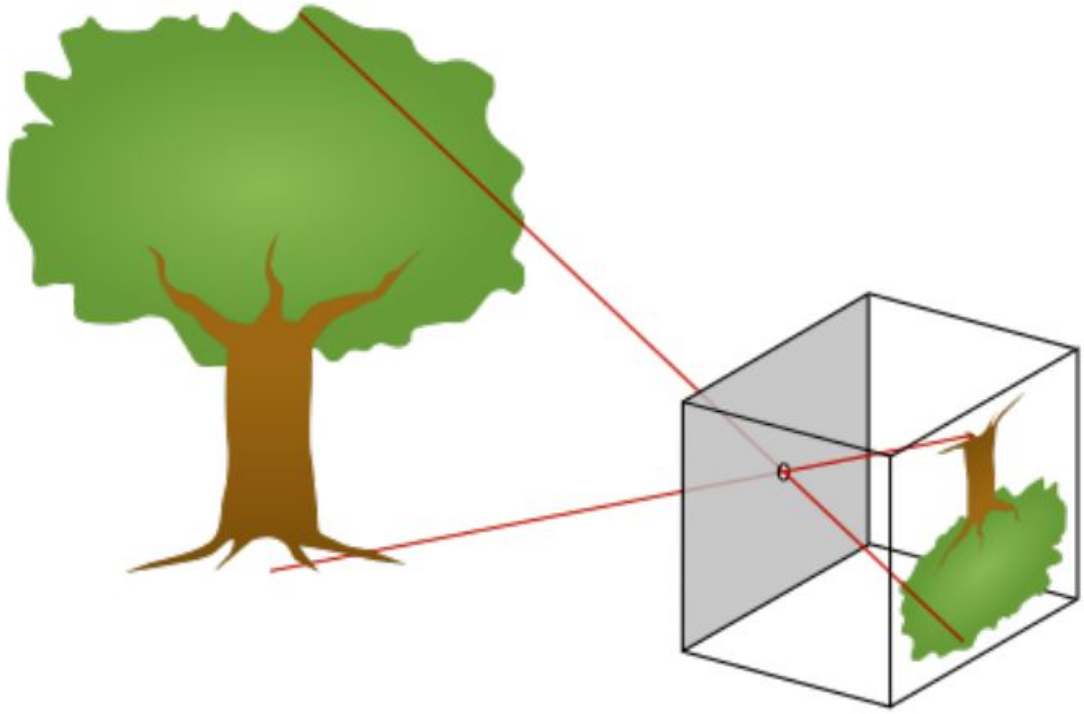
> MAYBE ADD MORE POINTS HERE

Image

Pinhole model **Youtube Link**

The pinhole model describes the process of reducing the incidence of light to such an extent that the section of the image in front of you can be seen (upside down). This process is used in all given cameras and is known

as "aperture".



Pixels

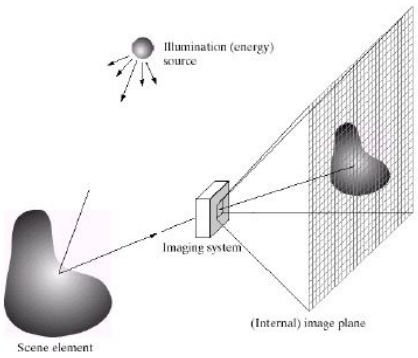
Pixels are the raw building blocks of an image. Each image consists of a set of pixels. There is no finer granularity than the pixel. Normally, a pixel is considered to be the "color" or the "intensity" of the light that appears at a particular location in our image.

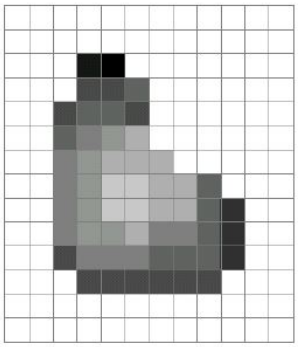
Most pixels are represented in two ways:

- Grayscale: single channel
- Color: three channels

What is a (digital) image?

The resolution of an image is given by the representation of columns_number (width) x rows_number (height) of pixels. For example, an image with a resolution of 1,000 x 750 is 1,000 pixels wide and 750 pixels high. We can think of an image as a (multi-dimensional) matrix with the form W x H x D (dimensions/channels). In this case, our matrix has 1,000 columns with 750 rows. Overall, there are 1,000 x 750 = 750,000 total pixels in this image. Each cell can take discrete/integer values (quantized samples ranging from 0 to 255).





255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

> HINT: Image processing libraries such as OpenCV and scikit-image represent RGB images as multidimensional NumPy arrays with shape (height, width, depth). So, height and width are reversed.

Color

Bayer Color filter

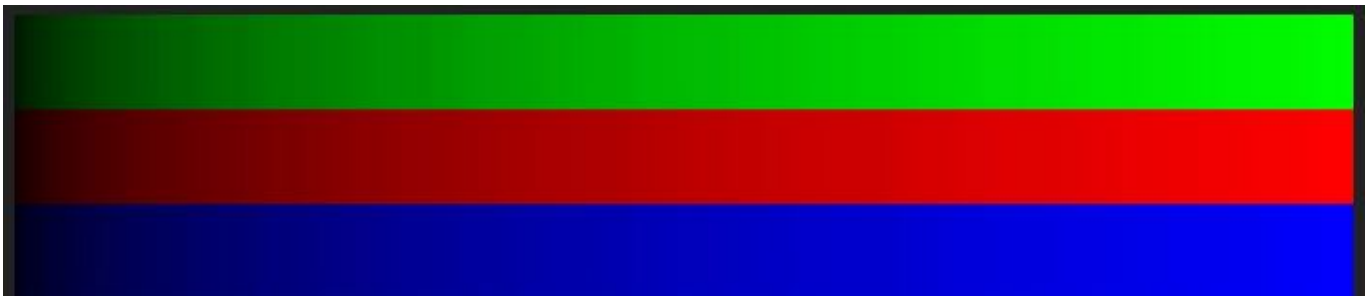
The Bayer pattern places green filters over half of the sensors (in a checkerboard pattern), and red and blue filters over the remaining ones. The reason that there are twice as many green filters as red and blue is because the luminance signal is mostly determined by green values and the visual system is much more sensitive to high frequency detail in luminance than in chrominance.

Grayscale

In a grayscale image, each pixel is a scalar (single-channel) value between 0 and 255, where zero corresponds to "black" and 255 to "white". Values in between are different shades of gray, with values closer to 0 being darker and values closer to 255 being lighter.

RGB

In a colored image, each pixel is a representation of three 2D images: R(x,y), G(x,y), B(x,y) with values ranging from 0 to 255. The values of a specific pixel (x,y) in the 3 images R, G, B describe the red-ness, green-ness and blue-ness of that particular pixel. Meaning 0 indicates the absence of the color, therefore "black" whereas 255 indicates the full saturation of that color. Given our three Red, Green, and Blue values, we can combine them into an RGB tuple in the form (red, green, blue). This tuple represents a given color in the RGB color space.



RGB Color Space (additive color space)

- A image consists of 25% red, 50% green and 25% blue pixels
- all colors can be reproduced by mixing Red, Green and Blue
- the more of each color is added, the brighter the pixel becomes and closer to white
- There is a maximum of 16.777.216 unique colors in the RGB color space (256 x 256 x 256)

Drawbacks of thr RGB Color Space:

- It's additive nature makes it a bit unintuitive for humans to easily define shades of color without using a "color picker" tool
- It doesn't mimic how humans perceive color

However, despite these drawbacks, nearly all images you'll work with will be represented (at least initally) in the RGB color space.

> Many other Color Spaces exist e.g. L*a*b* color space or HSI/HSV (Hue-Saturation-Value) Color spaces.

Linear Filtering

Questions to ask are:

- What are Linear Filters in Image Processing?
- What are they used for?

Filters

Filters form a new image whose pixels are a combination of the original pixels - Why?

- To get useful information from images
  - E.g., extract edges or contours (to understand shape)
- To enhance the image
  - E.g., blur to remove noise
  - E.g., sharpen to enhance image

Application of Filters: When using a filter we have two matrices. One big matrix representing the image and a small kernel or convolution matrix representing the filter of choice. Essentially, this tiny kernel sits on top of the image matrix and slides from left-to-right and top-to-bottom, applying a mathematical operation (i.e., a convolution) at each (x;y)-coordinate of the original image.

> SIDE NOTE: The Initial starting position within the pixel grid is the left upper corner at (0, 0).

Usually all the kernel cells need to have valid values. Meaning, we don't start at the corner of the images. This will introduce some loss as we are shrinking the image. There are ways to counter that, like zero padding etc.

Kernel

As already mentioned before we're sliding the kernel from left-to-right and top-to-bottom along the original image. At each (x,y)-coordinate of the original image, we stop and examine the neighborhood of pixels located at the center of the image kernel. We then take this neighborhood of pixels, convolve them with the kernel, and obtain a single output value. The output value is stored in the output image at the same (x,y)-coordinates as the center of the kernel.

Odd kernel sizes are required to ensure that there is a valid integer (x,y)-coordinate at the center of the image. Let's take a 3x3 matrix for example. Here you have the center at (1,1 => valid integer) if we start at the left upper corner. Whereas a 2x2 matrix mathematically would have it's center point at (0.5, 0.5) which is no valid expression. And visually it's even more obvious that there is no center point in a 2x2 matrix.

Convolution and cross-correlation

Application of Convolution:

- Blurring:
  - average/mean smoothing
  - gaussian smoothing
- Edge detection:
  - Laplacian
  - Sobel
  - Scharr
  - Prewitt
- Sharpen
- other filters

> SIDE NOTE: To sharpen an image, you blur the image and substract it from the original image and then add that result onto the original image again.

Convolutions are one of the most critical, fundamental building-blocks in computer vision and image processing. They're actually quite easy to understand. It's an element-wise multiplication of two matrices followed by a sum.

What's the difference between Convolution and cross-correlation?

Mathematically it's the same equation where simply a sign change took place. This effects the way we access the coordinates of the image (i.e., we don't have to "flip" the kernel relative to the input when applying cross-correlation).

Therefore, speaking about convolution altough using cross-correlation is by design.

Cross correlation

$$S[f] = w \otimes f$$

$$S[f](m,n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f(m+i, n+j)$$

Convolution

$$S[f] = w * f$$

$$S[f](m,n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f(m-i, n-j)$$

Putting everything together

A convolution requires three components:

1. An input image.
2. A kernel matrix that we are going to apply to the input image.
3. An output image to store the output of the image convolved with the kernel.

Convolution (or cross-correlation) is actually very easy. All we need to do is:

1. Select an (x;y)-coordinate from the original image.
2. Place the center of the kernel at this (x;y)-coordinate.
3. Take the element-wise multiplication of the input image region and the kernel, then sum up the values of these multiplication operations into a single value. The sum of these multiplications is called the kernel output.
4. Use the same (x;y)-coordinates from Step #1, but this time, store the kernel output at the same (x;y)-location as the output image.

Below you can find an example of convolving (denoted mathematically as the **star** operator) a 3x3 region of an image with a 3x3 kernel used for blurring:

$$O_{i,j} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \star \begin{bmatrix}93 & 139 & 101\\ 26 & 252 & 196\\ 135 & 230 & 18\end{bmatrix} = \begin{bmatrix}1/9 \times 93 & 1/9 \times 139 & 1/9 \times 101\\ 1/9 \times 26 & 1/9 \times 252 & 1/9 \times 196\\ 1/9 \times 135 & 1/9 \times 230 & 1/9 \times 18\end{bmatrix}$$

Therefore,

$$O_{i,j} = \sum \begin{bmatrix}10.3 & 15.4 & 11.2\\ 2.8 & 28.0 & 21.7\\ 15.0 & 25.5 & 2.0\end{bmatrix} \approx 132.$$

After applying this convolution, we would set the pixel located at the coordinate (i; j) of the output image O to Oi; j = 132. That's all there is to it! Convolution is simply the sum of element-wise matrix multiplication between the kernel and neighborhood that the kernel covers of the input image.

Mean filtering

Objective: Introduce blur to smoothen the image and remove noise.

Take the sum of each cell within the filter size and divide by the number of cells. For example:

3x3 Filter Size

x1	x2	x3
0	0	0
10	40	0
10	0	0

=> (0 + 0 + 0 + 10 + 40 + 0 + 10 + 0 + 0) / 9 = 6.66 = 7

> Pixels only take integer values. Therefore, 6.66 is being rounded to 7.

> There are two kinds of operations to apply a filter on a pixel. There are point operators or point processes that manipulate each pixel independently of its neighbors (Brightness, contrast, color correction/transformation). And then there are area-based operators, where each new pixel value depends on the value of it's neighbors.

Non-linear filters: Thresholding

By using non-linear filters we set a threshold which acts as our if condition . For example, we could set the threshold to 200 and set everything to 255 if true and 0 otherwise. This will result in a pure black & white image.

What if the threshold could be adaptive (instead of a pre-defined value)?

- Otsu's method performs automatic image tresholding

    - The algorithm exhaustively searches for the threshold that minimizes the instra-class variance, defined as a weighted sum of variances of the two classes.

Non-linear filters: Rectification

- g(m,n) = max(f(m,n), 0)
- crucial component of modern convolutional networks

Non-linear filters

- Sometimes mean filters does not work
- mean is sensitive to outliers
- median filter: Replace pixel by median of neighbors

Morphology PyImageSearch, Youtube

The most common binary image operations are called Morphological Operations

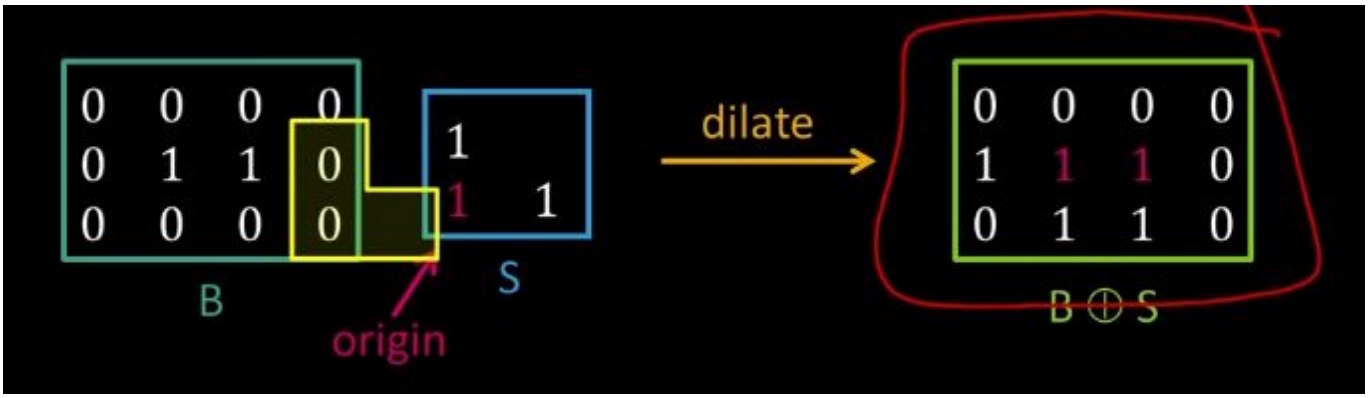
Basic Morphological Operations:

- Dilation
    - Increase white pixels, decrease black pixels
        - Case 1: Perfect match replace origin with 1
        - Case 2: Partial match replace origin with 1
        - Case 3: No match replace origin with 0
- Erosion
    - Decrease white pixels, increase black pixels
        - Case 1: Perfect match replace origin with 1
        - Case 2: Partial match replace origin with 0
        - Case 3: No match replace origin with 0
- Opening
    - Erosion then Dilation
- Closing
    - Dilation then Erosion

Structuring Element A shape mask used in basic morphological ops.

- Any shape, size that is digitally representable (Box, hexagon, disk, rectangle, etc.)
- With a defined origin

Application: The structuring element represents a pixel shape matrix with ones and usually has it's origin (highlighted) in the center of the shape. Then slide the shape from left to right, top to bottom and enable or disable pixels at the current position of the origin whenever there is any pixel match depending on the morphological operations.



> There are also Grayscale Morphological Operations, apart from Binary ones.

Connected Components Analysis

- Connected Component Analysis checks each pixel of an image for connectivity with it's neighboring pixels
- Each group of connected pixels are considered as one component and are assigned to the same label
- Connectivity is established if two neighboring pixels share same or similar intensity/color value.
- The method works on binary, grayscale, or color images.
- Different measures of connectivity arone possible (4-connectivity, or 8-connectivity are typical)

Perception for Autonomous Systems

Lecture 2 - Image Processing (08/02/2021)

Outline/Content:

- Edge Detection
  - What is an Edge?
  - Image Derivative
  - Gradient
  - Sobel
  - Laplacian
  - Canny
- Image Feature Detection and Description
- Feature Description
- Feature Matching
- Fitting Data to a Model (Handling Outliers)
  - Least Squares
  - Hough Transform
  - RANdom Sample Consensus (RANSAC)

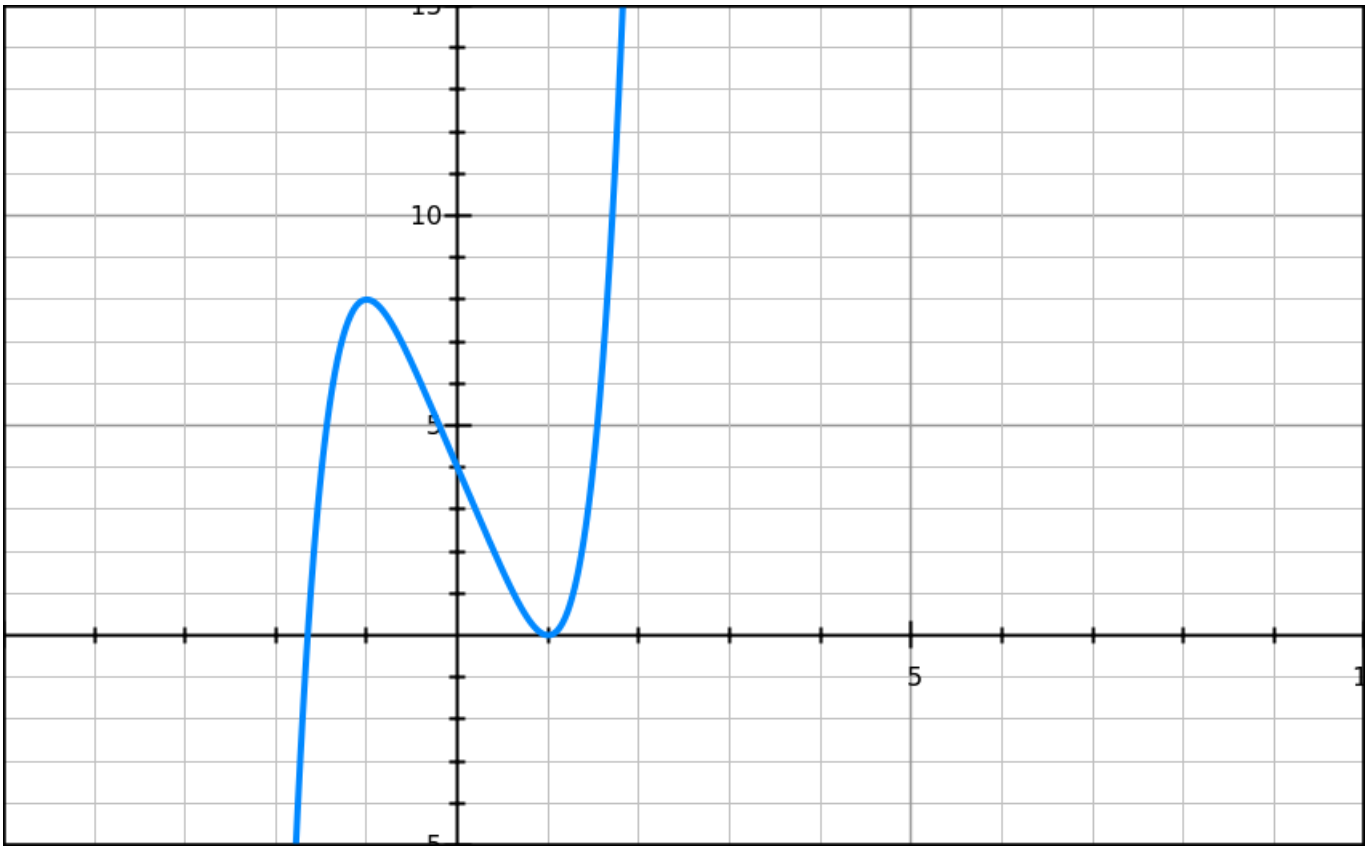
Topics and reading material

> - Feature detection
>   - Book A, Chapter 4.1.1
> - Feature description and Matching
>   - Book A, Chapters 4.1.2 - 4.1.3
> - Feature Tracking
>   - Book A, Chapter 4.1.4
> - Hough transform, RANSAC
>   - Book A, Chapter 4.3.2

Edge Detection

What is an edge?

Edge detection embodies mathematical methods to find points in an image where the brightness of pixel intensities changes distinctly. Simply put, edges occur at the boundaries between areas of different color, intensity, or texture.



> Visually speaking, do edges occur at locations of steep slopes, or equivalently, in regions of closely packed contour lines (extremas). The edge detection is calculated based on the derivative of an image.

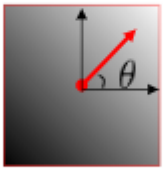
Gradient

The gradient is a vector which points in the direction of most rapid change in intensity:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Sobel

Sobel is a edge extractor operator, which is a seperable combination of a horizontal central difference (so called because the horizontal derivative is centered on the pixel) and a vertical filter (to smooth the results).

- Practically:

$$g_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad g_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- Magnitude:

$$g = \sqrt{g_x^2 + g_y^2}$$

- Orientation:

$$\Theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

Derivative Filters

Sobel

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Scharr

$$\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix} \quad \begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}$$

Prewitt

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

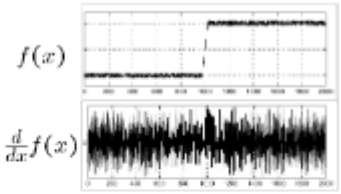
Roberts

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

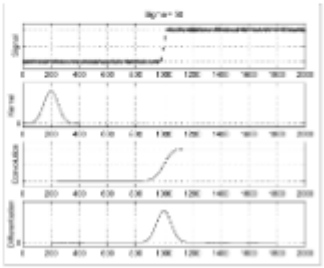
Preprocessing to Edge Detection

Taking image derivatives (required for edge detection) accentuates high frequencies and hence amplifies noise, since the proportion of noise to signal is larger at high frequencies. It is therefore prudent to smooth the image with a low-pass filter prior to computing the gradient.

- In reality derivatives are very prone to noise
  Consider the following example:



- To overcome this issue we can smooth the signal beforehand



> Because it's desirable for the response to be indepenedent of orientation it's necessary to use a circularly symmetric smoothing filter.

Laplacian of Gaussian

The Laplacian of Gaussian (LoG) is the convolution kernel of the Laplacian (gradient operator dot product with the gradient).

Canny Edge Detection

The Canny edge detector was developed way back in 1986 by John F. Canny. And it's still widely used today as one of the default edge detectors in image processing.

The Canny edge detection algorithm can be broken down into 5 steps:

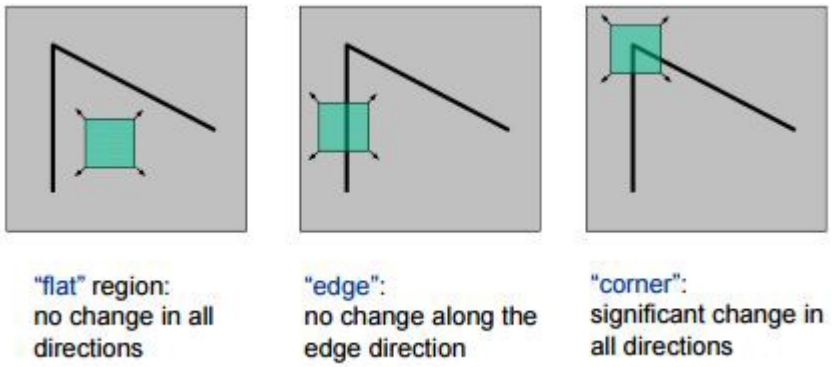
- Step 1: Smooth the image using a Gaussian filter to remove high frequency noise.
- Step 2: Compute the gradient intensity representations of the image.
- Step 3: Apply non-maximum suppression to remove "false" responses to to edge detection.
- Step 4: Apply thresholding using a lower and upper boundary on the gradient values.
- Step 5: Track edges using hysteresis by suppressing weak edges that are not connected to strong edges.

Image Features

- Feature Detection:
  - Find the most "prominent" Points (areas) in an images
- Feature Description:
  - Create a "unique" descriptor fingerprint for each Feature point
- Feature matching:
  - Find correspondences among different images

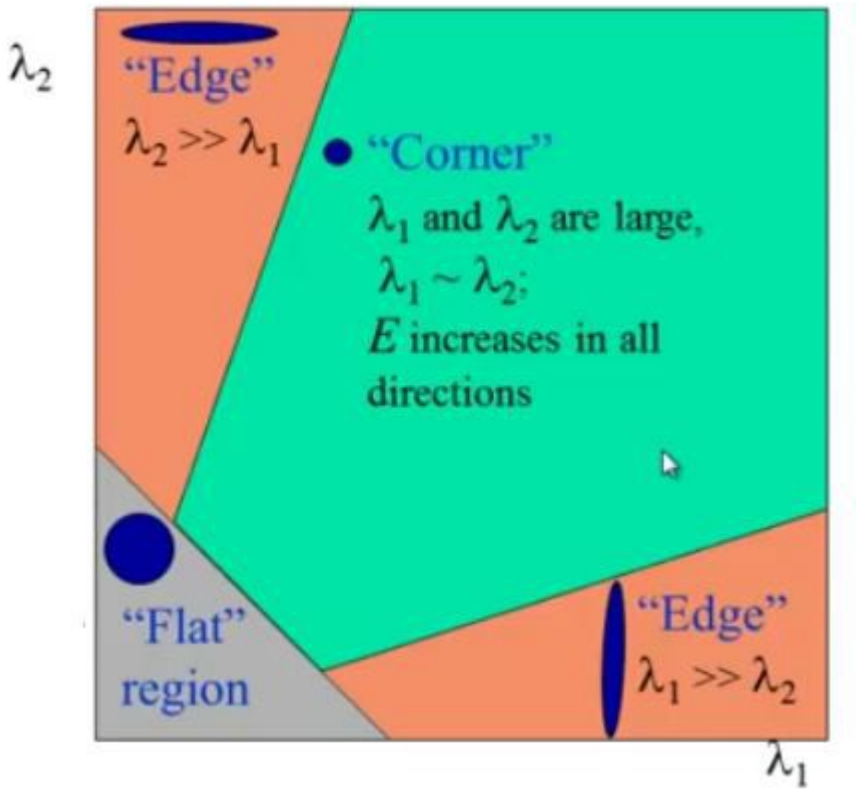
Feature Detection: Harris corner detector Medium

A corner is a point whose local neighborhood stands in two dominant and different edge directions. In other words, a corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness. Corners are the important features in the image, and they are generally termed as interest points which are invariant to translation, rotation, and illumination.



So let's understand why corners are considered better features or good for patch mapping. In the above figure, if we take the flat region then no gradient change is observed in any direction. Similarly, in the edge region, no gradient change is observed along the edge direction. So both flat region and edge region are bad

for patch matching since they not very distinctive (there are many similar patches in along edge in edge region). While in corner region we observe a significant gradient change in all direction. Due this corners are considered good for patch matching (shifting the window in any direction yield a large change in appearance) and generally more stable over the change of viewpoint.



> The values of these eigenvalues (lambdas) decide whether a region is a corner, edge or flat. **More information is to be found in the linked medium article.**

Feature Detection: Scale Space Theory Expert: Tony Lindeberg | Scale-space theory: A basic tool for analysing structures at different scales, Tony Lindeberg

A theory of multi-scale representation of sensory data developed by the image processing and computer vision communities. The purpose is to represent signals at multiple scales in such a way that fine scale structures are successively suppressed, and a scale parameter *t* is associated with each level in the multi-scale representation.

> Scale space is a collection of images having different scales, generated from a single image. Fore more information check the linked source.

The mother of Features - SIFT Article

SIFT, or Scale Invariant Feature Transform, is a feature detection algorithm in Computer Vision.

It helps to locate the local features in an image, commonly known as the 'keypoints' of the image. These keypoints are scale & rotation invariant that can be used for various computer vision applications, like image matching, object detection, scene detection, etc.

> This implies that simple corner detectors are not able to match features for images with different scales and rotations. That's when SIFT comes into play.

Characteristics of the SIFT algorithm: Contains:

- Detection
- Description
- Matching

It's robust in:

- Change of Translation
- Change in Scale
- Change in Rotation
- Change in 3D View Point
- Change in Illumination

Scale Invariant Feature Transform - SIFT

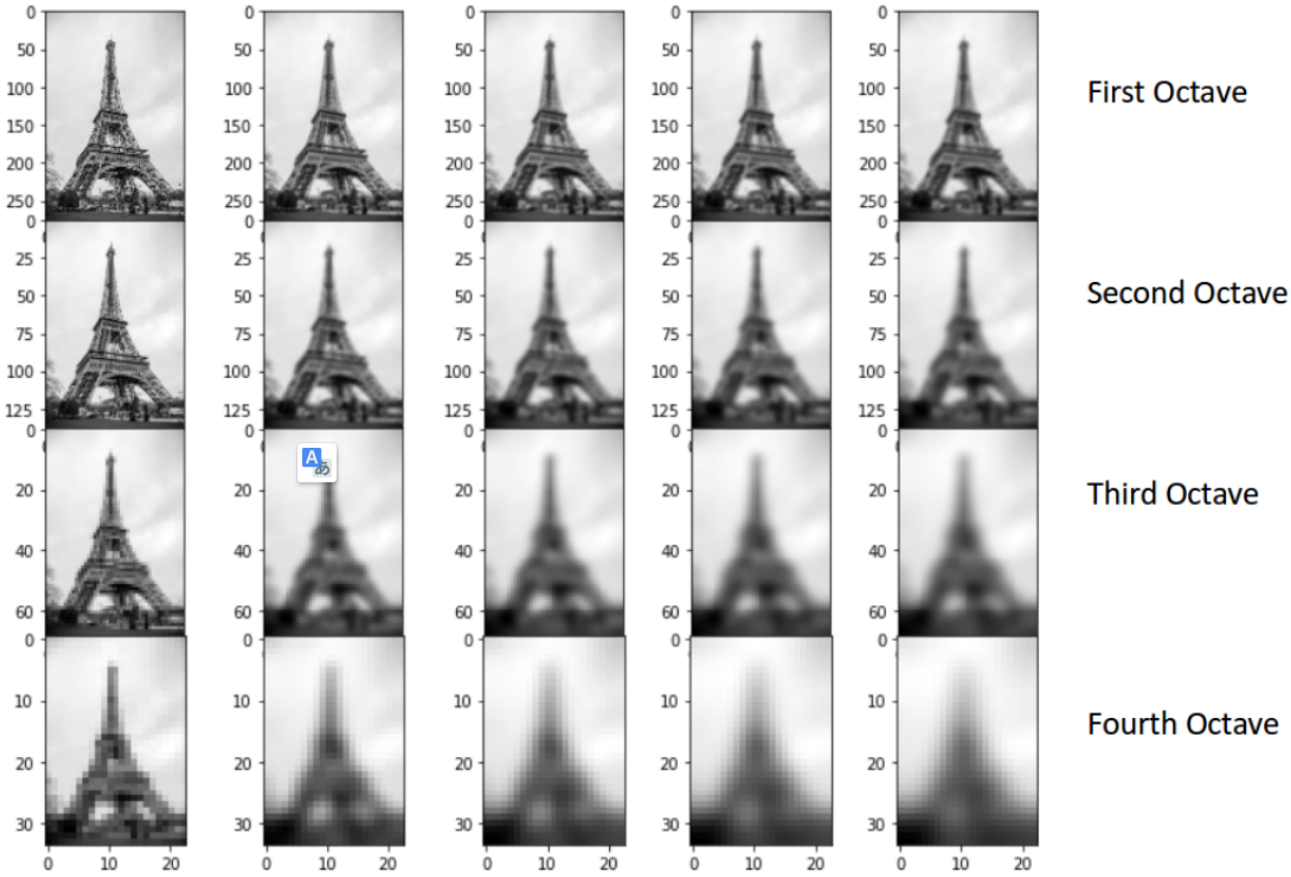
The entire process of that algorithm can be divided into 4 parts:

- Constructing a Scale Space: To make sure that features are scale-independent
- Keypoint Localisation: Identifying the suitable features or keypoints
- Orientation Assignment: Ensure the keypoints are rotation invariant
- Keypoint Descriptor: Assign a unique fingerprint to each keypoint

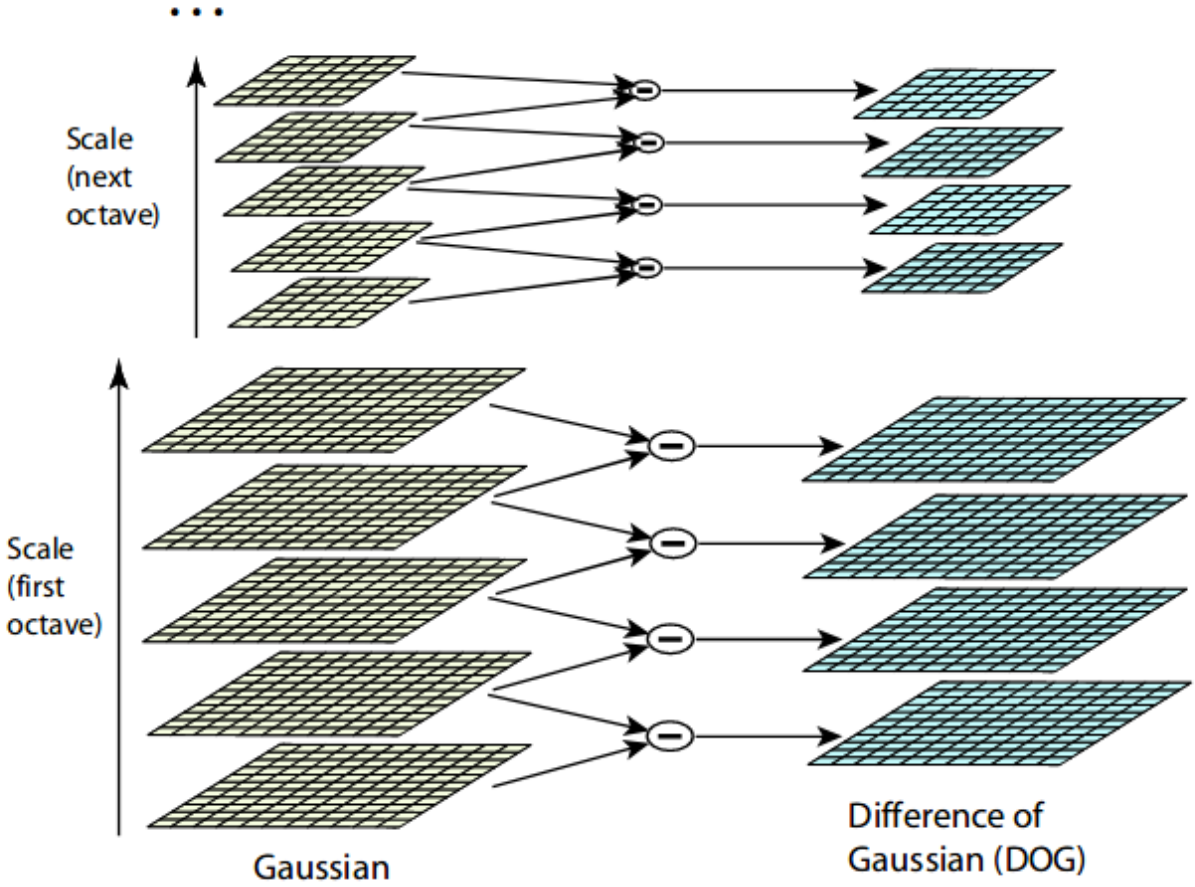
Applied Scale space

The first step of the sift algorithm is to create a number of octaves (images of different scales) and blur out all of those images repeatedly.

The ideal number of octaves should be four, and for each octave, the number of blur images should be five (recommended by the authors).



The second step is to enhance the images. DoG creates another set of images, for each octave, by subtracting every image from the previous image in the same scale. Here is a visual explanation of how DoG is implemented:



> Difference of Gaussian is a feature enhancement algorithm that involves the subtraction of one blurred version of an original image from another, less blurred version of the original.

The next step is to find the important keypoints from the image that can be used for feature matching. The idea is to find the local maxima and minima for the images. This part is divided into two steps:

- Find the local maxima and minima
- Remove low contrast keypoints (keypoint selection)

Fourth step is ensuring the orientation invariance. At this stage, we have a set of stable keypoints for the images. We will now assign an orientation to each of these keypoints so that they are invariant to rotation. We can again divide this step into two smaller steps:

- Calculate the magnitude and orientation
- Create a histogram for magnitude and orientation

> For more information check the linked article.

Feature points and areas

Feature matching algorithms:

- SIFT
- SURF
- BRISK
- FREAK
- MSER

- ORB

What applications do the Features have

More or less everything

- Motion estimation
- Localization
- Mapping
- Photogrammetry
- Image Retrieval
- Machine Learning:
    - Object Detection
    - Recognition
- Autonomous Driving
- etc.

Fitting Data to a Model (Handling Outliers)

Least Squares

Least Squares describes the process of minimizing the loss function by finding the partial derivative of that loss function.

Hough Transform Article

Hough transform is a feature extraction method for detecting simple shapes such as circles, lines etc in an image.

> The main advantage of using the Hough transform is that it is insensitive to occlusion.

It's important to use polar coordinates when applying the hough transform method, to have bounded parameters. Otherwise we could have unbounded parameters ranging from -infinity - infinity.

> Initialization of the hough space

RANSAC Article

The Random Sample Consensus (RANSAC) algorithm proposed by Fischler and Bolles is a general parameter estimation approach designed to cope with a large proportion of outliers in the input data. Its basic operations are:

- Select sample set
- Compute model
- Compute and count inliers
- Repeat until sufficiently confident

The RANSAC steps in more details are:

- Select randomly the minimum number of points required to determine the model parameters.
- Solve for the parameters of the model.

- Determine how many points from the set of all points fit with a predefined tolerance.
- If the fraction of the number of inliers over the total number of points in the set exceeds a predefined threshold, re-estimate the model parameters using all the identified inliers and terminate.
- Otherwise, repeat steps 1 through 4 (maximum of N times).

Briefly, RANSAC uniformly at random selects a subset of data samples and uses it to estimate model parameters. Then it determines the samples that are within an error tolerance of the generated model.

These samples are considered as agreed with the generated model and called as consensus set of the chosen data samples. Here, the data samples in the consensus as behaved as inliers and the rest as outliers by RANSAC. If the count of the samples in the consensus is high enough, it trains the final model of the consensus by using them.

It repeats this process for a number of iterations and returns the model that has the smallest average error among the generated models. As a randomized algorithm, RANSAC does not guarantee to find the optimal parametric model with respect to the inliers. However, the probability of reaching the optimal solution can be kept over a lower bound by assigning suitable values to algorithm parameters.

Perception for Autonomous Systems

Lecture 3 - Stereo Vision (15/02/2021)

Outline/Content:

- What is Stereo Vision?
- Stereo/Epipolar Geometry
- Rectified Stereo Case
- Depth from Stereo Matches
- Correspondence Problem
    - Dense vs Sparse Correspondence
    - Local vs Global Correspondence
    - (Dis-)-Similarity Measures
- Summary

> Topics and Reading Sources:
>
> - Optional Additional Reading Material: D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” International Journal of Computer Vision, vol. 47, no. 1-3, pp. 7–42, 2002.
> - L. Nalpantidis, G. C. Sirakoulis, and A. Gasteratos, “Review of stereo vision algorithms: from software to hardware,” International Journal of Optomechatronics, vol. 2, no. 4, pp. 435–462, 2008.

What is Stereo Vision?

Stereovision (aka., stereoscopic vision or stereopsis) describes the visual perception in three dimensions.

Two sensory inputs (both eyes for humans or two cameras for machines) which capture individual images are being used in this process. These two images have overlapping areas of common visual information, but also some unique visual information. The images are then being merged together, by matching up the similarities and adding in the small differences. By doing that we obtain a three-dimensional stereo picture.

> This implies that at least two visual based sensory inputs are needed to obtain a stereo picture.

Stereo/Epipolar Geometry Article

There are two key requirements, that need to be fulfilled to calculate the 3D structure out of two images:

- **The position of cameras:**: To obtain the positions of the cameras, it is necessary to calculate the 3D points by taking one of the camera positions as the origin. Then a calibration pattern is needed to calibrate the two cameras, which allows to find the 3D points.
- **The point correspondence**: All point correspondences between the two images must be calculated for each 3D point in the scene.

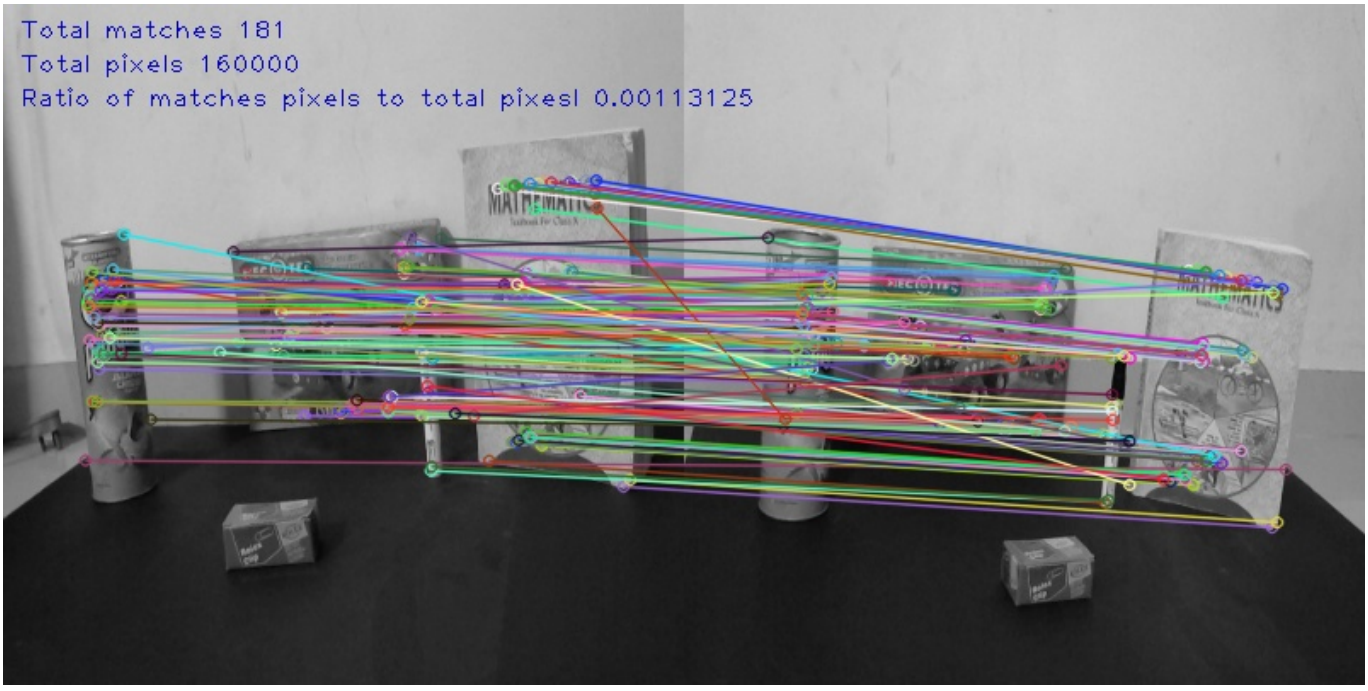
> Point correspondence is finding the corresponding pixel of one image in the other.

Point correspondence: Feature matching

Feature matching algorithms such as SIFT, SURF, or ORB are valid methods for matching features between the left and right stereo images.

However, these algorithms will most likely result in a very sparsely reconstructed 3D scene due to the poor ratio of total pixels to found pixels.

> Those approaches are also very computationally heavy, and would check each pixel, altough we only need to track pixels on our epipolar line.



Epipolar geometry to the rescue!

Epipolar geometry is crucial for stereo matching (the process of stereo vision) as it reduces the search space for point correspondence by eliminating false matches.

> Stereo matching is only feasible to do, if there's structure in the image. Therefore, stereo matching on a white wall performs very poorly.

Terminology

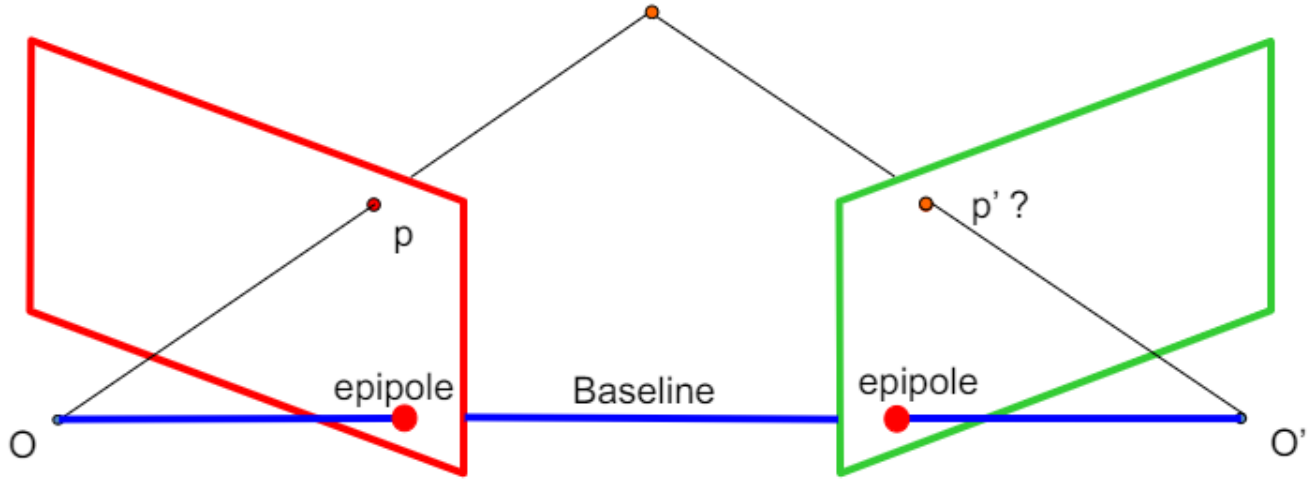
- *Baseline*: The line connecting the two camera centers.
- *Epipole*: Point of intersection of baseline with the image plane.
- *Epipolar plane*: The plane that contains the two camera centers and a 3D point in the world.
- *Epipolar line*: Intersection of the epipolar plane with each image plane.

How does it work?

We have two rigidly fixed cameras with different viewpoints capturing pictures. Now we want to know which pixels in an image pair (containing the left and right stereo images) corresponds to each other. If both cameras produce rays that pass through the red dots (projections of the 3D scene point onto the image plane), then they should intersect at that exact point, since each line passes through the scene point.

The ray from one image is reflected onto the image plane of the other image - displayed by the epipolar line that connects the epipole with the projection of the scene point in that image plane (red dot).

This way, the possible location of the right_image for example is constrained to a single line. Which reduces the search space for a pixel corresponding to a pixel of left_image.

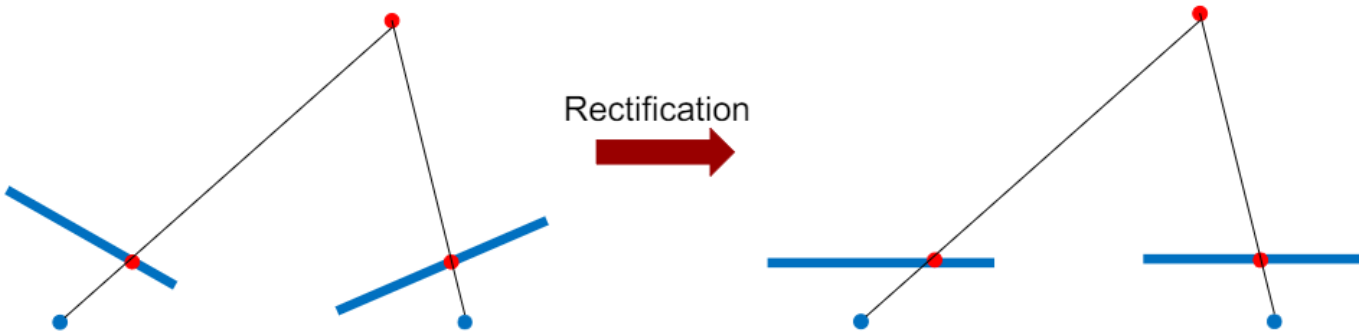


> We always spoke about finding the correspondence for one single pixel. To create a 3D structure of an image it's necessary to apply this method on each pixel of those image pairs.

Rectified Stereo Case

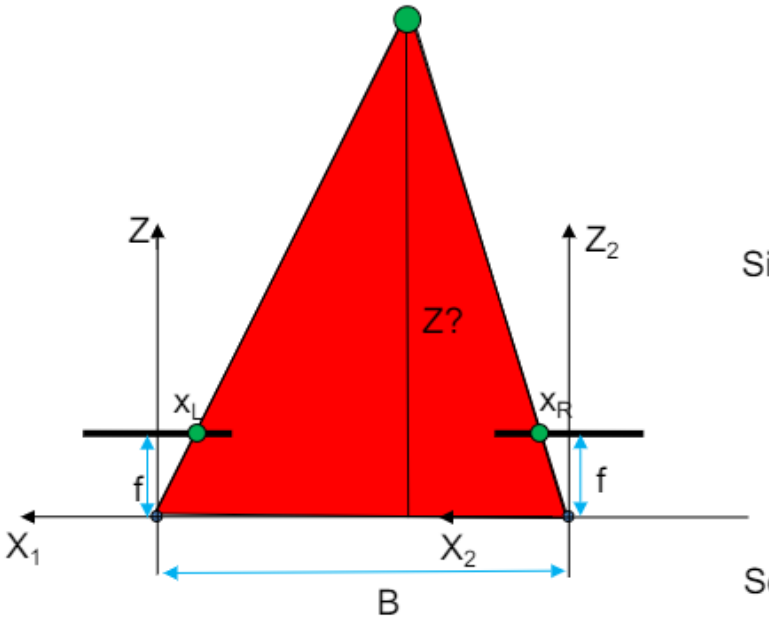
Rectification:

- The source images are projected onto a common plane parallel to the baseline B connecting the optical centers of the images
- Epipolar lines become parallel (and under certain conditions they become also horizontal)



Depth from Stereo Matches

It is important to note that Z extends from the scene point to the baseline. Therefore, we need to exploit similar triangles to formalize the following equation:



Similar triangles:

$$\frac{B+X_L-X_R}{Z-f} = \frac{B}{Z}$$

Solving for Z:

$$Z = f\frac{B}{X_L - X_R}$$

Disparity

1. The depth of the stereo image "Z" is proportional to the baseline of the point and inversely proportional to the difference of the corresponding points of the two images. This means that a small depth will result in a large difference or jump between the two images and vice versa.

2. The difference of the corresponding points of the two images is called the disparity.

3. A depth map is not the same as a disparity map. It's possible to convert from one to another as long as the focal length is known, but the difference between the two is that the disparity map is inversed and scaled, as only the pixel values are known. However, the information contained information is similar.

Correspondence Problem

Definition: The correspondence problem refers to the problem of determining which parts (clusters of pixels or individual pixels) of one image correspond to parts of another image. Most often, the differences arise from the movement of the camera, the flow of time and/or the movement of objects in the photos.

Beyond the hard constraint of epipolar geometry, there are "soft" constrains to help identify corresponding points:

- Similarity: The image pairs should be relatively similar in a sense, that they contain the same objects, colors, etc.
- Uniqueness: Objects in an image are assumed to contain unique features that are easily identifiable. These unique features should not differ in the two images, e.g. a person with a nose in the source image will most likely also have only one nose in the target image.
- Ordering: Let us assume that in a source image there is an apple, a coke and a teddy bear. We assume that this order remains the same in the target image, which is most likely the case. But there are exceptions to this rule.

- Disparity gradient is limited: The depth values should be continous for neighboring pixels. There will be jumps in depth and disparity, but they should bit limited.

> What is meant by hard constraint of epipolar geometry? This refers to the strict rule, that correpondences HAVE to be on the epipolar line. Soft constraints aren't always true, but most of the time they are.

To find matches in the image pair, we will assume:

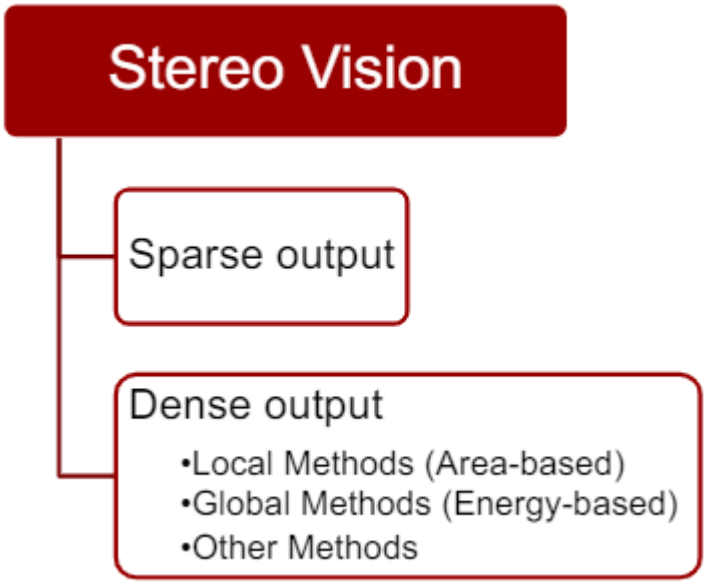
- most scene points are visibile in both images
- image regions for the matches are similar in appearance

Do we need dense or sparse stereo matching?

Before this question can be answered, we need to make sure that we understand the differences:

What's the difference between sparse or dense stereo matching? The difference lies in the number of calculations of your correspondences. With dense stereo matching, you need to calculate as many correspondences as possible, whereas with sparse stereo matching, only a small number of correspondences need to be calculated to meet the requirements of the application.

To come back to the question of which ones we need - it depends on the use case. For example, dense stereo matching is important for autonomous vehicles because you can't neglect any information to make sure it correctly assesses the situation and reacts accordingly. A face tracking system, on the other hand, needs only sparse stereo matching because it doesn't care about anything other than the face.

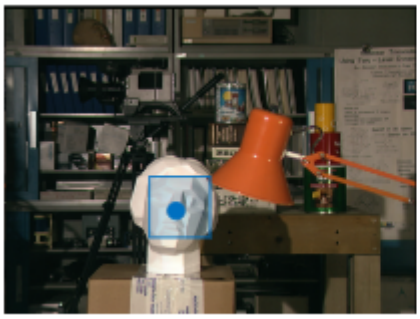
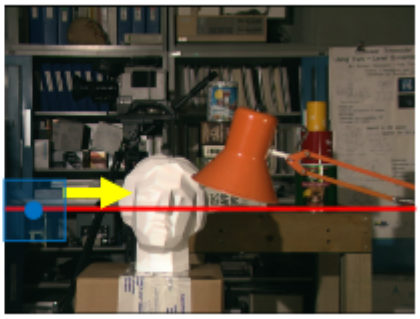


Dense Stereo Correspondence: Local Methods

Try to find correspondences for all the pixels of the reference image.

- For each epipolar line
    - For each pixel in the left image
        - Compare with every pixel on same epipolar line in right image
        - Choose the pixel that maximizes a similarity metric (or minimizes a dissimilarity metric!).

- Improvment: don't match individual pixels, but rather match windows!

> The proposed windows method slides from left to right along the epipolar line. To minimize dissimilarity while maximizing similarity.

Stereo Correpondence Metrics Different Metrics

Stereo Correspondence Metrics

- Sum of Absolute Differences (SAD)

$$SAD(x,y,d) = \sum_{x,y \in W} |I_l(x,y) - I_r(x,y-d)|$$

- Sum of Squared Differences (SSD)

$$SSD(x,y,d) = \sum_{x,y \in W} (I_l(x,y) - I_r(x,y-d))^2$$

- Normalized Cross-Correlation

$$NCC(x,y,d) = \frac{\sum_{x,y \in W} I_l(x,y) \cdot I_r(x,y-d)}{\sqrt{\sum_{x,y \in W} I_l^2(x,y) \cdot \sum_{x,y \in W} I_r^2(x,y-d)}}$$

- …many many more!!!

SAD Example:

$$\mathrm{A} = \begin{bmatrix} 2 & -10 & -2 \\ 14 & 12 & 10 \\ 4 & -2 & 2 \end{bmatrix}; \mathrm{B} = \begin{bmatrix} 6 & 10 & -2 \\ 0 & -12 & -4 \\ -5 & 2 & -2 \end{bmatrix};$$

$$\mathrm{C} = \mathrm{abs(A\text{-}B)} = \begin{bmatrix} |2-6| & |(-10)-10| & |(-2)-(-2)| \\ |14-0| & |12-(-12)| & |10-(-4)| \\ |4-(-5)| & |(-2)-2| & |2-(-2)| \end{bmatrix} = \begin{bmatrix} 4 & 20 & 0 \\ 14 & 24 & 14 \\ 9 & 4 & 4 \end{bmatrix};$$

$$\mathrm{SAD} = \mathrm{sum(C)}\ 4+20+0+14+24+14+9+4+4 = 93$$

> The closer the similarity metric calculated by the SAD is to 0, the stronger the similarity between these two images.

$$\mathrm{A} = \begin{bmatrix} 2 & -10 & -2 \\ 14 & 12 & 10 \\ 4 & -2 & 2 \end{bmatrix}; \mathrm{B} = \begin{bmatrix} 6 & 10 & -2 \\ 0 & -12 & -4 \\ -5 & 2 & -2 \end{bmatrix};$$

$$\mathrm{C} = \text{A-B } \odot A - B = \begin{bmatrix} (2-6)^2 & ((-10)-10)^2 & ((-2)-(-2))^2 \\ (14-0)^2 & (12-(-12))^2 & (10-(-4))^2 \\ (4-(-5))^2 & ((-2)-2)^2 & (2-(-2))^2 \end{bmatrix}$$

$$= \begin{bmatrix} 16 & 400 & 0 \\ 196 & 576 & 196 \\ 81 & 16 & 16 \end{bmatrix};$$

$$SSD = sum(C) = 16 + 400 + 0 + 196 + 576 + 196 + 81 + 16 + 16 = 1497$$

SSD Example:

> The closer the similarity metric calculated by the SSD is to 0, the stronger the similarity between these two images. Note that SSD is generally only used due to its simplicity and relatively low computational cost - in general better results are achievable by using Normalized Cross Correlation.

Normalized Cross-Correlation Example:

$$\mathrm{A} = \begin{bmatrix} 2 & -10 & -2 \\ 14 & 12 & 10 \\ 4 & -2 & 2 \end{bmatrix}; \mathrm{B} = \begin{bmatrix} 6 & 10 & -2 \\ 0 & -12 & -4 \\ -5 & 2 & -2 \end{bmatrix};$$

$$A^2 = \begin{bmatrix} 4 & 100 & 4 \\ 196 & 144 & 100 \\ 16 & 4 & 4 \end{bmatrix}; B^2 = \begin{bmatrix} 36 & 100 & 4 \\ 0 & 144 & 16 \\ 25 & 4 & 4 \end{bmatrix};$$

$$\mathrm{nom} = \mathrm{A} \odot B = \begin{bmatrix} 2*6 & ((-10)*10) & ((-2)*(-2)) \\ 14*0 & (12*(-12)) & (10*(-4)) \\ (4*(-5)) & ((-2)*2) & (2*(-2)) \end{bmatrix}$$

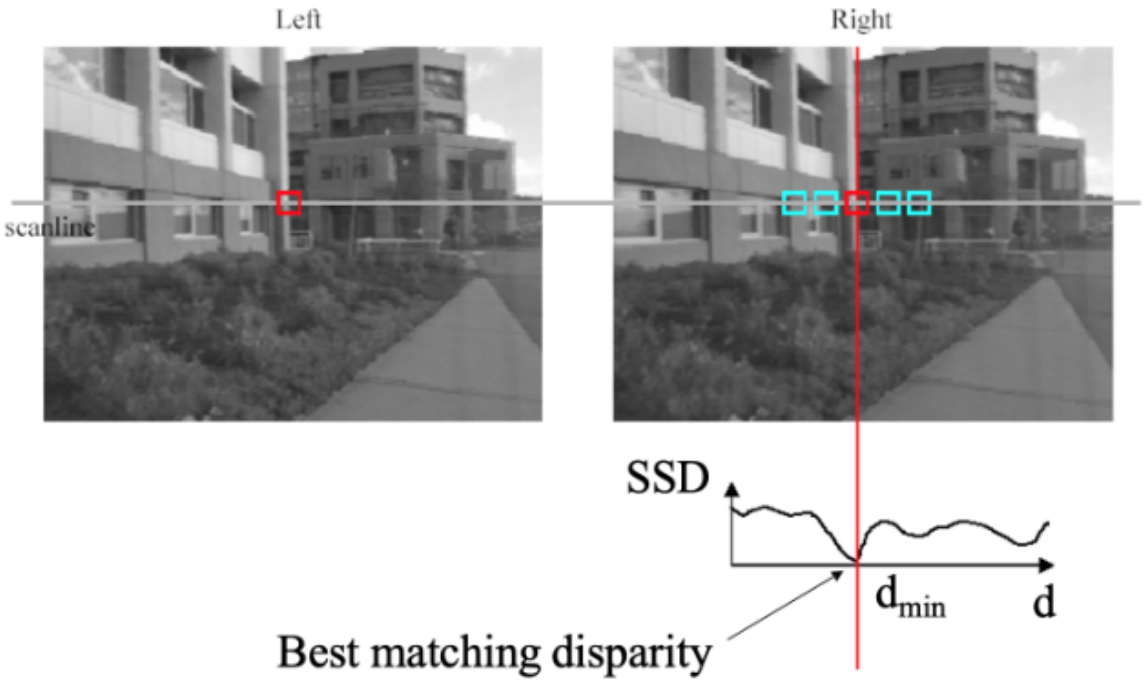
$$= \begin{bmatrix} 12 & -100 & 4 \\ 0 & -144 & -40 \\ -20 & -4 & -4 \end{bmatrix};$$

$$dom1 = A \odot A = \begin{bmatrix} 4 & 100 & 4 \\ 196 & 144 & 100 \\ 1644 & & \end{bmatrix}$$

$$dom2 = B \odot B = \begin{bmatrix} 26 & 100 & 4 \\ 0 & 144 & 16 \\ 25 & 4 & 4 \end{bmatrix}$$

$$NormalizedCross - Correlation = \frac{nom}{\sqrt{dom1} * \sqrt{dom2}} = \frac{-296}{\sqrt{572} * \sqrt{333}} = -0.678$$

Stereo Correspondence Metrics: SSD



> Visual illustration that the lowest SSD value found for a target pixel on the epipolar line indicates that this pixel is most similar to the source pixel and that there is therefore a correspondence between these pixels.

Windows Sizes

Large windows suppress noise better than smaller windows. However, if the window is too large, the image loses its fine granularity. With smaller windows, the structure is better preserved, but at the expense of a gain in noise.

Global Stereo Correspondence

Global Stereo Correspondence

- Up to this point, the disparity of each pixel was determined only by the information of the pixel itself and its neighborhood.
  - Thus, those methods are called ”local” or “area-based” methods.

- Global methods find better solutions in expense of more computations
  - Optimize jointly the disparity values of all the pixels of each scanline (e.g. Dynamic Programming)
  - Optimize jointly the disparity values of all the pixels of the image (e.g. graph cuts)

    - *In global algorithms, stereo correspondence is formulated as an energy function minimization problem, consisting of data and smoothness terms.*

Perception for Autonomous Systems

Lecture 4 - Calibration and Stereo Vision 2 (22/02/2021)

Outline/Content:

- Camera matrix (Book A 2.1.5)
- Distortion coefficients (Book A 2.1.6)
- Calibration (Paper A)
- Undistortion
- Algebraic Derivation of Stereo Vision (Book B 7.1)
    - (Essential and) Fundamental Matrix
- Stereo Calibration and Rectification
    - Ranging

> Additional Reading Material: Paper A; Z. Zhang, "A flexible new technique for camera calibration," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330-1334, Nov. 2000. doi: 10.1109/34.888718

Epipolar Geometry: General Case

In the general case, the following assumptions can be made for the determination of the components in the epipolar geometry:

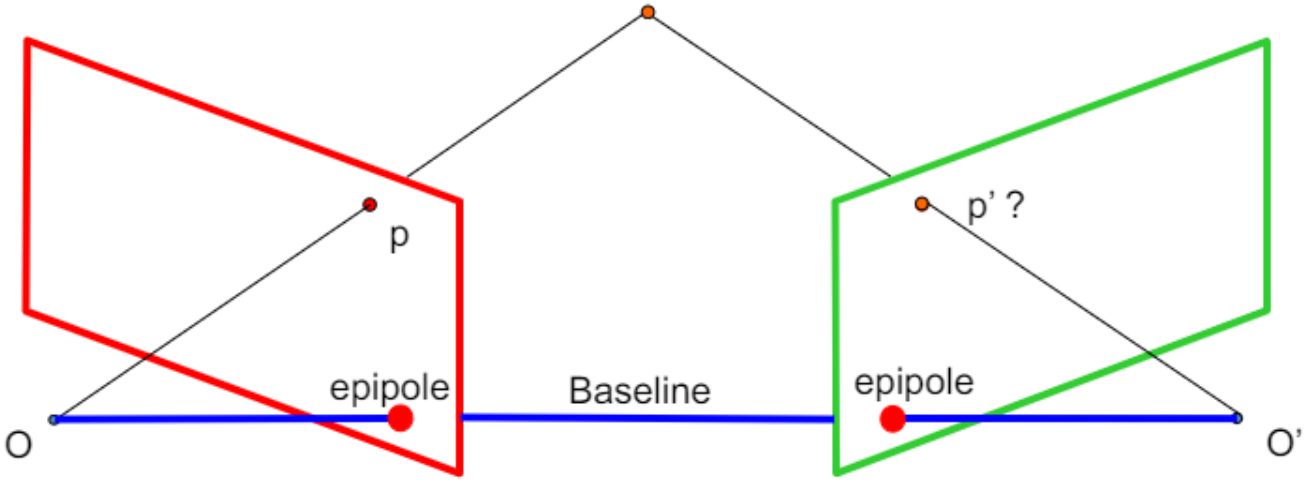
- 2 **Angled** camera views
- Rays passing through the camera center

> NOTE: If the camera views are parallel, we don't

With this knowledge, we can easily obtain the following two components (as shown in the next figure):

- Baseline
- Epipolar lines
- Epipolar plane
- Epipoles

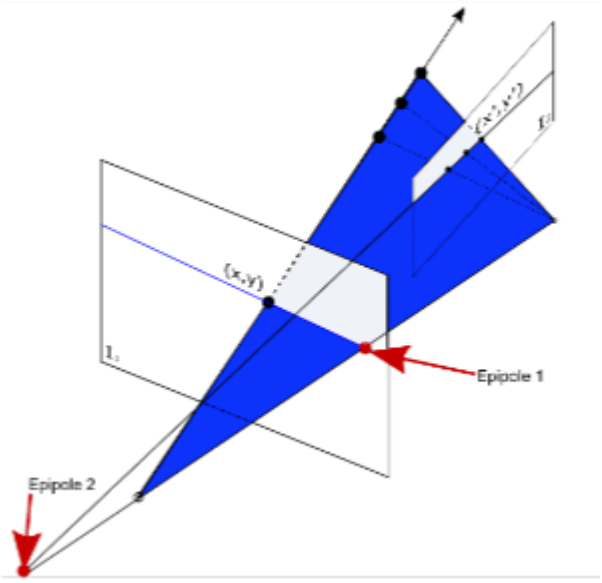
> Necessary knowledge from the previous lecture:
>
> - *Baseline*: The line connecting the two camera centers.
> - *Epipole*: Point of intersection of baseline with the image plane.
> - *Epipolar plane*: The plane that contains the two camera centers and a 3D point in the world.
> - *Epipolar line*: Intersection of the epipolar plane with each image plane.



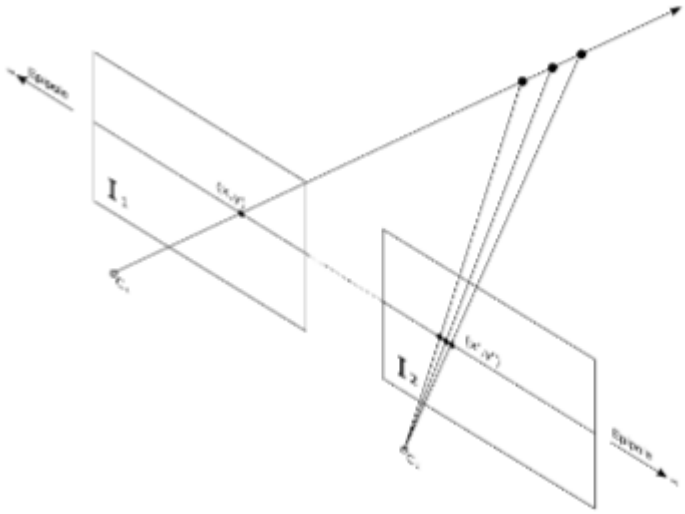
Epipolar Geometry: Special Case

For special cases it's not always possible to determine the epipolar geometry comonents.

Example 1:



In the first case, we have two staggered images. We're able to determine the **Baseline**, **Epipolar plane**, **Epipolar lines** and the **Epipoles**. However, Due to the placement of the cameras it's noticeable that one of the epipolar poles is not part of the epipolar plane.



Example 2:

In the second case we have vertically aligned images. We can determine the **baseline**, **epipolar plane**, **epipolar lines** but NOT the **epipoles**. This is because the alignment of the two images makes them parallel to the baseline. Therefore, an intersection between the baseline and the epipolar line is not possible.

> This case can be useful by using the scanline to calculate the disparity

Essential Matrix Lecture Slides - Carnegie University

The Essential Matrix E is a 3x3 matrix, and it relates corresponding image points between both cameras, given the rotation and translation.

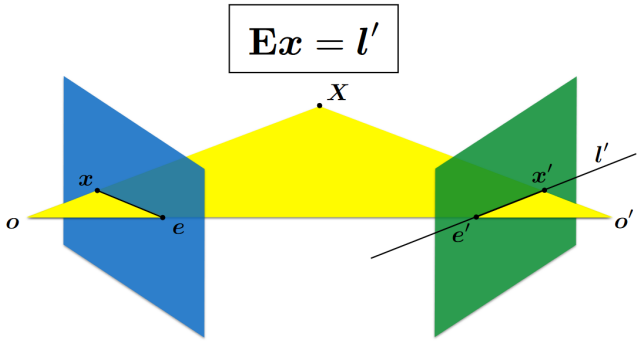
> 1. Given a point in one image, multiplying by the essential matrix will tell us the epipolar line in the second view.
> 2. It's assumed that both cameras are calibrated. For uncalibrated cameras, see section Fundamental Matrix.
> 3. See the link for the lecture slides of the Carnegie Mellon university, as it is perfectly explained, but unfeasible to include into this section.

---

properties of the E matrix

Longuet-Higgins equation	$\boldsymbol{x}'^{\top}\mathbf{E}\boldsymbol{x}=0$	
Epipolar lines	$\boldsymbol{x}^{\top}\boldsymbol{l}=0$ $\boldsymbol{l}'=\mathbf{E}\boldsymbol{x}$	$\boldsymbol{x}'^{\top}\boldsymbol{l}'=0$ $\boldsymbol{l}=\mathbf{E}^{T}\boldsymbol{x}'$
Epipoles	$\boldsymbol{e}'^{\top}\mathbf{E}=\mathbf{0}$	$\mathbf{E}\boldsymbol{e}=\mathbf{0}$

(points in normalized camera coordinates)

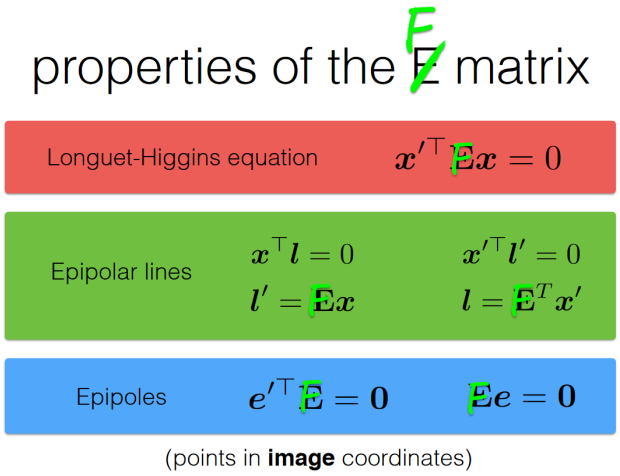


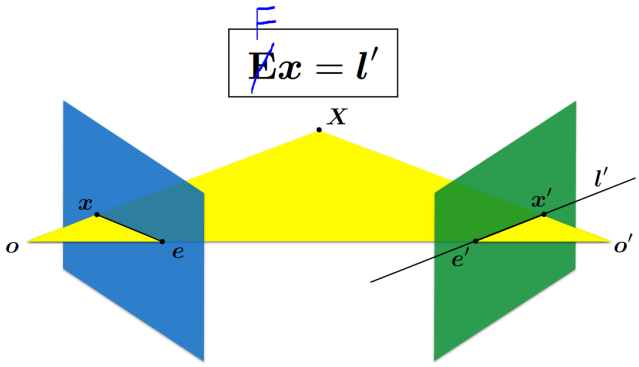
Characteristics:

– $E\,x'$ is the epipolar line associated with x' ($l = E\,x'$)
– $E^T x$ is the epipolar line associated with x ($l' = E^T x$)
– $E\,e' = 0$ and $E^T e = 0$
– E is singular (rank two)
– E has five degrees of freedom

Fundamental Matrix Lecture Slides - Carnegie University

The 3x3 Fundamental matrix is a generalization of the Essential matrix, where the assumption of calibrated cameras is removed.





Characteristics:

– $F\,x'$ is the epipolar line associated with x'
– $F^T x$ is the epipolar line associated with x
– $F\,e' = 0$ and $F^T e = 0$
– F is singular (rank two): det(F)=0
– F has seven degrees of freedom:

> See the link for the lecture slides of the Carnegie Mellon university, as it's well explained, but unfeasible to include into this section.

How to compute the Fundamental Matrix Lecture Slides - Carnegie University

The method for computing the Fundamental Matrix is called the 8 point algorithm, as 8 points are needed to solve it.

$$\mathbf{x}^T F \mathbf{x}' = 0$$

$$xx'f_{11} + xy'f_{12} + xf_{13} + yx'f_{21} + yy'f_{22} + yf_{23} + x'f_{31} + y'f_{32} + f_{33} = 0$$

$$\mathbf{A}\boldsymbol{f} = \begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_nx_v' & x_ny_n' & x_n & y_nx_n' & y_ny_n' & y_n & x_n' & y_n' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ \vdots \\ f_{33} \end{bmatrix} = \mathbf{0}$$

- Homography (No Translation)
- Correspondence Relation

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \Rightarrow \mathbf{x}' \times \mathbf{H}\mathbf{x} = \mathbf{0}$$

1. Normalize image coordinates

$$\tilde{\mathbf{x}} = \mathbf{T}\mathbf{x} \quad \tilde{\mathbf{x}}' = \mathbf{T}'\mathbf{x}'$$

2. RANSAC with 4 points
   - Solution via SVD
3. De-normalize:

$$\mathbf{H} = \mathbf{T}'^{-1}\tilde{\mathbf{H}}\mathbf{T}$$

- Fundamental Matrix (Translation)
- Correspondence Relation

$$\mathbf{x}'^T \mathbf{F}\mathbf{x} = 0$$

1. Normalize image coordinates

$$\tilde{\mathbf{x}} = \mathbf{T}\mathbf{x} \quad \tilde{\mathbf{x}}' = \mathbf{T}'\mathbf{x}'$$

2. RANSAC with 8 points
   - Initial solution via SVD
   - Enforce $\det(\tilde{\mathbf{F}}) = 0$ by SVD
3. De-normalize:

$$\mathbf{F} = \mathbf{T}'^T \tilde{\mathbf{F}} \mathbf{T}$$

> 3. See the link for the lecture slides of the Carnegie Mellon university, as it is perfectly explained, but unfeasible to include into this section.

Epipolar Geometry to Rectified Epipolar Geometry

Necessary steps:

1. Match some points
2. Calculate the fundamental matrix
3. Calculate the rectified images

Perception for Autonomous Systems

Lecture 5 - 3D Point Cloud Processing - Pose Estimation (01/03/2021)

Outline/Content:

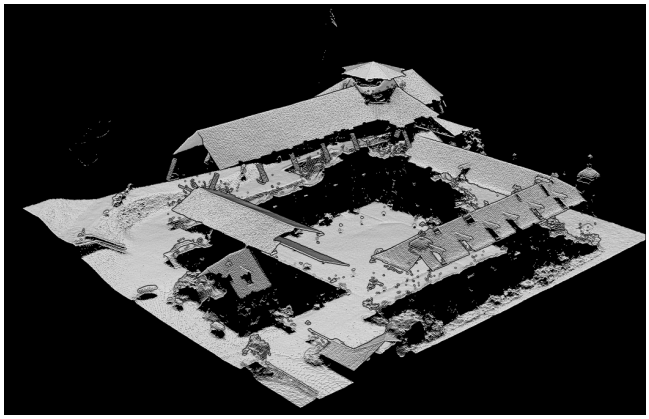
- Why do we need 3D Point Clouds?
- Why is Pose Estimation in 3D important?
- Point Cloud Registration
    - Local Alignment
        - Iterative Closest Point (ICP) algorithm
    - Global Alignment
        - 3D Feature Descriptors
        - Spin Images
        - PFH
        - FPFH
    - Random Sample Consensus (RANSAC) in 3D

> Reading Material:
>
> - Book A, Sections 6.1 and 6.2
> - [Paper 1 - ICP] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pp. 239-256, Feb. 1992.
> - [Paper 2 - Spin Images] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 5, pp. 433-449, May 1999.
> - [Paper 3 - PFH] R. B. Rusu, N. Blodow and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," 2009 IEEE International Conference on Robotics and Automation, Kobe, 2009, pp. 3212-3217.
> - [Paper 4 - FPFH] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, “Aligning Point Cloud Views using Persistent Feature Histograms,” in Proceedings of the 21st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nice, France, September 22-26, 2008.

What is a 3D Point Cloud? Point Cloud Expert: Florent Poux Medium

A point cloud is a set of data points in a three-dimensional coordinate system. These point are spartially defined by [x, y, z]coordinates and often present a surface of a physical asset or object. Reality capture devises obtain the external surface in its three dimensions to generate the point cloud. These are commonly obtained through Photogrammetry, LiDAR (Terrestrial Laser Scanning), Mobile Mapping, Aerial LiDAR, depth sensing, sonar, and more recently deep learning through Generative Adversarial Networks.

Photogrammetry	**Aerial LiDAR**

Photogrammetry	Aerial LiDAR
	

Why do we need 3D Point Clouds? Article

The world is in 3D

- Objects are not entirely described by 2D images
- 3D geometry and shape are often important

However,

- operations in 3D are computationally heavy
- SIFT/SURF/... do not work in point clouds

What is "Pose"

Pose

- the tranformation (translation + rotation) needed to map one point cloud (model) to another point cloud (model) of the same (fully or partially) object or scene.

or equivalently

- ... determining a camera's position relative to known 3D object or scene... "(Szelinisk)

Why is 3D Pose Estimation Important? Article

Many applications in Autonomous Systems

- Robot Grasping/Manipulation
- Quality Inspection
- Augmented reality
- Progressive map building (real-time)
- Localization (real-time)

Other applications:

- rescue and recovery
- explore dense forests

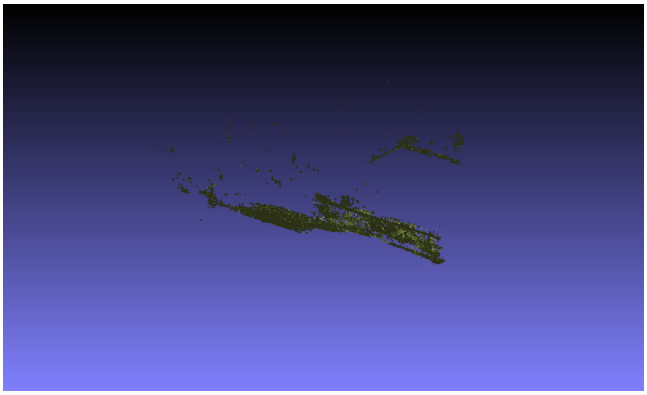
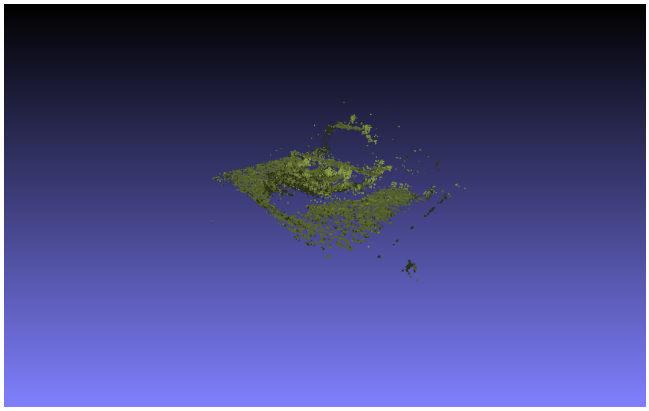
> - Rescue and recovery: Robots or drones with LiDAR sensors can reach dangerous locations and scan or generate useful data helping humans to navigate safely at such risky places and avoid mishaps. Natural disasters, unknown hidden dense forests and dark caves are the best examples of such dangerous locations.
> - Explore dense forests: A sprawling maya network in the guatemala jungle has been discovered with the help of Aerial LiDAR, which mapped the dense forest and discovered subtle sign of the archaelogical wonder.

Point Cloud registration Paper

Definition: Point Cloud Registration is a fundamental problem in 3D computer vision and photogrammetry. Given several sets of points in different coordinate systems, the aim of registration is to find the transformation that best aligns all of them into a common coordinate system. Point Cloud Registration plays a significant role in many vision applications such as 3D model reconstruction, cultural heritage management, landslide monitoring and solar energy analysis.

Generally 2 steps are needed to register 2 given point clouds of the same object (fully or partially)

- Global alignment (3D Feature Descriptors)
  - The 2 models are "roughly aligned"
- Local alignment (ICP)
  - Starting from a "rough" initial alignment, find the exact precise alignment

Before Registration	**After Registration**
	

Local Alignment Jupyter-Notebook

Having two scans (point clouds) of the same object P={pi} and Q={qi} we want to find a transformation (rotation R and translation t) to apply to P to match Q as good as possible (alignment).

This leads to the following two questions:

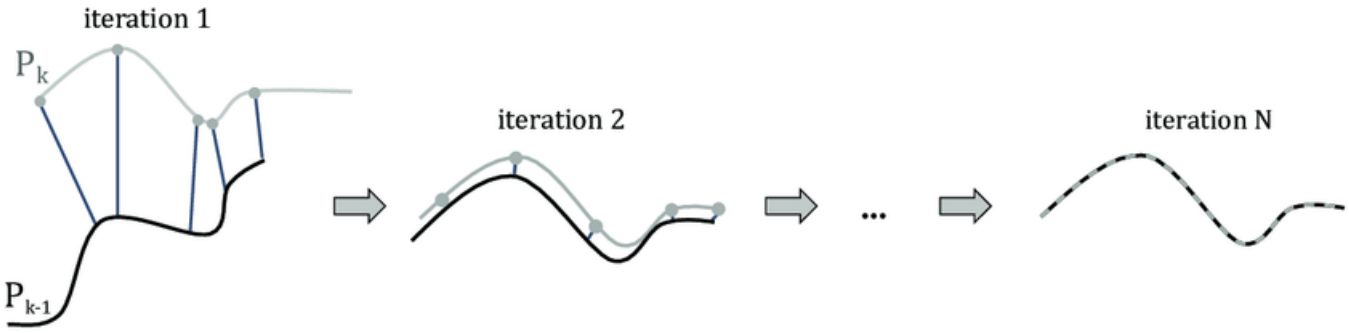
- What is the difference of theirs pose? ... or equivalently..
- What is the transformation that can fully align them?

> Checkout the linked jupyter notebook to see the IPC algorithm hands on

Iterative Closest Point (ICP)

1. For each object point p in model 1, find the nearest point q in model 2
2. Use all pairs (p,q) to estimate the transformation from model 1 to model 2
3. Apply the transformation to the points of model 1
4. Repeat steps 1-3 until convergence/stop criterion is met

Visual representation of the IPC algorithm



Elaborating the bullet points 1 and 2 in greater detail

For each object point p in model 1, find the nearest point q in model 2: How to compute all possible pairs for proximity?

Simple k-nearest neighbor approach, is being used to find correspondences (similair points) in the two point clouds. However, this approach can be computationally heavy and slow. A better and more suitable approach is K-d trees.

Use all pairs (p,q) to estimate the transformation from model 1 to model 2: How to estimate the transformation from model 1 to model 2, fiven pairs (p,q)

Kabsch Algorithm / Procrustes Analysis:

- Translate the centroids of both models to the origin of the coordinate system (0,0,0)
    - Compute centroids cp, cq of both models
    - Subtract from each point coordinates the coordinates of its corresponding centroid:
        - p' = p - cp and q' = q - cq
- Compute Covariance Matrix: Cpq = Sum(p' * q^'T)
- Compute the optimal rotation
    - Calculate the singular value decomposition (SVD) of the covariance matrix: Cpq = USV'
    - Calculate rotation matrix: R = UV^T
- Compute the optimal translation: T = cq - Rcp

Why does the Kabsch Algorithm needs to be applied at every iteration?

The reason for the Kabsch Algorithm being applied at every iteration is that our proximity approach is not able to initially find all correspondences for each (p,q) pair. Thus, the algorithm needs to revalue it's estimation based on the new values it's provided with, until it converges or a stop criterion is met.

Global Alignment

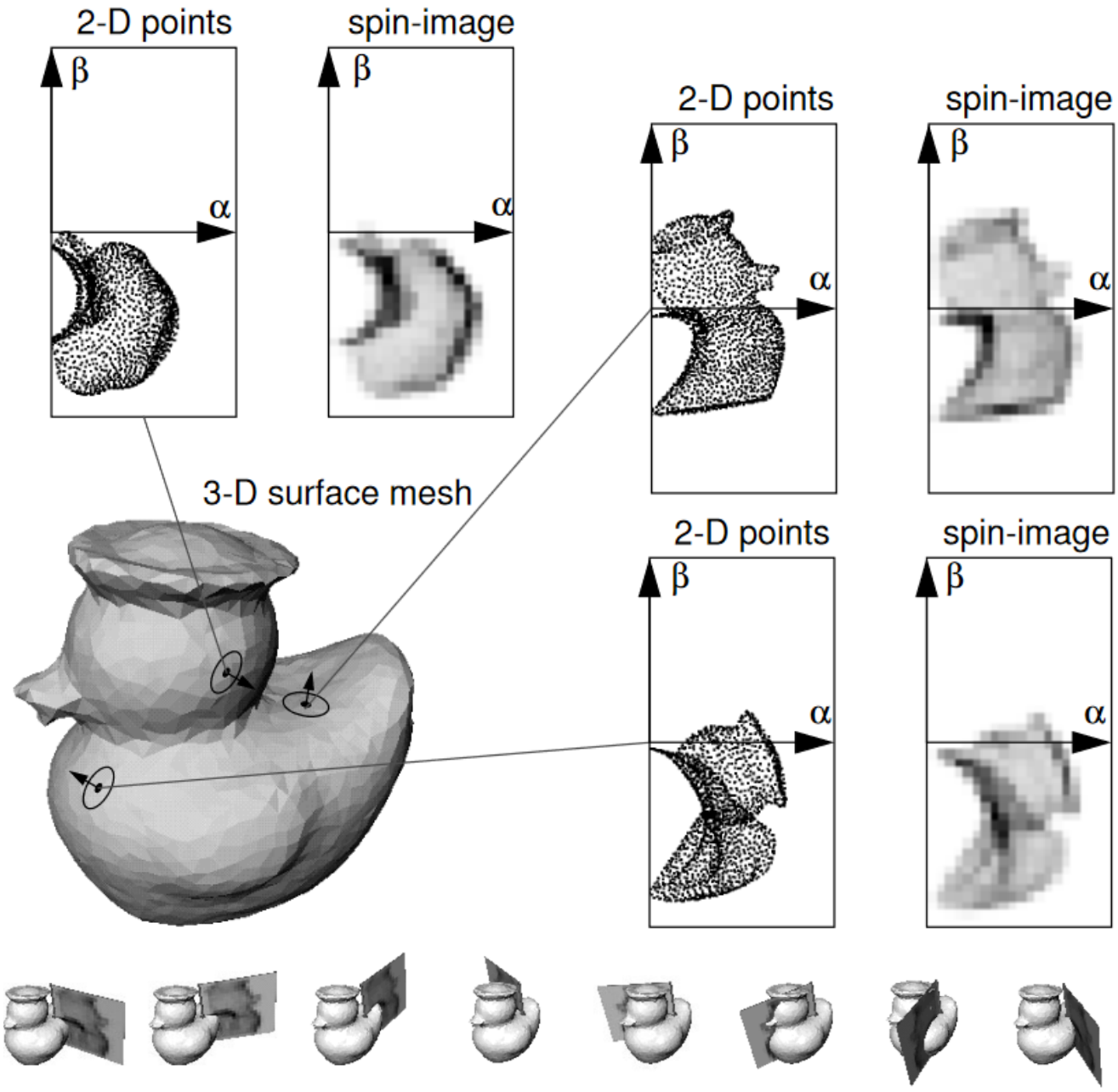
Consider 2 models (point clouds) of the same object (fully or partially)

- How can we "roughly" align them?
    - We cannot use ICP, if the initial poses are too different
    - Can we use some kind of (3D) "features" and match them?

Spin Images: Paper

- "Spin" a discretized 2D grid around the surface normal of a point.
- Accumulate neighboring pixels in the grid bins, while spinning.
- The descriptor is the 2D grid (image) where each element (pixel) contains the number of accumulated points.

Visualization:



Spin-images from points on one surface are compared by computing correlation coefficient with spin-images from points on another surface; when two spin-images are highly correlated, a point correspondence between the surfaces is established. More specifically, before matching, all of the spin-images from one surface (the model) are constructed and stored in a spin-image stack. Next, a vertex is selected at random from the other

surface (the scene) and its spin-image is computed. Point correspondences are then established between the selected point and the points with best matching spin-images on the other surface. This procedure is repeated for many points resulting in a sizeable set of point correspondences(~100). Point correspondences are then grouped and outliers are eliminated using geometric consistency. Groups of geometrically consistent correspondences are then used to calculate rigid transformations that aligns one surface with the other. After alignment, surface matches are verified using a modified iterative closest point algorithm. The best match is selected as the one with the greatest overlap between surfaces.

PHP (Point Feature Histograms) Tutorial

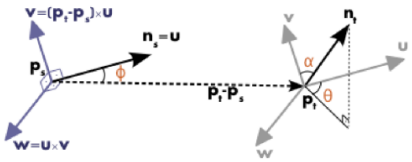
Global Alignment – 3D Feature Descriptors

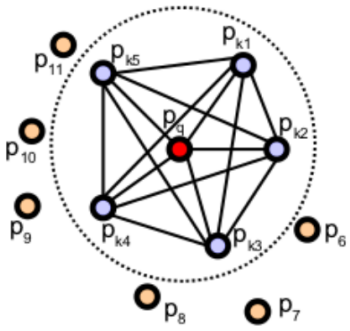
- **PFH (Point Feature Histograms)**

$$\alpha = \arccos\left(v \cdot n_t\right)$$

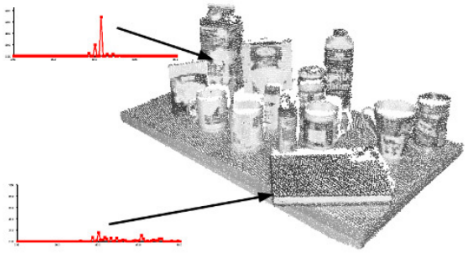
$$\phi = \arccos\left(u \cdot \frac{\left(p_t - p_s\right)}{\left\|p_t - p_s\right\|_2}\right)$$

$$\theta = \arctan\left(w \cdot n_t, u \cdot n_t\right)$$





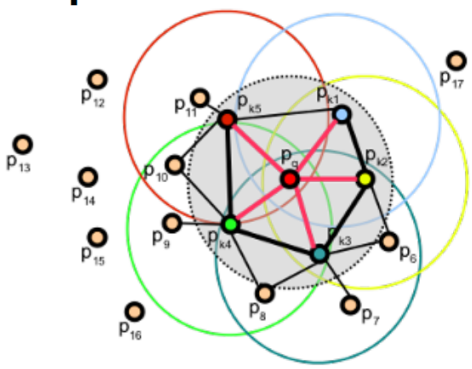
- Calculate these 3 values for all pairs of points within a radius *r* from the considered point (maybe also their distance *d*)
- The set of all triplets/quadruplets are binned in a histogram – This is the multi-dimensional descriptor

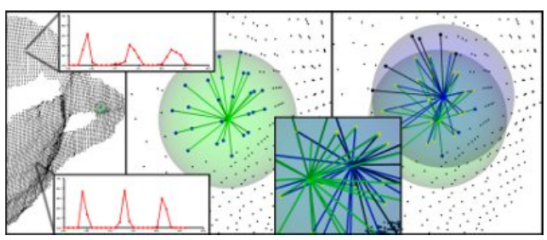


FPFH (Fast Point Feature Histograms) Tutorial

Global Alignment – 3D Feature Descriptors

- **FPFH (Fast Point Feature Histograms)**
  - Simplification/Approximation of the PFH formulation
    - reduces the computational complexity
    - retains most of the discriminative power of PFH.
  - Algorithm:
    - Find all oriented points in a spherical neighborhood of radius *r* around each point (k-d tree)
    - Compute relative angles using surface normals and direction vector from the source point to each neighbor
  - The descriptor is a multi-dimensional histogram





RANSAC in 3D Youtube

Matching 3D features generally results in many false-matches

- Big additional computational cost, brings only small increase in the matching accuracy
- Need for an algorithm that is robust to outliers

> RANSAC to the rescue

We can use:

- Model 1 (point cloud with normals)
- Model 2 (point cloud with normals)
- Feature matches (containing many outliers)

In order to:

- Estimate the "rough" pose difference between the 2 models

RANSAC for Pose Estimation

- Randomly choose 3 pairs of matched points
- Estimate the relative pose
  - Kabsch/ Procrustes
- Apply the transformation and assess its "validity":
  - number of inliers: matched point pairs whose distance became "small" after applying the specific transformation
  - Sum of distances between all matched point pairs after transformation
  - ...
- Keep the transformation with most inliers
- Repeat random sampling

Pros:

1. RANSAC often produces better results, even in noiseless cases
2. RANSAC does not require a good initial estimate of the transform between the two data sets
3. RANSAC can be used with data sets that do not have as many local features

Perception for Autonomous Systems

Lecture 4 - 3D Point Cloud Processing (Clustering & Regression) (01/03/2021)

Outline/Content:

- A taxonamy of ML algorithms
- Why is ML important for Perception of Autonomous Systems?
- Regression
- Clustering
    - k-means
    - Mean Shift
    - DBSCAN
    - Hierarchical Clustering

> Reading Material:
>
> - Book A, Sections 6.1 and 6.2
> - [Paper 1 - ICP] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pp. 239-256, Feb. 1992.
> - [Paper 2 - Spin Images] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 5, pp. 433-449, May 1999.
> - [Paper 3 - PFH] R. B. Rusu, N. Blodow and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," 2009 IEEE International Conference on Robotics and Automation, Kobe, 2009, pp. 3212-3217.
> - [Paper 4 - FPFH] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, “Aligning Point Cloud Views using Persistent Feature Histograms,” in Proceedings of the 21st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nice, France, September 22-26, 2008.

A taxonamy of ML algorithms

Supervised Learning: The target values are known:

- Regression: The target value is numeric
- Classification: The target value is nominal

Unsupervised Learning: The target values are unknown

- Clustering: Group together similar instances

Reinforcement learning: Interacting with a dynamic environment the system must perform a certain goal.

Why is ML important for Perception of Autonomous Systems? Article

Machine learning can be employed as a replacement for traditional computer-vision algorithms, making it useful in autonomous vehicles for **object detection, classification, segmentation, tracking, and prediction.**

Doing this will impact the system's level of determinism, safety, and security.

**The Benefits of Using ML for Object Detection and Classification*

ML algorithms

- achieve greater degrees of accuracy than vision-based systems (when trained sufficiently)
- are more adaptable and scalable than vision systems
- can easily handle large volumes of data
- can evolve without human input
- reliance on determinist behavior

> For greater elaboration on those bullet points, see the linked article.

The Limitations of Machine Learning for Object Detection and Classification

ML algorithms

- rely heavily on the data quality on which the ML model is trained on. High data quality crucial for good performance in the real world. Otherwise, it could lead to undesirable and possibly dangerous outcomes.
- take long to train/validate and require huge amounts of computing processing resources.
- are limited when it comes to teaching a system how to respond to something that humans have innately. For instance, the "sixth sense" that a car may be about to pull in front of you, or a truck may suddenly slam on the brakes.

> For greater elaboration on those bullet points, see the linked article.

What are Clustering and Regression?

Regression

Regression tries to assign correct significance (weights) to input variables and formulate a weighted linear combination of them that can predict the output variables.

- The number of input and output variables can vary depending on the specific problem
- The final output of the algorithm is a regression line
- The goal is to find an equation f() that models the relationship between input x and output y such as

Regression Models:

- Linear Regression: Assume that function f is linear.
- Locally Weighted Regression: Creates multiple linear models on small neighbor data.

> Other types of Regression models exist, but are not relevant for the continuing part of the course Link:
>
> - Logistic Regression
> - Lasso Regression
> - Ridge Regression
> - Elastic Net Regression
> - Stepwise Regression

The performance for the linear regression method is evaluated by an error function. The most used one is the Mean Squared Error (MSE):

Equation:

$$MSE=\frac{1}{N}\sum_{n=1}^{N}(t-y)^2$$

where t is the true value of the learned output and y is the predicted value.

Linear Regression Medium

Linear Regression assumes that f(x) is a linear combination between the inputs x and a set of regression coefficients b:

Equation:

$$y=f(b,x)=b_1x_1+b_2x_2+b_3x_3+...+b_nx_n+c=b^Tx+c$$

The goal of linear regression is to find regression coefficients b that minimize the squared error between the true values of the output and the predicted values.

How do we minimize the squared error?

This can be done by using the Gradient Descent algorithm. The main goal of Gradient descent is to minimize the cost value of a function.

Gradient descent has an analogy in which we have to imagine ourselves at the top of a mountain valley and left stranded and blindfolded, our objective is to reach the bottom of the hill. Feeling the slope of the terrain around you is what everyone would do. Well, this action is analogous to calculating the gradient descent, and taking a step is analogous to one iteration of the update to the parameters.

Polynomial Regression Medium Article

Linear regression only works with linear data, while polynomial regression, which is a special case of linear regression, is used to deal with non-linear data points. In this case, a curvilinear relationship is established between input x and output y.

This is done by adding dimensions to the input data, where n is the degree of polynomial.

> Polynomial regression is seen as a special case of linear regression since it is still linear in the regression coefficients.

Equation:

$$Y=\theta_0+\theta_1x+\theta_2x^2+\theta_3x^3+...+\theta_nx^n$$

Where:

- Y is the target
- x is the predictor
- Omega_0 is the bias
- Omega_1 - Omega_n are the weights in the equation of the polynomial regression
- n is the degree of the polynomial

> The degree of polynomial can be found by cross-validation

Pros	Cons
Provides the best approximation of the relationship between the dependent and independent variables	The presence of one or two outliers in the data can seriously affect the result of the nonlinear analysis
A broad range of functions can be fit under it	Too sensitive to outliers
Polynomial basically fits a wide range of curvature	Presence of fewer model validation tools for the detection of outliers in nonlinear regression

Use cases for Regression on 3D point clouds:

- Line fitting
- Plane fitting
- Grid point fitting Paper
- Fitting other (non-linear) surfaces

> If you were wondering about whether 3D point clouds and 3D meshes are the same:
>
> - A point cloud is a collection of points to represent an object in the given co ordinate system. For example, in a 3 dimensional (X, Y, Z) co ordinate system, a point cloud represent a 3D object. Point clouds are used to create 3D meshes and other models used in 3D modeling
> - A 3D mesh is the structural build of a 3D model consisting of polygons. 3D meshes use reference points in X, Y and Z axes to define shapes with height, width and depth.

Clustering (or "Cluster Analysis")

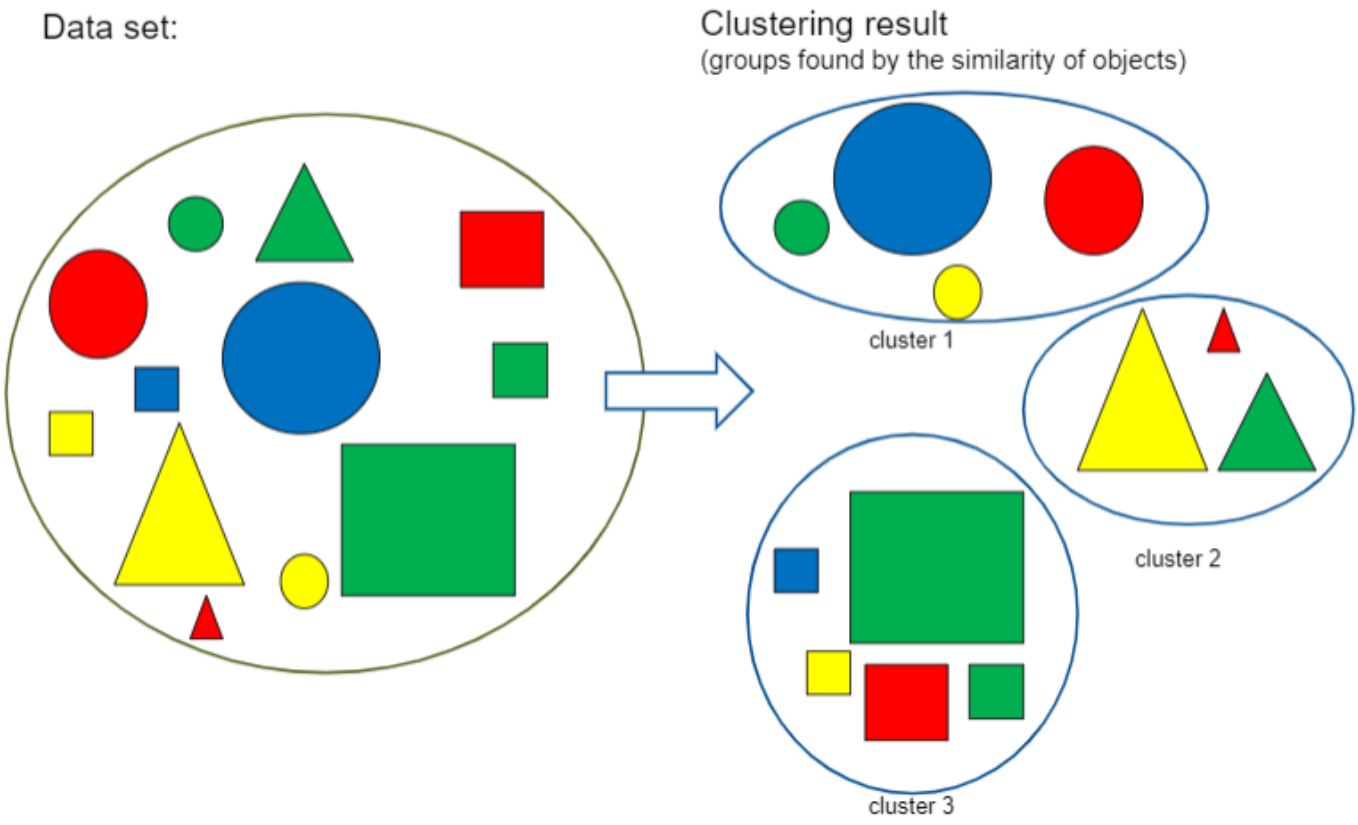
Find previously unkown groups (="clusters") in a data set, such that:

- data objects within one cluster are **similar to each other**
- data objects of different clusters are **dissimilar to each other**

Some applications of clustering:

- find different types of customers
  - e.g. an online shop could find groups of customers that are likely to spend a lot of money and customers that have not spend money for a long time
- detect communities in social networks
- group similar pixels in images (image processing)
- group similar documents (search engines, text mining)
- find similar recording from technical systems (automotive, automation)

- etc.



> The image could be clustered in different ways, for example shape, sizes or colors. Clustering is known as unsupervised learning because no information about classes is available for the instances.

The most common types of clustering methods are:

- partitioning methods
- hierarchical methods
- density-based methods
- grid-based methods

Partitioning method (k-means)

- find k clusters in the data set (k has to be pre-defined !)
- each cluster must contain >= 1 instances
- each instance must belong to exactly one cluster
- usually distance-based

> One distance based approach is to minimize the sum of squared euclidean distances (distortion)
>
> - $$J = \sum_{k} \sum_{n \in k} d(x_n, \mu_k)^2$$
> - where x is a vector representing the n^{th} data point assigned to cluster k and μ_k is the centroid of the cluster k. The function d() is the Euclidean distance between x and μ

Steps:

1. Creation of initial partitioning (e.g. randomly)
2. Iterative improvement of the partitioning by moving objects from one cluster to another. This is done by optimizing some criterion of what a "good" partitioning should look like.
3. Stop, if the partitioning quality criterion is satisfied.

Algorithm k-Means visualised

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

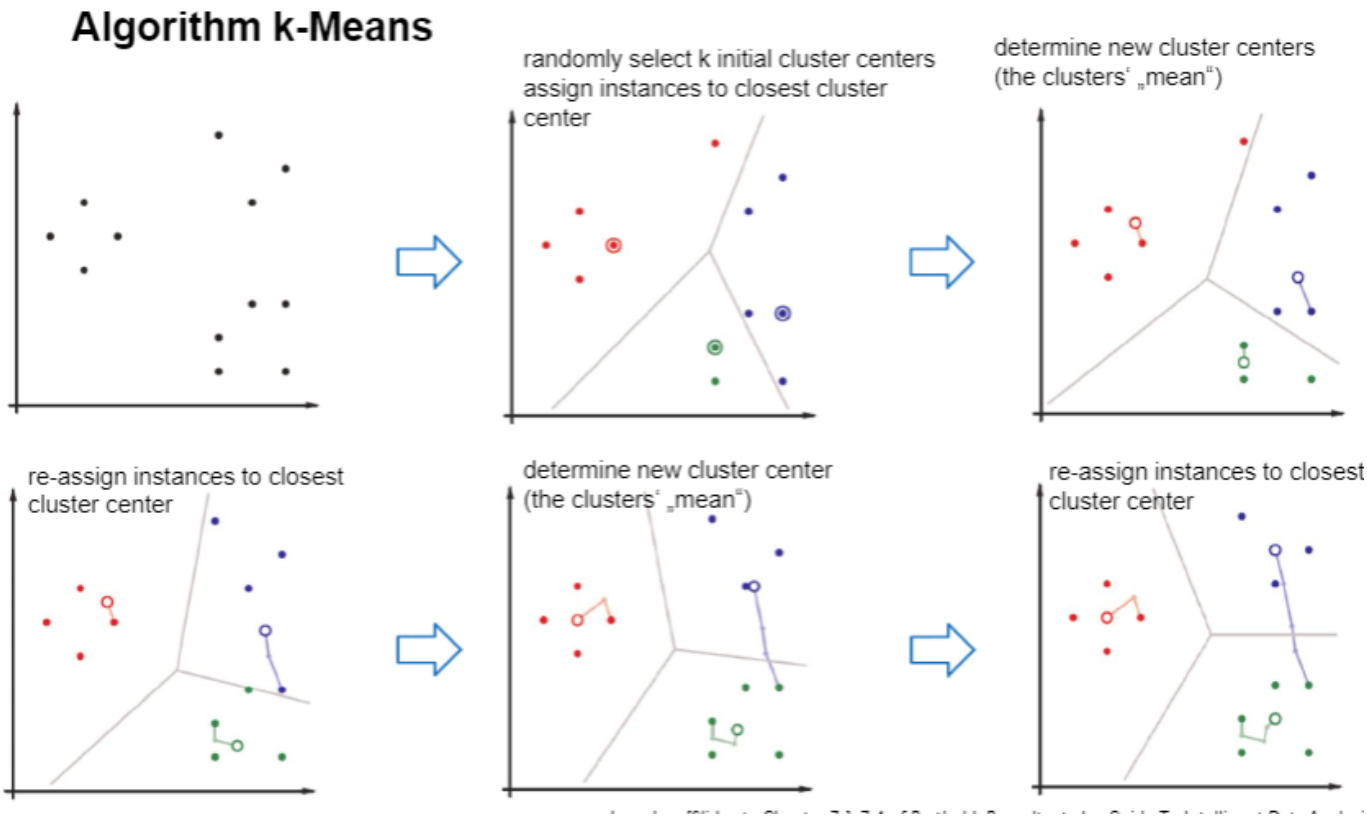
- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, that is, calculate the mean value of the objects for each cluster;
(5) **until** no change;

Algorithm k-Means visualised



How to define k?

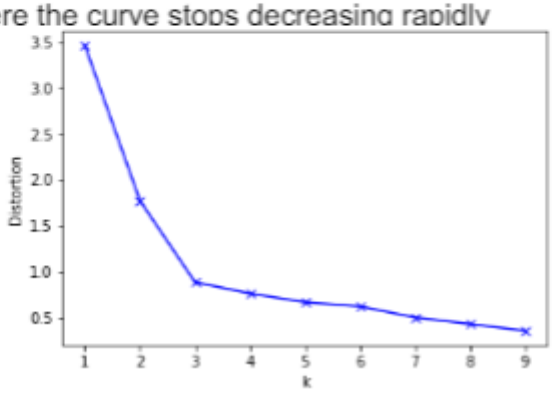
There are multiple ways to find the best k value for a given data set. However, two approaches are the most popular, namely:

- Elbow method
- Silhouette Analysis

> The following illustrations elaborate the key points of those approaches, nicely.

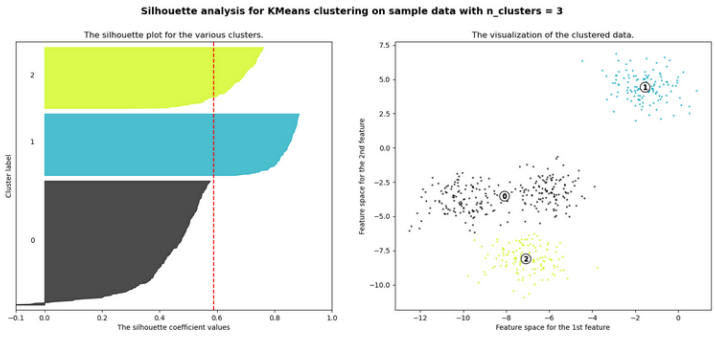
How to define *k* ?

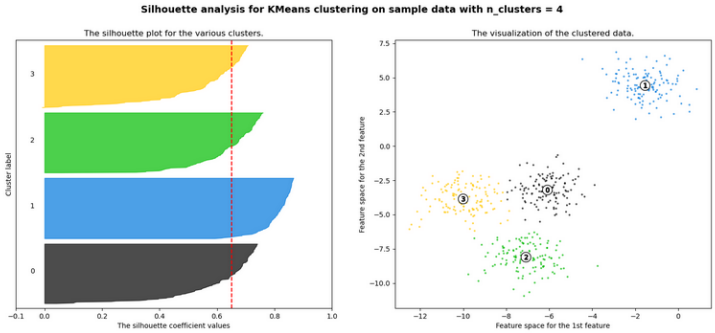
– **Elbow method**
- Run k-means for several *k* and calculate distortion for each *k*
  - » Distortion: sum of squared distances of each point to the center of the closest cluster
- Plot distortion vs *k*
- Look for k where the curve stops decreasing rapidly



How to define *k* ?

– **Silhouette Analysis**
- Calculate Silhouettes for various *k*
  - – Silhouette coefficient shows how far each sample is from other cluster centers
    - » +1: far from others
    - » 0: at the boundary
    - » -1: sample is on average closer to the points of another cluster
- Thickness shows cluster size (number of points assigned to cluster)
- Avoid *k* that leads to:
  - – Scores for clusters < average
  - – Individual scores < 0
  - – Big differences in cluster sizes (thickness)





Limitations of k-means:

- K needs to be specifically defined

- Final cluster assignments depend on initialization
    - Cluster assignments may be different on different runs
    - K-means may not achieve the global optimum
- Can describe only convex clusters geometries
- Computationally demanding as the number of points dimensions increases

3D Point Cloud Segmentation by k-means Point Cloud Expert: Florent Poux Medium

My take on that is, that in terms of dimensionality it should be 3, for k means and mean shift respectively, since clustering methods usually act in a manner of dimensionality reduction. However, there might be exceptions of that rule? Or maybe, I'm wrong with my assumption altogether.

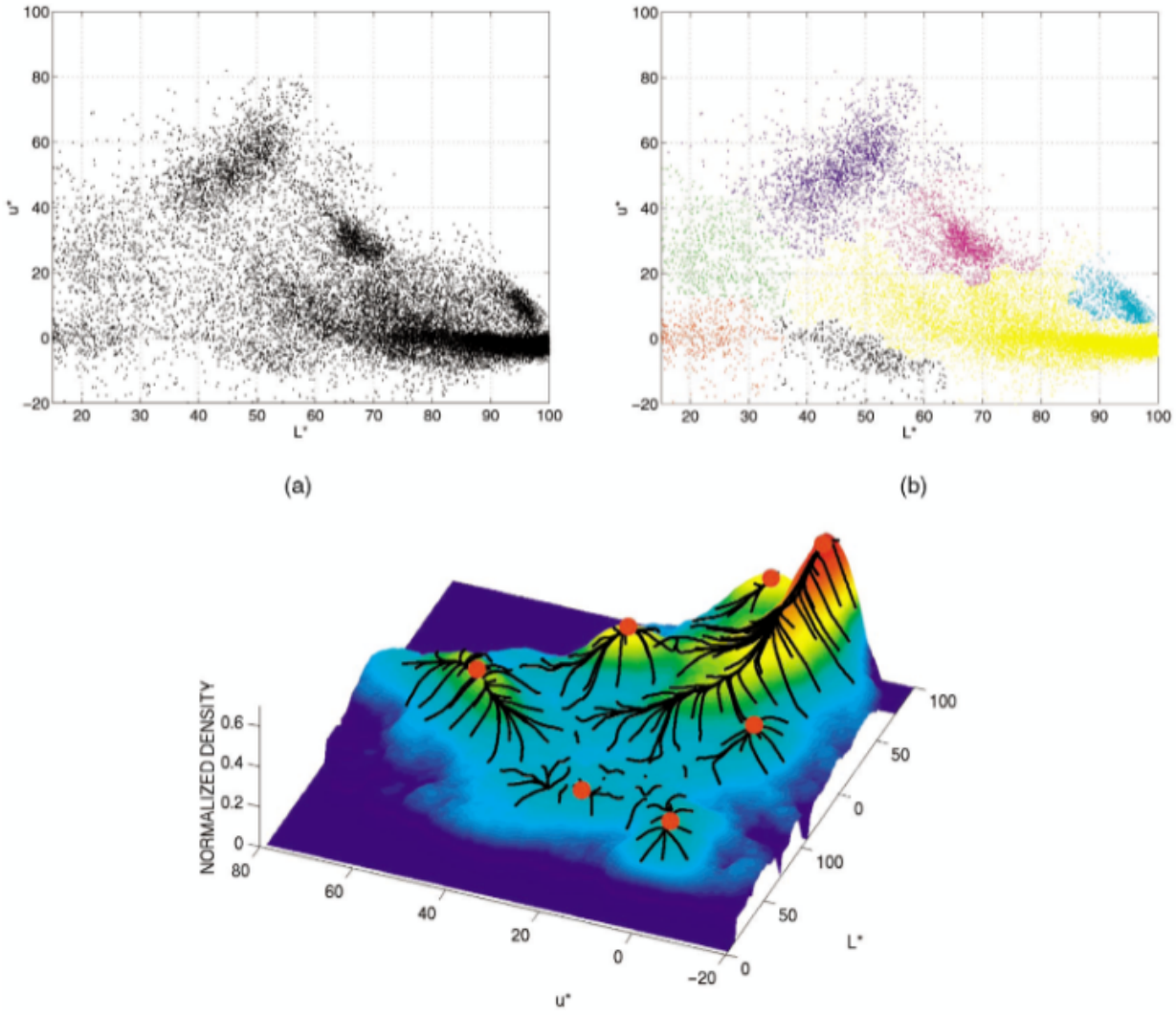
Clustering - Mean shift

Definition: Mean Shift is an algorithm that finds the maxima - the modes - of a density function given discrete data sampled from that function. Thus, it is a density hill climbing algorithm

- Every point gives rise to a cluster.
- Each cluster is defined by the radius, a.k.a "bandwith", h of its region, and a kernel function K that is used to calculate the contribution of the points included within this radius.
- In the end, only unique clusters are considered.
- So, we do not need to set the number of clusters!
    - However, we need to define the bandwith (and the kernel function).

Further definitions:

- Attraction basin: the region for which all trajectories lead to the same mode
- Cluster: all data points in the attraction basin of a mode



Pros	Cons
General, application-independent tool	Output depends on window size
Model-free, does not assume any prior shape (spherical, elliptical, etc.) on data clusters	Window size (bandwith) selection is not trivial
Just a single parameter (window size h)	Computationally (relatively) expensie (~2s/image)
Finds variable number of modes	Does not scale well with dimension of feature space
Robust to outliers	

Kernel density estimation

Kernel density estimation function

$$\widehat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

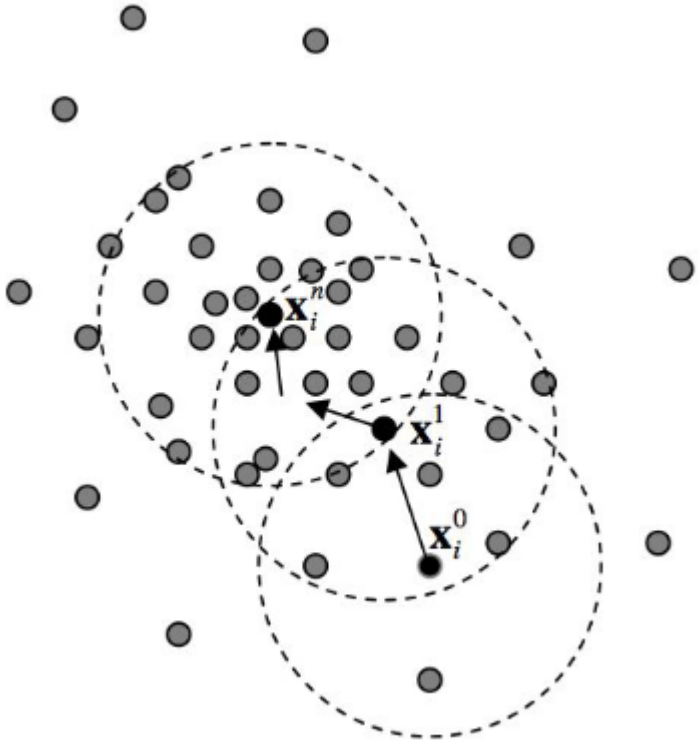
Gaussian kernel (typically used)

$$K\left(\frac{x - x_i}{h}\right) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h^2}}.$$

Mean shift clustering

The mean shift algorithm seeks modes of the given set of points

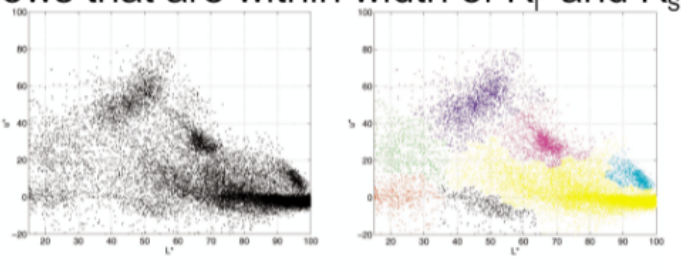
1. Choose kernel and bandwidth
2. For each point:
    1. Center a window on that point
    2. Compute the mean of the data in the search window
    3. Center the search window at the new mean location
    4. Repeat (2,3) until convergence
3. Assign points that lead to nearby modes to the same cluster

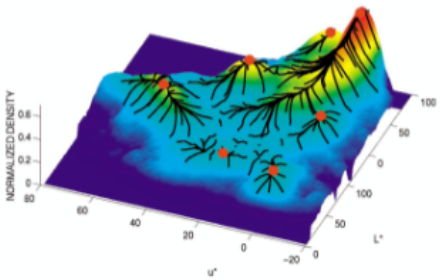


> Mean shift procedure. Starting at data point x_i, run the mean shift procedure to find the stationary point of the density function. Superscripts denote the successive window centres, respectively, and the dotted circles denote the density estimation windows.

2D image Segmentation by Mean Shift

- Compute features for each pixel (color, gradients, texture, etc)
- Set kernel size for features K_f and position K_s
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge windows that are within width of K_f and K_s





3D Point Cloud Segmentation by Mean ShiftPoint Cloud Expert: Florent Poux Medium

My take on that is, that in terms of dimensionality it should be 3, for k means and mean shift respectively, since clustering methods usually act in a manner of dimensionality reduction. However, there might be exceptions of that rule? Or maybe, I'm wrong with my assumption altogether.

Clustering - DBSCAN Article

Definition: DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise and is a unsupervised density-based clustering algorithm.

DBSCAN requires only two parameters: ϵ and minPoints. ϵ is the radius of the circle to be created around each data point to check the density and minPoints is the minimum number of data points required inside that circle for that data point to be classified as a Core point.

In density-based clustering:

- we partion points into dense regions separated by not-so-dense regions.
- a cluster is defined as a maximal set of density-connected points
- we can discover clusters of arbitrary shape

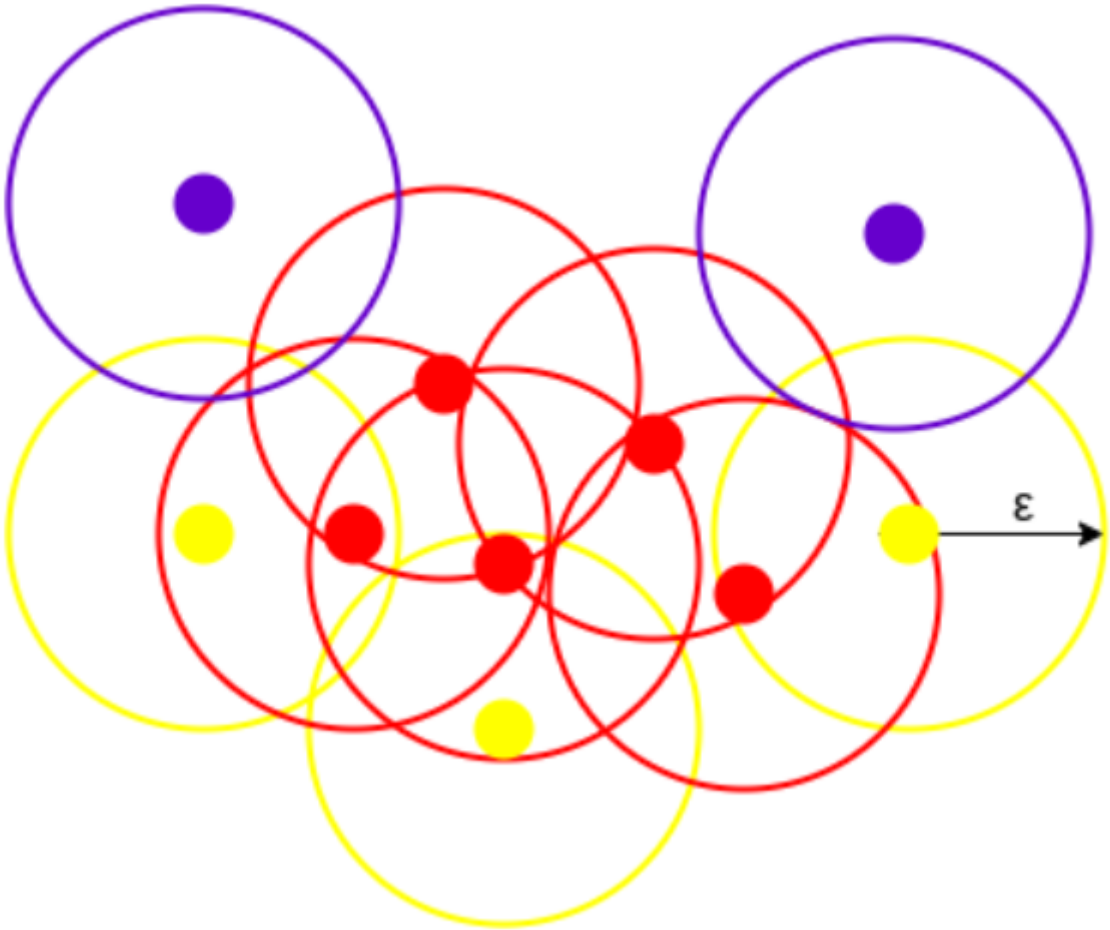
Density definition:

- Density at point p is defined as the number of points within a circle/sphere of radius ϵ
- A region is dense if the circle/sphere of radius ϵ contains at least MinPts points.

Types of points:

- A core point has more than a specified number of points (MinPts) within ϵ
- A border point has fewer than MinPts within ϵ, but is the neighborhood of a core point.
- A noise point is any point that is not a core point or a border point.

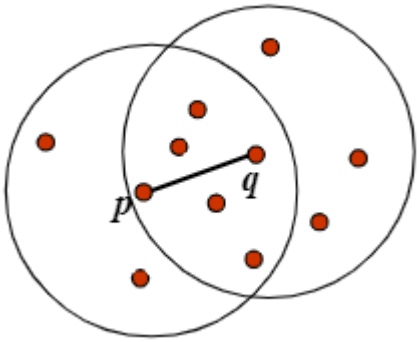
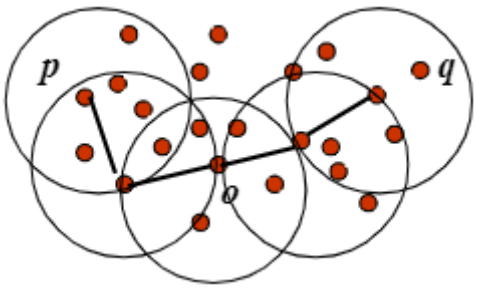
Example



> All the data points with at least 3 points (MinPnts = 2) in the circle including itself are considered as Core points represented by red color. All the data points with less than 3 but greater than 1 point in the circle including itself are considered as Border points. They are represented by yellow color. Finally, data points with no point other than itself present inside the circle are considered as Noise represented by the purple color.

Reachabillity and Connectivity

Reachability states if a data point can be accessed from another data point directly or indirectly, whereas Connectivity states whether two data points belong to the same cluster or not.

Density edge	Density-connected
	

Density edge	Density-connected
We place an edge between two core points q and p if they are within distance ϵ	A point p is density-connected to a point q if there is a path of edges from p to q

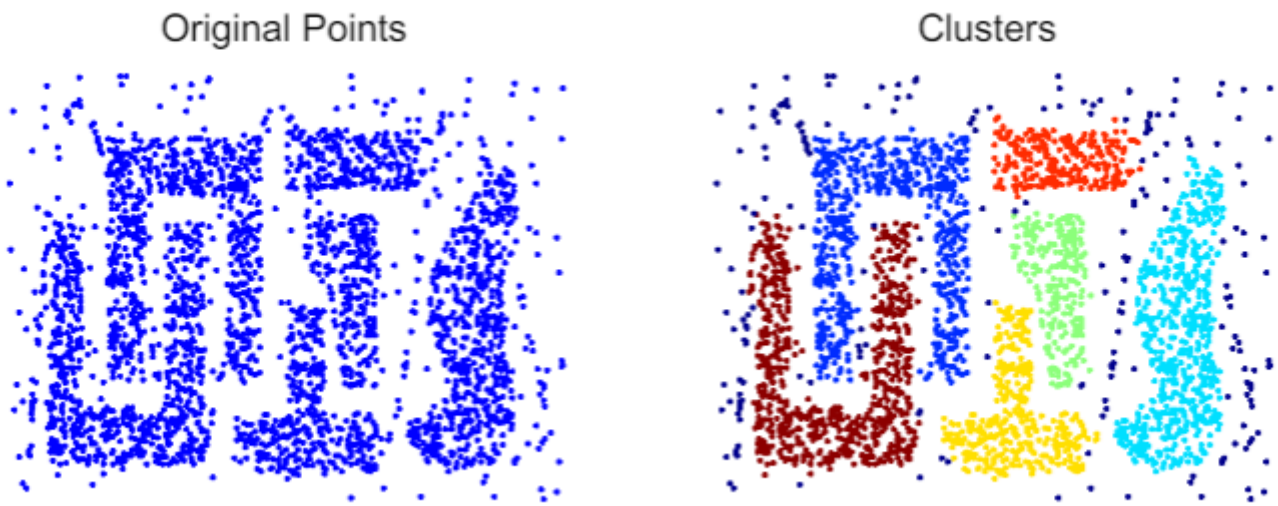
> There are other accessibility and connectivity metrics: e.g. direct density-reachable and density-reachable. See the linked article for further explanation.

DBSCAN algorithm

1. Label points as core, border and noise
2. Eliminate noise points
3. For every core point p that has not been assigned to a cluster
    - Create a new cluster with the point p and all the points that are density-connected to p.
4. Assign border point to the cluster of the closest core point.

> For locating data points in space, DBSCAN uses Euclidean distance, although other methods can also be used (like great circle distance for geographical data). It also needs to scan through the entire dataset once, whereas in other algorithms we have to do it multiple times.

Visualization



Pros	Cons
Can create clusters of arbitrary shapes	Some points (noise points) are not assigned to any cluster (is this necessarily bad?)
No need to define the number of clusters	Need to define ϵ and MinPts
Resistant to noise	Problems with point clouds that contain regions of varying densities

Hierarchical Clustering

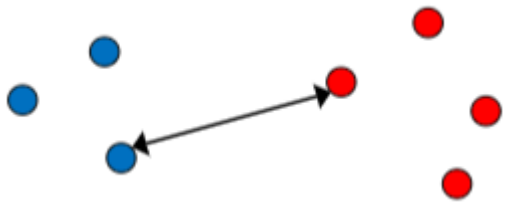
Hierarchical methods create a hierarchy of splits/merges of a dataset:

- **agglomerative methods (bottom-up):**
    1. start with each data object being one cluster
    2. iteratively merge the clusters
    3. stop when all clusters are merged or a stopping condition is met
- **divisive methods (top-down):**
    1. start with all data objects being one cluster
    2. iteratively split each cluster into smaller ones
    3. stop when each object forms its own cluster or a stopping condition is met

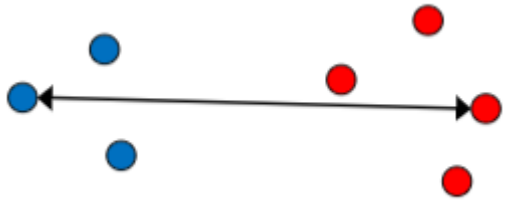
> - merging or splitting is done based on dissimilarities (= distances, so-called "linkages") or on densities
> - a merging or splitting step can not be undone

Linkage variants

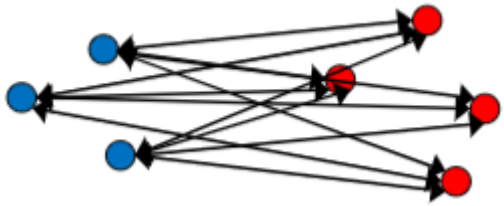
Single-link: the distance between two clusters is equal to the minimum distance (dissimilarity) between the two most similar data objects of the two clusters.



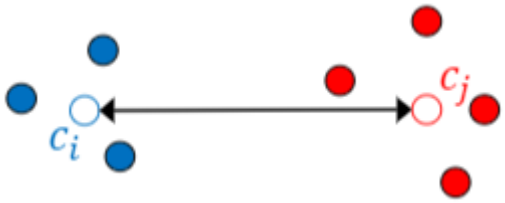
Complete-link: the distance between two clusters is equal to the maximum distance (dissimilarity) between the two most similar data objects of the two clusters.



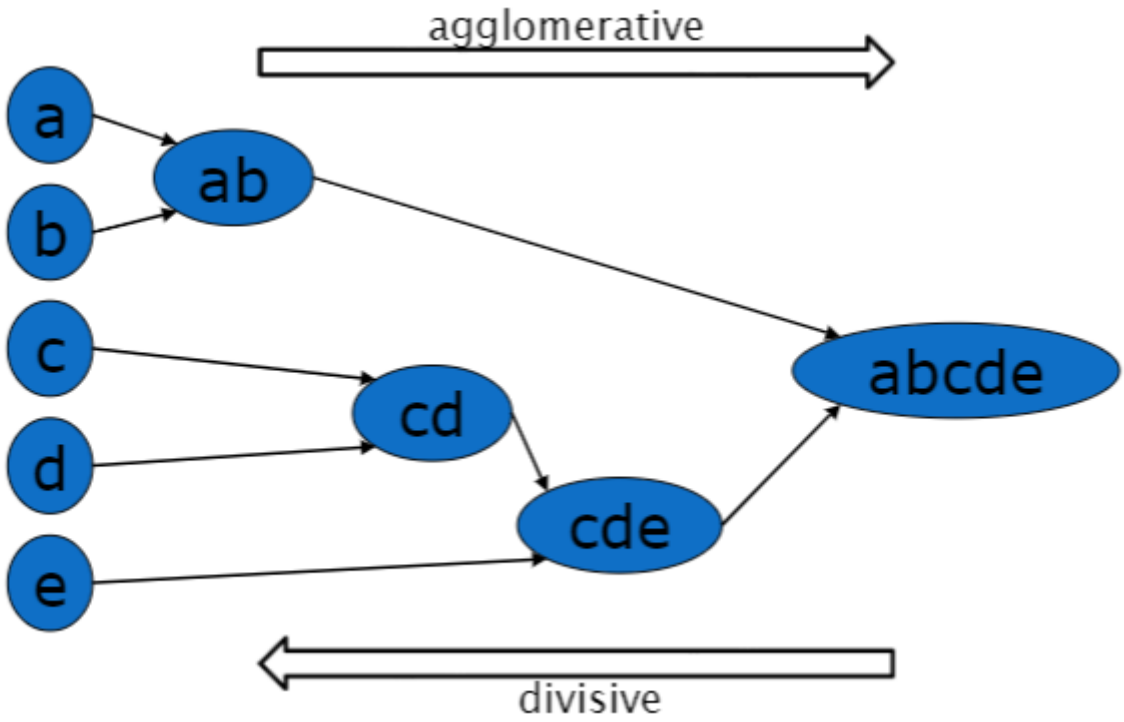
Average-link: the distance between two clusters is equal to the average distance from any member of one cluster to any member of the other cluster.



Centroid: the distance between two clusters is equal to the dissimilarity between the centroids, e.g. the mean value vectors of those two clusters.



The following figure illustrates the process of the hierarchical methods:



Hierarchical clustering representation: The output of the hierarchical clustering is a dendogram.

A dendogram:

- Consists of many Π-shaped lines that connect data points in a hierarchical tree like structure.
- The height of each Π represents the distance between the two data points being connected.

Characteristics of Hierarchical Clustering:

- Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level
- They may correspond to meaningful taxonomies

Illustration of a dendogram

dendrogram of iris dataset
(with colours corresponding to cluster assignments, for 3 clusters)