CV Summary

Version 2.0

Summary

im Fachgebiet Computer Vision

vorgelegt von: Christopher Syben

Studienbereich: Master Informatik

© 2014

This is a summary of the lecture "Computer Vision"given by Elli Angelopoulou. Based on the Slides from Summer Semester 2014 and on the Videos from Summer Semester 2012. It's sadly not a thorough Summary cause of the very short time (changed exam type and date.....), but i tried to focus on the Topics which are frequently asked in the BraunDumps from the oral exams and on the Topics which seems to be important for Elli Angelopoulou. The theory should be correct. For a correction of grammar and spelling mistakes was not enough time, so sorry for the bad language :)

Abstract

Abstract

Inhaltsverzeichnis

Abkürzungsverzeichnis **IV**

Abbildungsverzeichnis **V**

Tabellenverzeichnis **VII**

1. Image Formation **1**
1.1. Pinhole Camera and Lenses 1
1.2. Optical properties 5
1.3. Geometry and Coordinates 9

2. Image Capture **11**

3. Noise, Filtering and Smoothing **12**
3.1. Noise 12
3.2. Filtering 12

4. Edge Detection **16**
4.1. Edge Detection Steps 16
4.2. Gradient-Based Edge Detection 17
4.2.1. Canny Edge Detection 18
4.3. Second Order Derivative Edge Detectors 19
4.4. Gaussian Pyramid 20

5. Texture **22**
5.1. Image Segmentation 22
5.2. Texture Detection 24
5.3. Texture Mapping 24

6. Color **25**
6.1. Trichromacy 25
6.1.1. Grassmann's Law 26

7. HoughTransform **27**
7.1. Template Matching 27
7.2. Hough Transform 27

8. Deformable Contours **31**
8.1. Procedure 31
8.2. Forces 31
8.3. Minimization 33

9. Binocular Stereo **34**
9.1. Basic Binocular Stereo Setup 34

10. Structured Light **37**
10.1. Parenthesis: active vs. passive 37
10.2. Basic concept 38

11. Multiview Geometry **39**
11.1. Epipolar Plane 39
11.1.1. Key Points of Epipolar Geometry 42
11.1.2. Estimation of **E** (or **F**) 42

12. Motion **44**
12.0.3. Optical Flow vs. Motion Field 45
12.0.4. Differential Methods 48

13. Kalman Filter **49**
13.1. Dynamic System 49
13.1.0.1. Noise 51
13.1.0.2. Kalman Filter Setup (summary) 51
13.1.1. KF Steps 52
13.2. Conclusion 53

14. Particle Filter **54**
14.1. Setup Particle Filters 54
14.2. PDF Estimation and Importance Sampling 56
14.3. Advantages of Particle Filters 57

15. Remarks from the Exercises **59**
15.1. Gaussian and Kernel size 59
15.2. Perspective Projection 59
15.3. Convolution 59
15.4. Corner Detection 60
15.5. Gabor Filters 60
15.6. Biased Hough Transform 60
15.7. Hough Transform and tracking 60
15.8. Stereo 61

15.9. Calibration 62

A. Anhang **i**

Abkürzungsverzeichnis

Abk. Abkürzung

Abbildungsverzeichnis

1.1. Pinhole camera 1
1.2. thin-Lens 2
1.3. Lens 3
1.4. Lens 4
1.5. Lens 5
1.6. The definition of Solid Angle 6
1.7. Irradiance measure the concentration of Power on the Patch 7
1.8. Radiance 7
1.9. light-surface-camera 8
1.10. radiometry of the image formation process (only for understanding no exam sketch) 8
1.11. BRDF Setup 9
1.12. World coordinate system and Camera coordinate system 10
1.13. Radial Distortion 10

2.1. Color Camera Systems 11

3.1. Frequency Response of Box-Filter(left) and the Gaussian-Filter(right) 15

4.1. Different kind of Edges 16
4.2. Laplacian 19
4.3. Gaussian Pyramid 21

5.1. Example Spot Filter 23
5.2. Example Bar Filter 23
5.3. minimal Filterbank 24

7.1. HT - Line Detection 28
7.2. HT - Line Detection 28
7.3. HT - Line Detection 29
7.4. HT - Line Detection 30
7.5. HT - Line Detection 30

8.1. HT - Line Detection 32

Abbildungsverzeichnis

9.1. HT - Line Detection 34
9.2. The triangulation 35
9.3. Small Baseline (left); Large Baseline(right) 35
9.4. Vergence 36
9.5. Rectification 36

11.1. The triangulation 39
11.2. Way to one coordinate System 41

12.1. Effects which lead to wrong conclusions by only subtract Images . . 44
12.2. Aperture Problem 45
12.3. Optical Flow 45
12.4. Projection of the motion Vector on the image plane 46
12.5. The Barber's Pole Illusion 46

13.1. Combine $\hat{\vec{x}}_k^-$ and $\vec{z}_k$ 52
13.2. Steps of the Kalman Filter 53

14.1. importance Sampling 56
14.2. Resampling Example 57

15.1. Fast Circle Detection using gradient pair vectors 61
15.2. An Effective and Fast Soccer Ball Detection and Tracking Method . 61
15.3. Calibration Pattern 62

Tabellenverzeichnis

1. Image Formation

There are three major components that determine the appearance of an image.

- Geometry of the scene and of the projection to the camera
- Optical properties of the materials in the scene and of the sensors
- Illumination conditions

1.1. Pinhole Camera and Lenses

Most of the Time we use the model of the Pinhole Camera (figure 1.1). The hole of the Camera is so small, that only one Light-Ray goes trough. Advantages of the

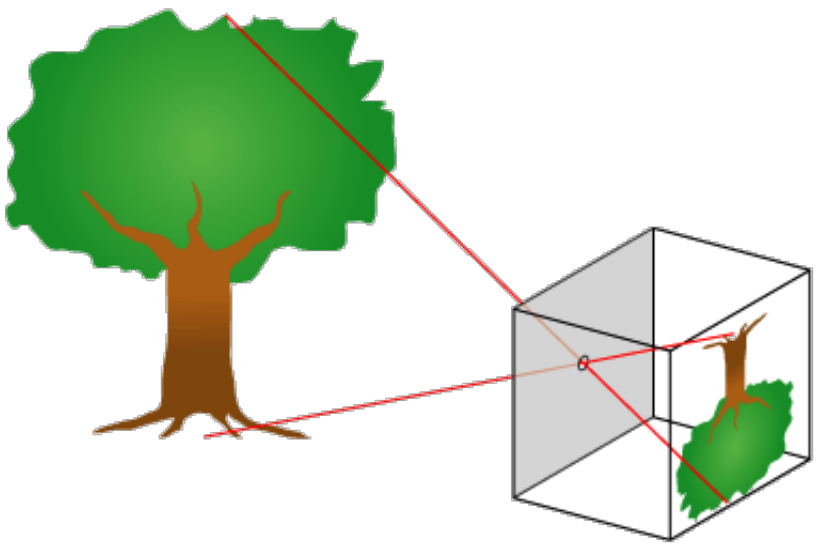


Abbildung 1.1.: Pinhole camera

Pinhole Camera:

- The Projection is very simple because only one lightray can pass the hole.
- clear one-to-one Relationship between the 3D-Point in the scene and the 2D-Point in the Image

- very crisp Images of stationary scenes

Disadvantages of the Pinhole Camera:

- needs a long exposure Time to get enough light on the sensors. This leeds to a problem for scene which contains moving objects
- very blurred Images for scenes which are not stationary

A Solution for the stationary-scene Problem would be to open the Hole. But this leads also to blurred Images (the clear 1:1-Relationship gets violated). A Solution is to use a Lens fore more Light but preserve the 1:1-Relationship

Lens

The Lens gather a cone of Rays from a Point $\vec{P}$ to a single Point $\vec{p}$ on the Other side (Figure 1.2). Thats why the Image is much brighter but not blurred.
Most of the time we use the model of a thin Lens (figure 1.2) Where the Center of

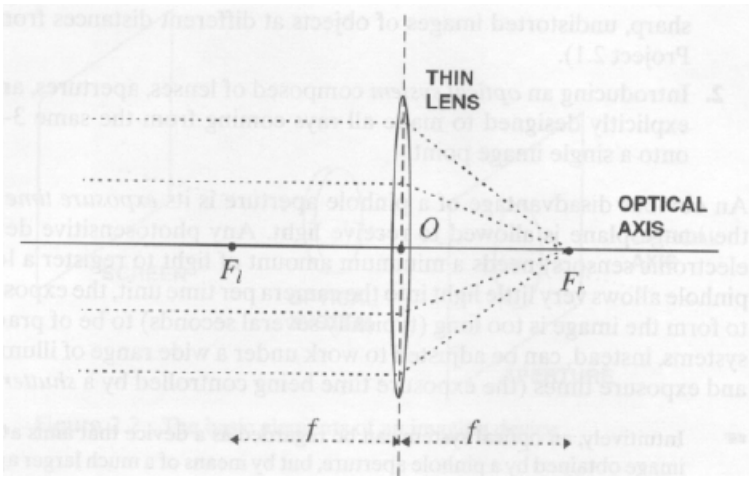


Abbildung 1.2.: thin-Lens

projection is center of the lens, the Point O and the optical axis is perpendicular to the plane of the lens and passes trough the Center of Projection.
The property of the thin Lens is that their exists two focal Points $\vec{F_l}$ and $\vec{F_r}$. The focal Point is defined in that way, that all Rays which are parallel to the optical Axis (which means they are perpendicular to the plane of the Lens) will converge at a single Point, which is the focal Point of the Lens. The distance of the focal Point to the center of the Lens is determined by f and is called focal length. This value is a property of the Lens, e.g. if you use a 12 mm Lens you mean that $f = 12\ mm$.
Now use that information to use a lens in the Context of Imaging.
On the right side from the Lens is the Sensor and on the left side is the scene.

The Thin Lens Law (eq. 1.1) is derived from Figure 1.3 where you can find similar

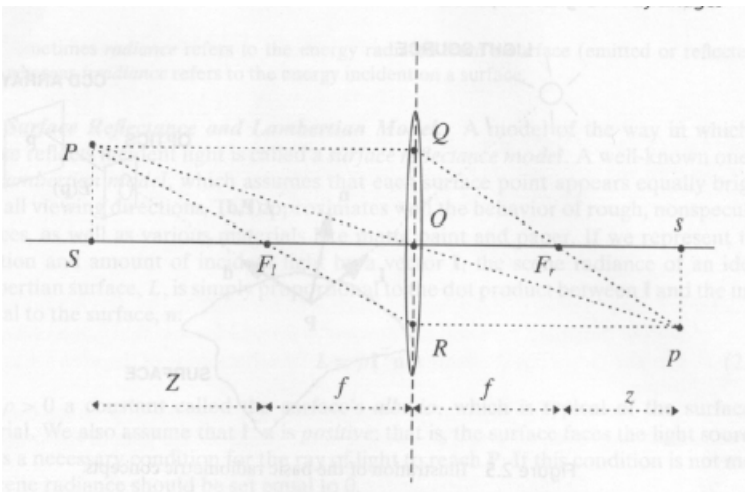


Abbildung 1.3.: Lens

Triangles and combine the equations.

$$\frac{1}{f} = \frac{1}{z+f} + \frac{1}{Z+f} \tag{1.1}$$

The Law express exactly the relationship between the distance in the scene and the distance to the image plane.
The implication of this equation is:
if my Object that I'm Imaging is infinitely far away the distance where the object is projected approaches the focal length.

$$(Z+f) \to \infty \Rightarrow (f+z) \to f \tag{1.2}$$

For all following applications we assume that $(z+f) = f$ which is not true but simplify the math.
Another important aspect is the effective diameter of a Lens d which is adjustable with a iris.

Image Blur

In figure 1.4 you can see how the Iris can change how the image will be blurred. There exists a lower bound for sharpness: If the Blur circle gets smaller than one Pixel then your Image will not get sharper independent how much you close the iris

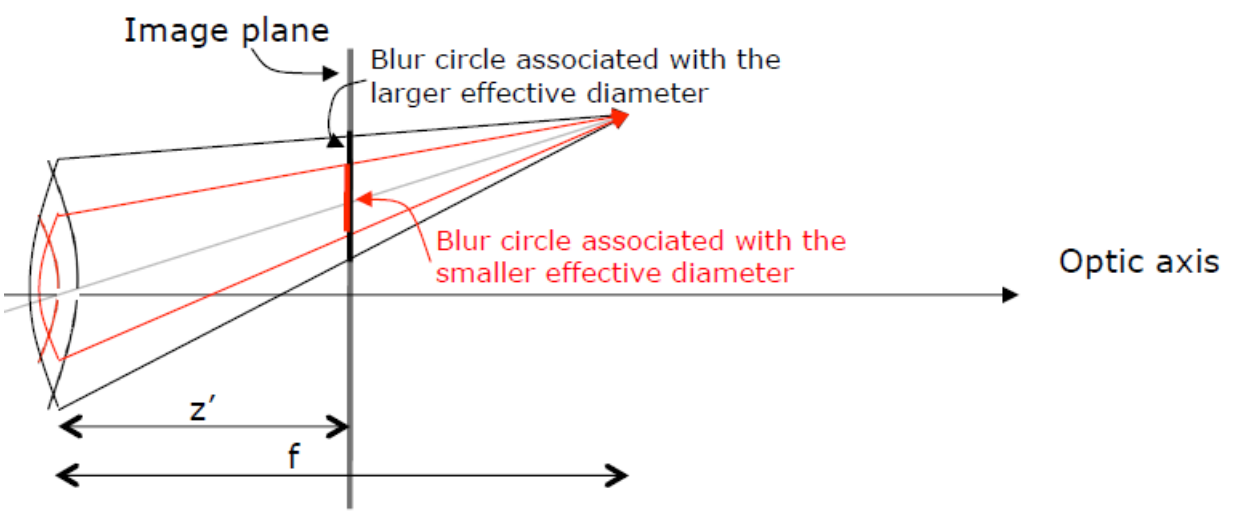


Abbildung 1.4.: Lens

Depth of Field

The depth of field is the distance between the nearest and farthest points in the scene that appear in focus. It is defined as the difference between the far and near planes.

1.2. Optical properties

All following discussions consider an infinitesimal area patch dA centered around a point $\vec{P}$. The d's (like dA) express that we are handling with small values.

Foreshortening

A patch dA can have different orientations with respect to the image plane (figure 1.5). Every patch whose surface normal forms an angle with the optical Axis has a foreshortened area. The tilted patch appears smaller to the viewer, i.e. foreshortened.

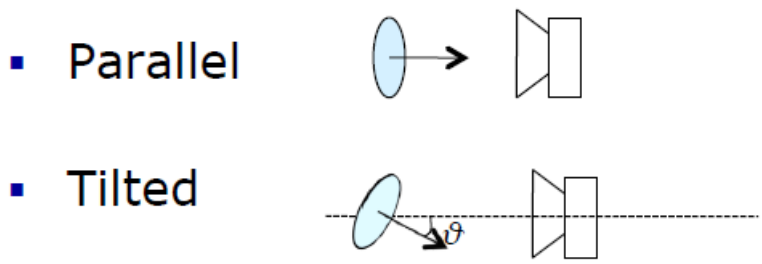


Abbildung 1.5.: Lens

Foreshortening cause different effects:

- How bright a patch appears to the viewer, $E(p)$
- How much light falls on a patch and hence how bright it is, $L(P)$

Solid Angle

In 3D you cant describe an Angle as easy as in the 2D World where you can use 2 Lines. Therefore its much easier to use the Patch $d\omega$ on the unit Sphere to describe the Angle.
The Solid Angle is a way to describe the Angle of the Cone from a Point $\vec{P}$ to the patch dA.

- Step1: Draw a unit sphere centered at q
- Step2: Project dA on the perimeter of that sphere
- Step3: The area of that projection is the angle in steradians. It depends on:
  - The angle ϕ between the normal to the patch dA and the radius that reaches the center of dA.

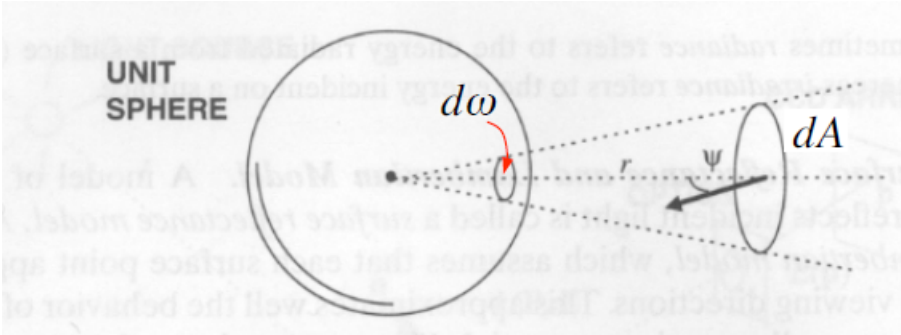


Abbildung 1.6.: The definition of Solid Angle

- The distance r between the center of the patch dA and the center of? t he circle q
-

$$d\omega = \frac{dA \cdot \cos\psi}{r^2} \tag{1.3}$$

Light

We are interested in the Power of the Light which reaches our camera.

- Light: electromagnetc energie, Q
- Flux: Power carried by the EM radiation

$$P = \frac{dQ}{dt} \tag{1.4}$$

- Intesity: Power of light traveling in a specific direction

$$I = \frac{dP}{d\omega} \qquad \text{where} \quad d\omega \text{ is the Solid Angle} \tag{1.5}$$

Irradiance

Irradiance is the power of light falling on a surface patch (figure 1.7):

$$E = \frac{dP}{dA} \qquad in \quad \frac{\text{Watt}}{m^2} \tag{1.6}$$

It is **independent** of direction! If you want to measure the amount of Light which is leaving (reflected) the Patch you use also the Irradiance.

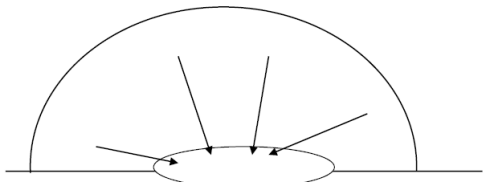


Abbildung 1.7.: Irradiance measure the concentration of Power on the Patch

radiance

Radiance is the power of light falling on a surface patch from a specific direction.

$$L = \frac{d^2 P}{d\omega \cdot dA \cdot \cos\theta} \qquad in \qquad \frac{\text{Watt}}{\text{steradians} \cdot m^2} \tag{1.7}$$

Radiance means look for the power (the d^2 shows that the power is very small) with respect to both the solid Angle and the foreshorten Area. Its also the measurement

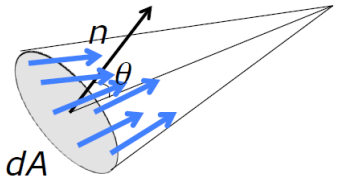


Abbildung 1.8.: Radiance

if you want know the amount of Power which is leaving the patch in the **direction** of a Point $\vec{P}$.

Light-Surface-Camera

The whole Situation is drawn in figure 1.9 The Intensity which reaches the CDD-Sensor at the Point $\vec{p}$ is described by:

$$I(p) \propto E(p) \propto L(p) \tag{1.8}$$

Where $E(\vec{p})$ is the irradiance which reach the Sensor at Point $\vec{p}$ and $L(\vec{p})$ is the radiance leaving the surface in the direction of the camera.
The Irradiance at Image Point $\vec{p}$ on the Image Plane (figure 1.10) is described by:

$$E = L\frac{\Pi}{4}\frac{d^2}{f^2}\cos^4\alpha \tag{1.9}$$

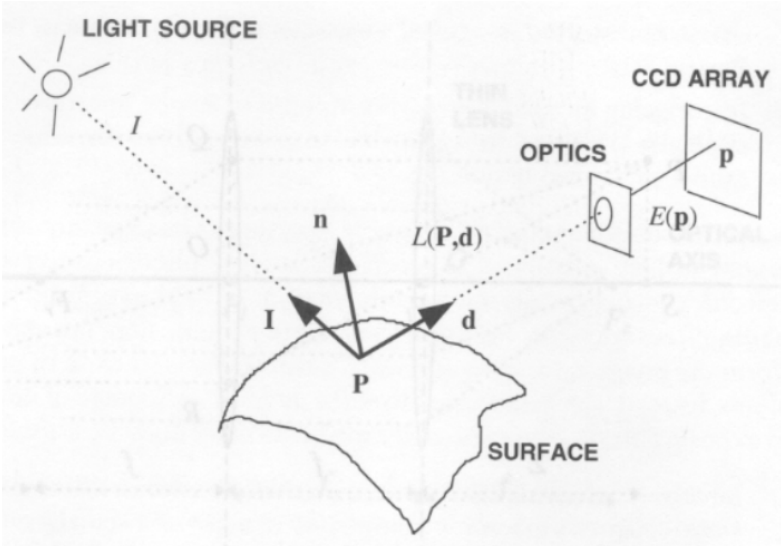


Abbildung 1.9.: light-surface-camera

The irradiance incident on the Image Plane at the small Patch dI is proportional to the radiance L leaving the Point $\vec{P}$ from the small patch dO reaching the lens multiplied by the ratio $\frac{\Pi}{4}$, by the effective Lens diameter d^2, the focal Length f^2 and by $\cos^4 \alpha$. The $\cos^4 \alpha$ is an important Aspect:

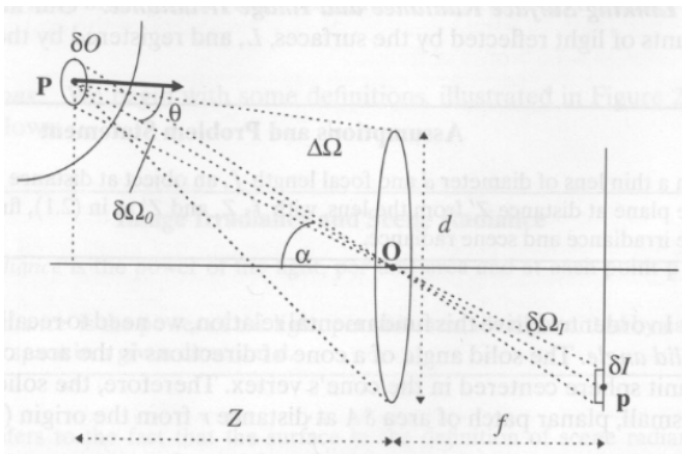


Abbildung 1.10.: radiometry of the image formation process (only for understanding no exam sketch)

That means if you use a Lens to take a Picture of a uniform Area, like a white Wall, which is uniformly illuminated then the light is dimmed the more the Angle α increases although the amount of light leaving the different Points from the Wall are equal.
The white wall in your Image will not get the same Values although the White has the same color and is uniformly illuminated !

The next Important Topic is the relationship between light falling on a surface and gets reflected.
Here we use the Bidirectional Reflectance distribution Function (BRDF)(for variable see figure 1.11):

$$\text{BRDF}(\vartheta_r, \phi_r, \vartheta_i, \phi_i) = \frac{dL_r(\vartheta_r, \phi_r)}{dE_i(\vartheta_i, \phi_i)} = \frac{\text{reflected radiance L}}{\text{incident irradiance E}} \tag{1.10}$$

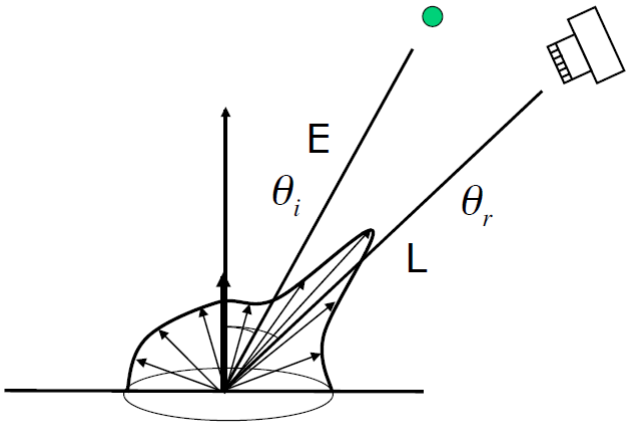


Abbildung 1.11.: BRDF Setup

1.3. Geometry and Coordinates

The Coordinate Origin is the Center of Projection. The pinhole camera model implies perspective projection, i.e. all projection rays pass through a single point (COP). For some Algorithm its easier to put a "Virtual Image Planeïn Front of the Lens at the focal length f (in this case the Image is not flipped)

Extrinsic Camera Parameters

Extrinsic parameters: A set of geometric parameters that uniquely identify the transformation between the unknown camera frame and a known reference frame (the world reference frame). The relationship between the coordinates of a point P in world Pw and camera Pc frames is:

$$P_c = R(P_w - \vec{t}) \tag{1.11}$$

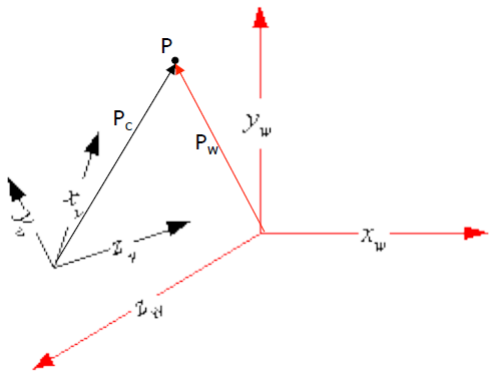


Abbildung 1.12.: World coordinate system and Camera coordinate system

- The relative position of the 2 optical centers
- The relative orientation of the two image planes

Intrinsic Camera Parameters

Intrinsic parameters: A set of geometric parameters that link the pixel coordinates of an image point to the corresponding coordinates in the camera reference frame.

Radial Distortion

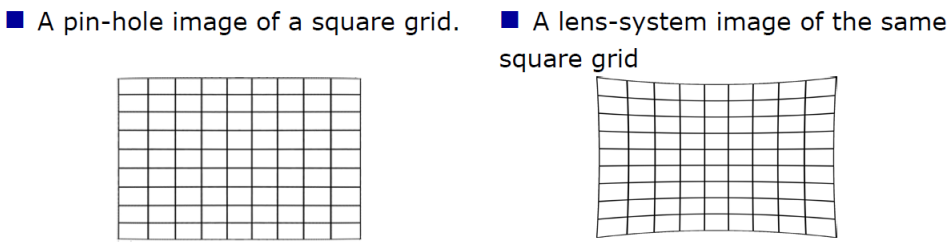


Abbildung 1.13.: Radial Distortion

The amount of distortion depends on the distance between the principal point and the pixel of interest. The distortion can be corrected.

- Pixel size
- Focal length
- Principal point

2. Image Capture

Most color cameras give a triplet of color values per pixel. Either a separate chip is used per color, or a filter composed of a mosaic of smaller individual color filters is laid over the CCD chip (figure 2.1).

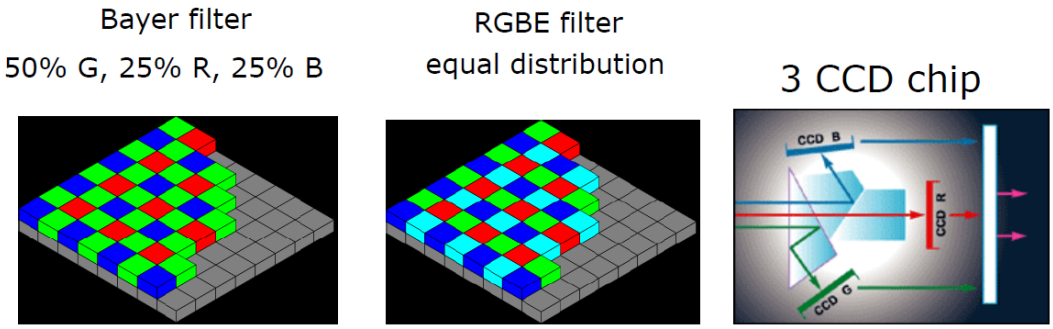


Abbildung 2.1.: Color Camera Systems

3. Noise, Filtering and Smoothing

3.1. Noise

Different Effects create Noise in our Images:

- Photon noise: variation in the #photons falling on a pixel per time interval T
- Saturation: each pixel can only generate a limited amount of charge
- Blooming: saturated pixel can overflow neighboring pixel
- Thermal noise: heat can free electrons and generate a response when there is no real signal
- Electronic noise
- Burned pixel
- Black is not black
- nonlinearity between the sensor response and the incoming #photons

The Detector Noise can be modeled as an independent additive noise which can be described by a zero-mean Gaussian.

3.2. Filtering

The Goal of Filtering, in the special Case of Noise is to clean the Image:

$$\text{Noisy Image }_{in} \longrightarrow \text{ Filter } \longrightarrow \text{ Clean Image}_{out} \tag{3.1}$$

or more General for a wide range of Transformations(which is interchangeable with Filter):

$$\text{Image }_{in} \longrightarrow \text{ Filter } \longrightarrow \text{ Image}_{out} \tag{3.2}$$

From the mathematically Point of View a Filter H can be treated as a function on an input Image I:

$$H(I) = R \tag{3.3}$$

Linear Transformation / Filter

A transformation H is linear of, for any inputs $I_1(x, y)$ and $I_2(x, y)$, here input images, and for any constant scalar α we have:

$$H(\alpha \cdot I_1(x, y)) = \alpha \cdot H(I_1(x, y)) \tag{3.4}$$

and

$$H(I_1(x, y) + I_2(x, y)) = H(I_1(x, y)) + H(I_2(x, y)) \tag{3.5}$$

The Interpretation is that a scaling of the input corresponds to a scaling of the the output and Filtering an additive image is equivalent to filtering each image separately and then adding the results.

Shift-Invariant Transformation

A transformation H is shift-invariant if for every pair (x_0, y_0) and for every input image $I(x, y)$, such that

$$H(I(x, y)) = R(x, y) \tag{3.6}$$

we get

$$H(I(x - x_0, y - y_0)) = R(x - x_0, y - y_0) \tag{3.7}$$

which means that the filter H does not change as we shift it in the image. In other words the Filter is independent of the position and behaves always the same.
For Example calculating the mean of the Neighborhood and set the current Pixel to that Value **is** a shift-invariant Filter. But calculating the median of the Neighborhood and set the current Pixel ti that Value is **not** a shift-invariant Filter because it depends on the Values which Neighborhood Pixel the Filter choose, and that is not similar behavior too chose one time the Pixel-value above- and one time the Pixel-value under the current Pixel as the target Value.

Convolution

Convolution is if a transformation respectively a Filter is linear shift-invariant then one can apply it in a systematic manner over every pixel in the image. Convolution is defined as:

$$R(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} H(x - i, y - j) I(i, j) \tag{3.8}$$

and is denoted as:

$$R = H * I \tag{3.9}$$

Another look at Convolution is that filtering often involves replacing the value of a pixel in the input image F with the weighted sum of its neighbors.
Represent these weights as an image H. In this case H is called the Kernel. The operation for computing this weighted sum is called convolution: $R = H * I$.
convolution is:

- commutative
$$H * I = I * H \tag{3.10}$$
- associative
$$H_1 * H_2(*I) = (H_1 * H_2) * I \tag{3.11}$$
- distributive
$$(H_1 + H_2) * I = (H_1 * I) + (H_2 * I) \tag{3.12}$$

An Example for such a Filter is a simple Averaging Filter which is smoothing the Image:

$$H = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} \tag{3.13}$$

A much better smoothing Kernel is a Isotropic Gaussian Filter which increase the weight the closer the Pixels are to the Center. The weights in the Kernel H are assigned according to the Gaussian function(the higher σ the more it looks like the Kernel above):

$$H(i,j) = e^{-\frac{(i^2+j^2)}{2\sigma^2}} \qquad H_{\text{Gauss}} = \begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix} \tag{3.14}$$

All Values in the Kernel sum up to 1 !(the Pixel wont get Brighter or Darker with respect to the Image)
Convolution can be very slow cause you have to process your Kernel for each Pixel in the Image. But the convolution has a nice property which can be used to speed up the calculation.
You can transform the Image I and the Kernel H with the Fourier-Transformation

into the frequency Domain. If you have done that the convolution ends up as a simple multiplication.

$$R = H * I \xrightarrow[H \text{ and } I]{\text{Fouriertransform of}} \hat{R} = \hat{H} \cdot \hat{I} \xrightarrow[\text{of } R]{\text{inverse Fouriertransform}} R \tag{3.15}$$

With using FFT this is in a lot of cases much faster than the convolution in the spacial Domain!
In Figure 3.1 you can see based on the Filters in the frequency Domain why the Gaussian-Filter has no "ringingëffects like the Box-Filter respectively the Mean-Filter (where all weights are the same) Gaussian Filtering works very well for images

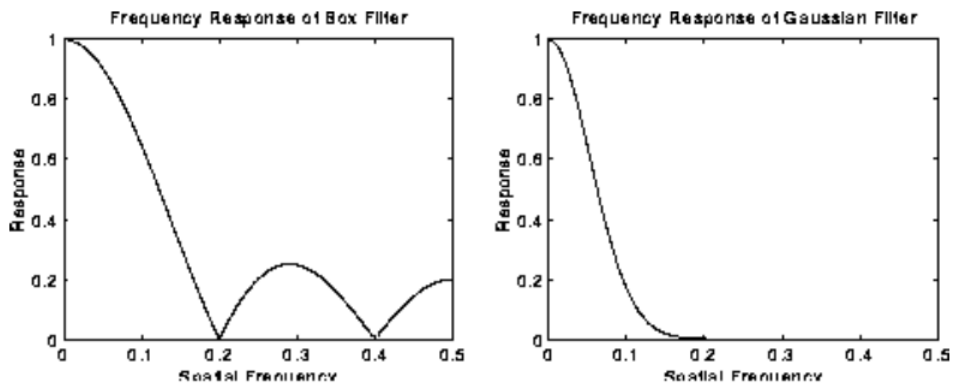


Abbildung 3.1.: Frequency Response of Box-Filter(left) and the Gaussian-Filter(right)

affected by Gaussian noise but not very good for images affected by Salt & Pepper noise.

4. Edge Detection

An edge is:

- a Significant change in intensity values
- Related to object boundaries, patterns (brick wall), shadows, etc.
- a property attached to each pixel
- calculated using the image intensities of neighboring pixels

Example of 1D Edges (figure 4.1):

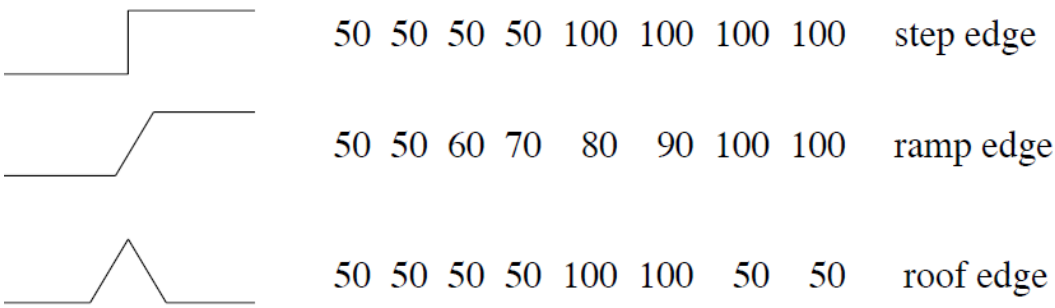


Abbildung 4.1.: Different kind of Edges

4.1. Edge Detection Steps

1. Noise Smoothing
   - Suppress as much noise as possible without destroying edge information
2. Edge Enhancement
   - Design a filter that gives high responses at edges and low response at non-edge pixels.
3. Edge Localization
   - Decide which high responses of the edge filter are responses to true edges and which ones are caused by noise or other artifacts

The Key-Concept of detecting Edges is that the detection is equivalent to detecting changes in intensity Values. When we differentiate the Image we can detect these Changes.
If we take the first derivative we get an Gradient-based edge detector.
If we take the second derivative we get an Laplacian edge detector, which looks for zero-crossings instead of extrema, which is a easier Task and so a faster computation.

4.2. Gradient-Based Edge Detection

The gradient vector $G(x, y)$, at an image pixel $I(x, y)$ is:

$$G(x, y) = \left(\frac{\delta I(x, y)}{\delta x}, \frac{\delta I(x, y)}{\delta y}\right) = (I_x(x, y), I_y(x, y)) \tag{4.1}$$

The gradient vector points in the direction of maximum change. The Orientation (its angle with the x-axis) can be calculated by:

$$\theta = \arctan\left(\frac{I_y(x, y)}{I_x(x, y)}\right) \tag{4.2}$$

Our Images we use are discrete and not continuous. Therefore we get a very simple Kernel H which detect edges based on the first derivative:

$$H_x = \begin{bmatrix} -1 & +1 \end{bmatrix} \qquad H_y = \begin{bmatrix} -1 \\ +1 \end{bmatrix} \tag{4.3}$$

This Kernel is sensitive to noise and evaluate at half-pixel locations. To handle this Problems prewitt and sobel improve the Kernel:

$$\text{Prewitt Kernel:} \qquad \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} \tag{4.4}$$

$$\text{Sobel Kernel:} \qquad \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{4.5}$$

Gradient Edge Detection Process

- Compute $I_x = H_x * I$
- Compute $I_y = H_y * I$
- Compute $\|G(x, y)\|$
- If $\|G(x, y)\| \geq t$
  then pixel (x, y) is an edge-pixel
  compute the angle θ for that pixel

4.2.1. Canny Edge Detection

The Canny Edge Detector based on gradient edge detection and is a optimal Edge Detector. Canny describe when an edge detector is optimal:

- good detection: find as many real edges as possible
- good localization: estimate the position of the edge as close as possible to its location in the image
- minimal (single) response: detect each edge only once (no ghost or ringing effects)

The Canny edge detector uses as input the output from the gradient edge detectors like Sobel or Prewitt.
There are two main steps:

1. Non-maximum suppression

Non-maximum suppression is an edge thinning technique and represents the minimal response criterion.
Non-maximum suppression examines parallel edges in small neighborhood and eliminates the ones with the smaller (not max.) gradient magnitude

Hyteresis Thresholding

After the Non-maximum suppression there are still edges which are responses to noise. Canny removes these ghost edges using thresholding.
To avoid Problems which occurs by using a too low (still ghost edges) or too high (remove real edge) Canny uses two thresholds.
The main Idea is:

- Assumption: Important edges should form continuous curves in the image.
- Idea: Follow a faint direction of a given line and discard a few noisy pixels that do not constitute a line but have produced large gradients.
- Do this by using a high threshold.
- After the high thresholding we are left with edges which are most probably real edges.
- Do a second pass tracing (following) the curves. During the tracing use the lower threshold. If an edge strength is larger than the lower threshold it is a real edge.

4.3. Second Order Derivative Edge Detectors

Another way to detect an extremal first derivative is to look for a zero-valued second derivative. The Laplacian is a function that gives the magnitude of change without direction (result is scalar):

$$\nabla^2(I(x,y)) = \left(\frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2}\right) \tag{4.6}$$

In the discrete World the Laplacian Kernel looks like:

$$H_{\text{Lap}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad H_{\text{Lap}} = \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} \tag{4.7}$$

The result is an new image with negative values on one side of the edge and positive values on the other side of the edge: The computions of the second order derivative

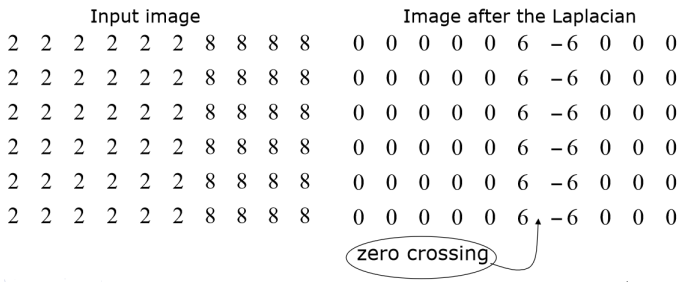


Abbildung 4.2.: Laplacian

is very sensitive to noise, cause of that smooth the Image first with a Gaussian and then apply the Laplacian. Both are Filters so we can combine them and we get the Laplacian of Gaussian:

$$R_{\text{LapEdge}} = (H_{\text{Lap}} * H_{\text{Gauss}}) * I \tag{4.8}$$

the function of the combined Filters ("mexican hat") is:

$$\nabla^2(\text{Gauss}(x,y)) = \nabla^2(e^{-\frac{(i^2+j^2)}{2\sigma^2}}) = \frac{(x^2+y^2-\sigma^2)}{\sigma^4} \cdot e^{-\frac{(i^2+j^2)}{2\sigma^2}} \tag{4.9}$$

the Kernel from the Laplacian of Gaussion looks in the discrete World like:

$$H_{\text{LoG}} = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \tag{4.10}$$

4.4. Gaussian Pyramid

Gaussian Pyramid is a hierarchy of low pass filtered versions of the original image (figure 4.3). Successive Layers correspond to lower frequencies (larger σ). Each successive layedr is also sub-sampled version of the previous level. Sub-sampling is typical by a factor of 2 in each coordinate direction. This is a form of multi-resolution analysis.
Let l be the level of the Gaussian Pyramid, then (k is typically 2):

$$G_l(x,y) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} H_{\text{Gauss}}(m,n) * G_{l-1}(2x+m, 2y+n) \tag{4.11}$$

This function is the REDUCE operation:

$$G_l = \text{REDUCE}(G_{l-1}) \tag{4.12}$$

The Pyramid is recursively constructed:

$$G_0 = I \qquad G_{l+1} = \text{REDUCE}(G_l) \tag{4.13}$$

If you expand the images to the Original size you can calculate the difference. If you do that and store the difference between two successive levels then you can build another type of pyramid based on these difference images, called the Laplacian pyramid.

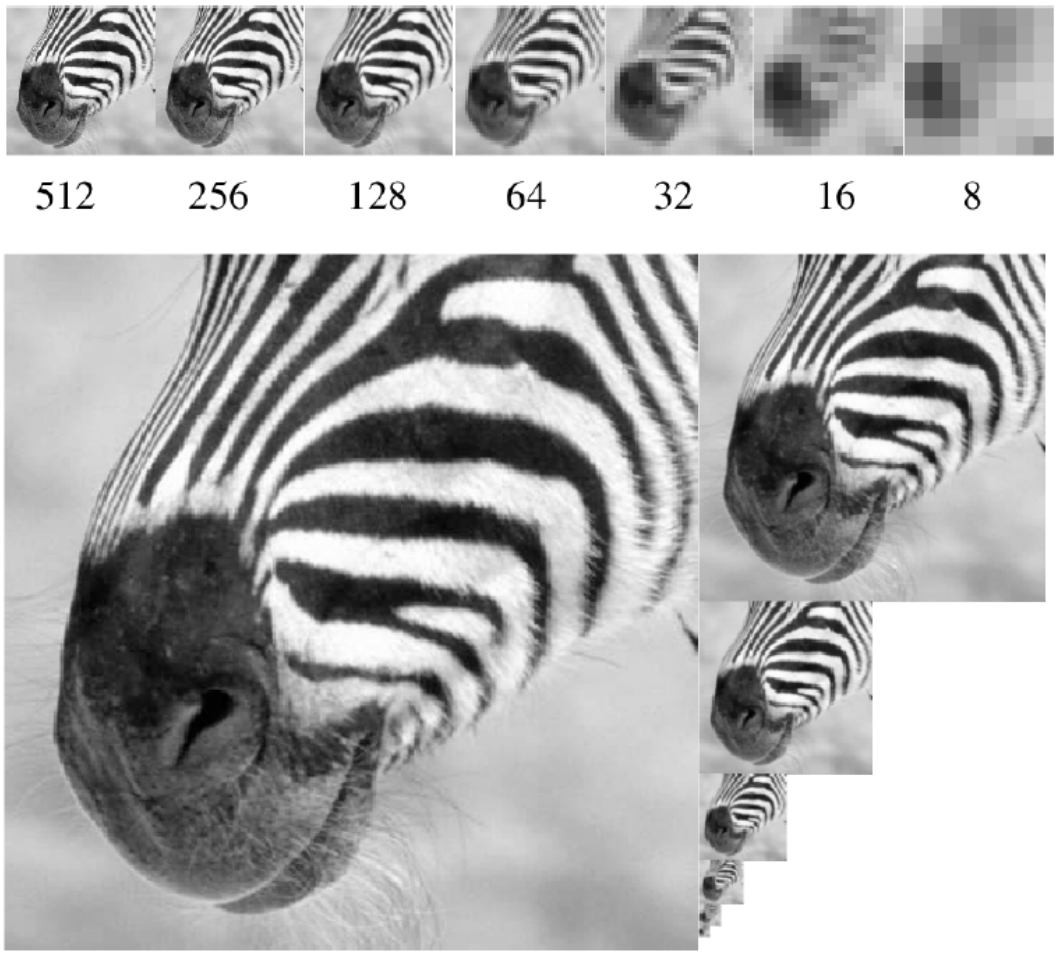


Abbildung 4.3.: Gaussian Pyramid

5. Texture

In CV we talk only about 2D-Texture. A Texture is a repeatable pattern of small elements. If you have for example only one leaf or one Brick that is not a Texture. A Brick wall or stripes are Textures!
A Texture itself can vary from highly regular to purely stochastic. Textons are the small elements that keep repeating themselves in a regular pattern. They can be thought of as the atomic untis of texture.
Textures are used for:

- Image Segmentation
  Since texture is an important clue in images, the goal of image segmentation is to identify continuous regions of uniform texture.
- Texture Synthesis
- Shape from Texture
  The key idea behind shape from texture is to exploit texture deformations to infer 3D shape.

5.1. Image Segmentation

The Goal is to Segment an image in disjoint regions, where each region has a distinct texture. To Achieve that we need to identify unique textures ot at least distinguish one form of texture from another.
Its very difficult to identify Texture in a Image because you don't know what in the Image a Texture is. It's something like trying to read Language without knowing the alphabet.
To come up with a Solution you should think of popular subelements and then design filters which will produce a high response if a subelement is present.
The Human vision suggests that spots and bars are different scales and orientation are valid filters for finding subelements.

Spot-Filter

The Goal is to design a filter that when it is superimposed with a spot it will give a high response. A common spot filter is a weighted sum of symmetric Gaussians (figure 5.1):

$$S = w_1 G(\sigma_1) + w_2 G(\sigma_2) + w_3 G(\sigma_3) \quad \text{where} \quad G(\sigma) = e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \tag{5.1}$$

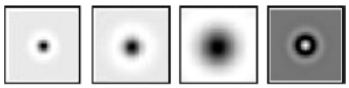


Abbildung 5.1.: Example Spot Filter

Bar-Filter

The Goal is to design a filter that when it is superimposed with a spot it will give a high response. When we stretch the Gaussian in one direction it becomes a ridge, a bar (figure 5.2):

$$S = w_1 G(x, y_1, \sigma_{x_1}, \sigma_{y_1}) + w_2 G(x, y_2, \sigma_{x_2}, \sigma_{y_2}) + w_3 G(x, y_3, \sigma_{x_3}, \sigma_{y_3}) \tag{5.2}$$

$$\text{where} \quad G(x, y_c, \sigma_{x_c}, \sigma_{y_c}) = e^{-\left(\frac{x^2}{2\sigma_x^2} + \frac{(y-y_c)^2}{2\sigma_y^2}\right)} \tag{5.3}$$

To find all orientations of bars and not only Horizontal, so use bar filters with

Abbildung 5.2.: Example Bar Filter

different Orientations

Filter Bank

Studies have shown that we need at least 6 orientations. Use a lot of dots and bar filters with different Orientation and Scaling is called a filter Bank (figure 5.3).

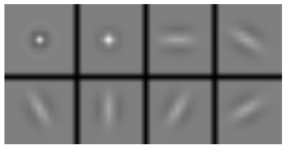


Abbildung 5.3.: minimal Filterbank

5.2. Texture Detection

5.3. Texture Mapping

wenn ich noch lust hab

6. Color

There are inaccuracies in using the Word Color. You can use the word color to refer to the color you perceive an object has. Or you can use the word color to describe the part of the visible light that is not absorbed by the object (physical description). A very Important aspect is that the Color depends also on the Illumination. There exists different types of light sources and they differ in their spectrum. If you program an Algorithm you should consider these fact. There are two main groups:

- Indoor Illumination
- Outdoor Illumination

A mentionable Term is the Black Body Radiator. The spectrum of a Black Body Radiator depends only on temperature. Many lightsources are such Black Body Radiators. The Sun, for example, is a black Body Radiator a fluorescent lamps are not.

6.1. Trichromacy

Instead of using the full spectrum we try to describe this spectrum with a combination of three basic colors. So most algorithms in CV and CG operate in trichromatic space.
In other words, any light T can be described as:

$$T = w_1 P_1 + w_2 P_2 + w_3 P_3 \qquad \text{where} \qquad w_1, w_2, w_3 \geq 0 \tag{6.1}$$

Almost any perceived color can be expressed as a linear combination of three primary colors. The three Colors which are used as the primary colors have to fulfill the following conditions:

- independent
- span the space of perceived color

6.1.1. Grassmann's Law

The theory of mixing color is called Grassman's Law:

- Consider 2 colord lights T_a and T_b:

$$T_a = w_{a_1}P_1 + w_{a_2}P_2 + w_{a_3}P_3 \tag{6.2}$$

$$T_b = w_{b_1}P_1 + w_{b_2}P_2 + w_{b_3}P_3 \tag{6.3}$$

1. Law Mixing the lights = Mixing the wheights (matches)

$$T_a + T_b = (w_{a_1} + w_{b_1})P_1 + (w_{a_2} + w_{b_2})P_2 + (w_{a_3} + w_{b_3})P_3 \tag{6.4}$$

2. Law If two lights are matched by using the same weights (matches) then they must be the same:

$$w_{a_i} = w_{b_i}, \quad \forall i \Rightarrow T_a = T_b \tag{6.5}$$

3. Law Matching is linear:

$$kT_b = kw_{b_1}P_1 + kw_{b_2}P_2 + kw_{b_3}P_3 \tag{6.6}$$

Simplified Model for Pixel Response

There exist a simplified model for the Pixel response in a Camera:

$$p_k(\vec{x}) = g_d(\vec{x})d_k(\vec{x}) + g_s(\vec{x})s_k(\vec{x}) \tag{6.7}$$

where:

- d_k is the image value for the kth color filter of the diffuse reflection of an equivalent flat frontal surface viewed under the same light.
- g_d is a geometric term that captures the variation in brightness caused by changes in the surface orientation.
- s_k is the image value for the kth color filter of the specular reflection of an equivalent flat frontal surface viewed under the same light
- g_s is a geometric term that captures the variation in the amount of energy that is specularly reflected.

7. HoughTransform

All features before (like: edges, texture,color) are local. But Textons and color can be used for mre general scene analysis or Edges are typically convey geometry information.
The Houghtransform generalize from edge-pixels to more abstract geometry shapes (lines, circles,etc)

7.1. Template Matching

Is one Method (not the HT) to find geometry shapes. It works in that way that you create a template (mask, mini-Image) of an object(e.g. curve) that you are looking for. Convolve the image with that template, so that exact match give a high response. A problem is that you need to create an exemplary mask that is accurate and general enough. The advantage of that method is that the Idea generalizes to arbitrary shapes.

7.2. Hough Transform

The basic Idea of the HT is to measure a distinct characteristic/property(for a line: slope and intersect) of the shape. If a pixel has this characteristic/property, then it is highly probable that it belongs to the shape you are looking for.
The advantages of the HT are:

- Edges need not be connected
- The object (line,circle) may be only partially visible

HT for line Detection

The HT can be generalized, but to understand how its works is easier to look at core concept.

- Each white pixel in Figure 7.1, if considered in isolation, could lie on an infinit number of straight lines (e.g. yellow lines)
- Each pixel votes for every line it could be on
- the line(s) with the most votes win. In the case of 7.1 the line which is constructed by the white pixel gets #whitePixel votes in contrast to the yellow lines with one vote

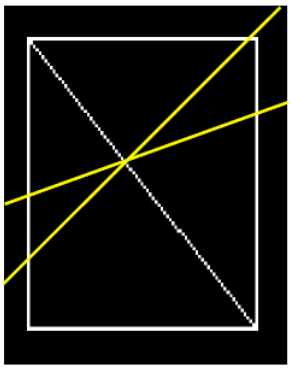
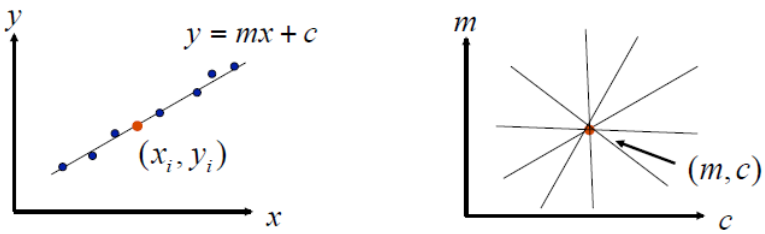


Abbildung 7.1.: HT - Line Detection

Step 1: Measure property

The equation of a line is: $y = mx + c$. A line is uniquely identified by the parameter pair (m, c). A line in the image space I is represented by a single point in the parameter space(respectively Hough Space) (m, c)-space (figure 7.2). A Point (x_i, y_i)



(a) Image Space (b) Image Space

Abbildung 7.2.: HT - Line Detection

can be seen as a point of:

$$\text{either line} \quad y_i = mx_i + c \qquad \text{[Image Space]} \tag{7.1}$$

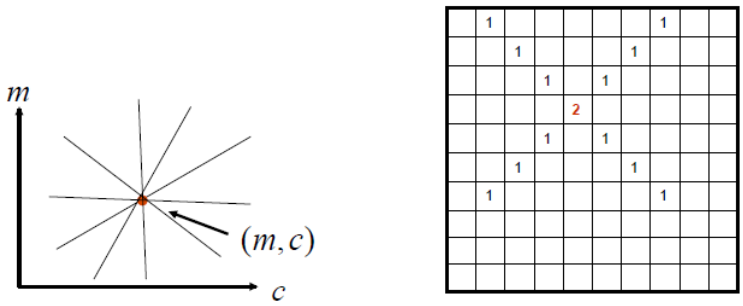
$$\text{or a line} \quad c = -x_i m + y_i \quad \text{[parameter Space]} \tag{7.2}$$

Step 2: Count

Create a Accumulator Array $A(m, c)$ and initialize it with Zero. Then for each image edgel (in figure 7.1 the white Points) compute for each m

$$c = -x_i m + y_i \tag{7.3}$$

and increment count in the Accumulator Array at $A(m, c) = A(m, c) + 1$. Take the local maxima in $A(m_{max}, c_{max})$ (the red two in 7.4) and you find your line $y = m_{max} \cdot x_i + c_{max}$.



(a) Image Space (b) Accumulator Array

Abbildung 7.3.: HT - Line Detection

Foot of the normal Representation

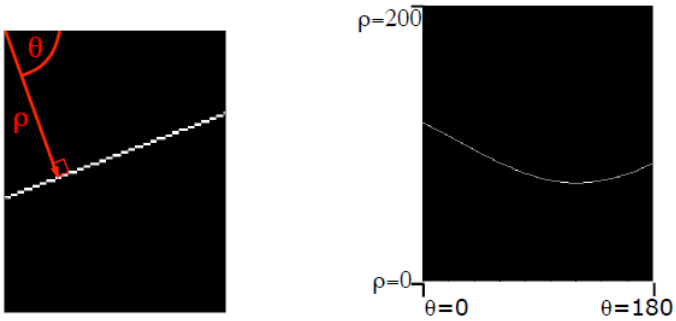
There are two problems in the Method above:

- 1. we have a infinite number of lines where a Point can lie on but we have a finite Accumulator Array
- 2. a vertical Line can not be represented with a Function.

The solution is to use Another representation of a line:

$$\rho = -x\cos\theta + y\sin\theta \quad \text{bounded parameter space:} \quad \begin{array}{c} 0 \leq \theta \leq 2\pi \\ 0 \leq \rho \leq \rho_{max} \end{array} \tag{7.4}$$

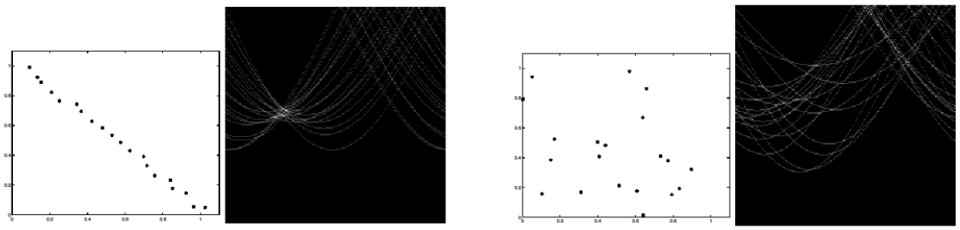
In 7.5 are examples for the HT on an Image with a noisy Line (left) and on an Image without any Line.



(a) Foot of the Normal

(b) appropriate parameter Space

Abbildung 7.4.: HT - Line Detection



(a) Noise Lines + Vote

(b) No Lines + Vote

Abbildung 7.5.: HT - Line Detection

HT - more then Lines

The HT can more detect than Lines. If you find a characteristic/property which is linked to the object you are looking for then you can use he HT to find it. But there are limitations cause of the Power of the Computer, to use more than 3 variable are not common. The calculation is to expensive.
A circle can be uniquely identified by the radius and the location of its center. For circle detection of known radius r, a point in image space become a circle in Hough Space ($A(a, b)$), if the radius unknown you get a 3D Hough space ($A(a, b, r)$. The Houghtransform could also use the orientation of the gradient to find the circles.

8. Deformable Contours

The Goal of deformable Contours is to find a contour that best approximates the perimeter of an object.
The main Idea is that the image information (usually edges) guide an elastic band that is sensitive to the intensity gradient (or other image features). The band is initially located near the image contour of interest and the band is deformed, pulled, by the edges (or other information) to fit the target contour.
The edge-based deformable contours explicitly use the intensity gradient of the image (not like HT which concentrate on existence of edge Points).

8.1. Procedure

- A contour (open or closed) is placed near the image contour of interest.
  - The initial placement can be done manually or be the output of some other algorithm.
  - ?Seeding? the snake (step 1) can be critical in the success of finding the contour
- During an iterative process, the active contour is attracted towards the target contour by various forces that control the shape and location of the snake within the image
- The active contour deformation ends either when it becomes relatively stable (stops to evolve), or after a fixed number of iterations.

8.2. Forces

The deformation of the contour will be done by an weighted sum of three Forces:

- Continuity term (force), E_{cont} which encourages continuity of the contour
- smoothness term (force), E_{curv} which encourages smoothness in the contour

- edge attraction term (force), E_{img} whichpulls the contour towards the closest image edge

The energy terms can be separated into *internal energy* (E_{cont}, E_{curv}) and *external energy* (E_{img})
The contour itself is given in parametric form:

$$c(s) = (x(s), y(s)) \tag{8.1}$$

where $x(s)$ and $y(s)$ are the coordinates along the contour and s is the arc length $s \in [0, 1]$ (figure 8.1) Often the contour is not continues its represented by N points on

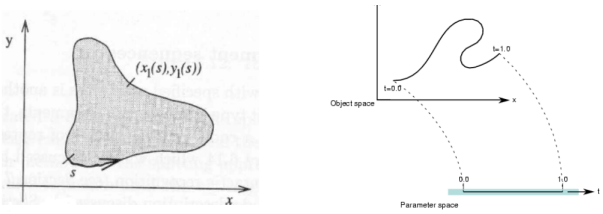


Abbildung 8.1.: HT - Line Detection

the contour. The contour is deformed by using an energy functional which measures the appropriateness of the contour:

$$E = \min \sum_{i=1}^{N} \alpha_i E_{cont} + \beta_i E_{curv} + \gamma_i E_{img} \tag{8.2}$$

where α, β, γ control the relative influence of the corresponding energy terms and can vary along $c(s)$(for each of the N points on the contour).
Good solutions correspond to minima of the functional!

Continuity Term

The continuity term E_{cont} encourages continuity of the contour. It is based on the first derivative. The Term is defined as:

$$E_{cont} = (\bar{d} - ||p_i - p_{i-1}||)^2 \quad \text{where} \quad \bar{d} = \frac{1}{N-1} \sum_{i=2}^{N} ||p_i - p_{i-1}|| \tag{8.3}$$

The term $\bar{d}$ is introduced that the Term try to move the Points p of the contours are compact and roughly the same distance between each other(without $\bar{d}$ the points would over represent the contour at some points and at other the points would miss). For a compact form we want to minimize the distance of the points between each other, so the Continuity Term needs to be minimum.

Smoothness Term

The smoothness Term E_{curv} encourages smoothness of the contour and is based on the second derivative. The Term is defined as:

$$E_{curv} = ||p_{i+1} - 2p_i + p_{i-1}||^2 \quad \text{where} \quad i = 2, 3, ..., N-1 \tag{8.4}$$

We want to avoid oscillations so we have to penalize high curvature which mean we have to minimize the smoothness Term.

Edge Attraction Term

The edge attraction term E_{img} attracts the contour towards an edge-defined target contour and is defined as:

$$E_{img} = -||\nabla I|| \tag{8.5}$$

where ∇I is the spatial gradient of the intensity image I, computed at each contour point. The minus is the consequence we want to minimize the Energy Function. The closer the contour is to the edge the smaller (negative) will be the Force.

8.3. Minimization

There are different minimization concept which can be used.

Greedy Algorithm (special for deformable contours)

- 1. Greedy minimization: move Point p_i within a small neighborhood to the point that minimize the energy function.
- Corner Elimination: look at the smoothness value of all $p_i's$, and set the weight of the Point with the highest Value for this Iteration to Zero. (let move the other points)
- Iterate until convergence

The Forces must be normalized (divide by the largest value in the neighborhood). There is no guarantee of convergence to the global minimum and the method is very dependent on the Initialization. The iteration until convergence is proportional to the number of Points on the contour.

9. Binocular Stereo

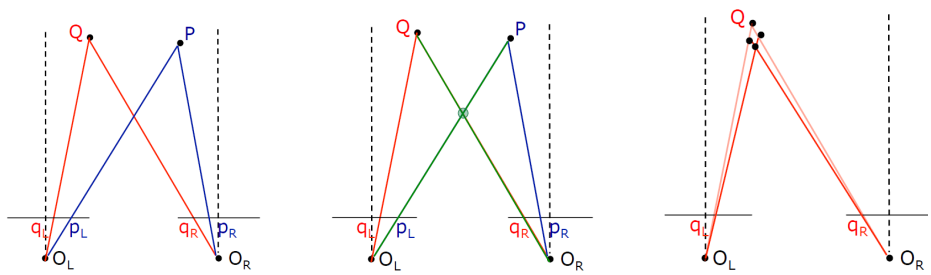
The Goal of Binocular Stereo is to infer information about the 3D structure and distances of a scene from two or more images taken from different viewpoints. There exists two major Problems:

- Correspondence problem
- Reconstruction

When the Correspondence Problem is solved we can compute th relative shift, the disparity, between the two projections. The disparity data is then converted to a 3D map. For this transformation we need some knowledge about the geometry of the stereo system.

9.1. Basic Binocular Stereo Setup

In 9.1(a) you can see a simple Binocular Stereo Setup. There exists two image planes (lie on the same plane) which optic axes are parallel to each other (dashed lines). The Point Q in the scene is represented in the left Image as the point q_l and in the right Image as the point q_r. Accordingly for the Point P.
When the correspondence is correct, which means q_l and q_r are matched together (same for p_r, p_l), then the intersection of the corresponding rays gives the 3D location of scene point that generated the projections (i.e. Q and P accordingly 9.1(a)). If the



(a) Correspondence and Triangulation

(b) Impact of a wrong Correspondence

(c) Noise and Correspondence

Abbildung 9.1.: HT - Line Detection

Correspondence is wrong, e.g. match p_l and q_r, the intersection of the corresponding

rays result in a wrong 3D location (figure 9.1(b)).
The noise in the image capture process introduces inaccuracies in the projection rays that directly affect the triangulation process (figure 9.1(c)).

Triangulation

To recover the 3D Information of a Point we can use the Setup we introduced above. Define the source our main coordinate system at the Center of Projection of the left Image(figure 9.2).
The depth can be calculated with (based on the Triangles in 9.2):

$$Z = f\frac{T}{x_L - x_R} = f\frac{T}{d} \tag{9.1}$$

The distance between the Center of Projections are a main variable in this trian-

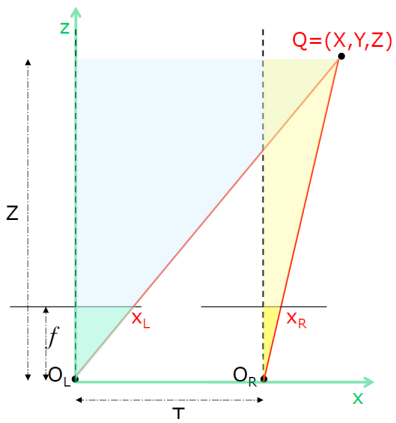


Abbildung 9.2.: The triangulation

gulation equation.Due to this the distance T have a huge impact.
If you use a small Baseline (figure 9.3(left)) you get a large depth error because all

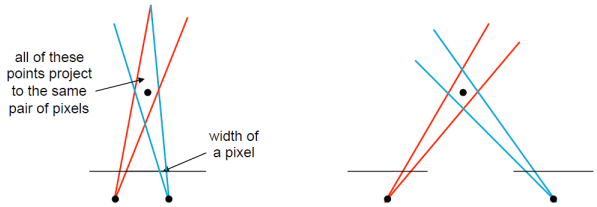


Abbildung 9.3.: Small Baseline (left); Large Baseline(right)

Points of that surrounded Area are projected to the same pair of Pixels-
If you use a large Baseline (figure 9.3(right)) you get a much better depth-value but the scene that both cameras can see gets smaller and another big problem is that

the cameras see a point from a different perspective which could lead to the problem that the same Point appears differently in the Images (due to reflectance, etc.).
A Solution is Vergence ! This increases the field of View and increases the accuracy in the correspondence.

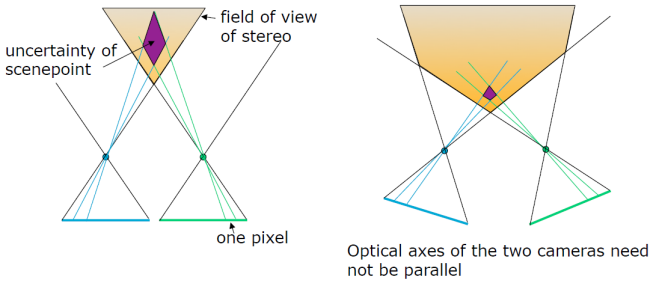


Abbildung 9.4.: Vergence

Stereo Image Rectification

So far we have assumed that we have parallel optic axes and a scan-line coherence. But such a setup can lead to inaccuracies.
To have a more usable setup we allow the Image plane to violate these assumptions(figure 9.5). As a preprocessing step we project the real image planes (the grey ones) onto virtual image planes (the yellow ones). Then we can use all the math from above for these virtual rectified (virtual) image planes.

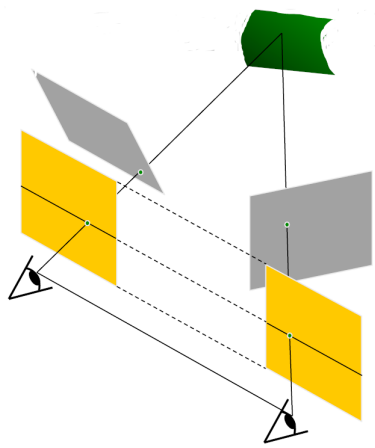


Abbildung 9.5.: Rectification

10. Structured Light

Structured Light is also a method to reconstruct 3D-Information. But in contrast to the passive stereo concept, structured Light is a active method.

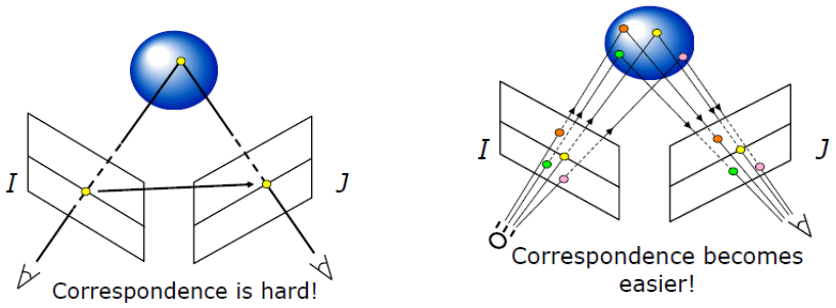
10.1. Parenthesis: active vs. passive

- Passive (stereo, motion)
  - easy data collection (just take pictures)
  - non-instrusive setup
  - can produce dense depth maps
  - may not work for featureless surfaces
- Active (range scanning, ToF, structured light)
  - more robust correspondence
  - can recover data even at featureless parts of the scene
  - higher accuracy but possibly sparser deph maps
  - more complex hardware
  - instrusive (active illumination may alter scene appearence)
  - limited range of deph

Both passive and active methods follow the same underlying principle of ray triangulation.

10.2. Basic concept

The triangulation idea can be applied in a setup that uses a projector (or laser beam) and a camera, instead of 2 cameras. The ray og the controlled incident light replaces the projectionen of the second Camera.
The Object surfaces are illuminated with a known pattern if light, this structured light is the main source of the illumination. Depending on the shape of the object the pattern is distorted, this distortion of the known pattern is captured by the camera.
Prior knowledge that is used: known geometry of light pattern and known relative position of light an camera.
To solve the correspondence Problem with this light pattern is much easier (figure 10.2) The pattern can be a single laser stripe or a very complex stripe pattern with



(a) Correspondence with passiv (stereo) Method

(b) Correspondence with active method

different color in it. The advantage of the complex colored pattern is that you need very few images (one or two). But you need a more complex correspondence algorithm.

11. Multiview Geometry

Still we have not solved the correspondence Problem. To solve that Problem we introduce Epipolar Geometry.

11.1. Epipolar Plane

The Epipolar Plane (Γ) is defined by the two Center of Projections O and O' and a point in the scene P(figure 11.1).

The Epipolar Line $(OP, O'P)$ is the intersection of the epipolar plane with the

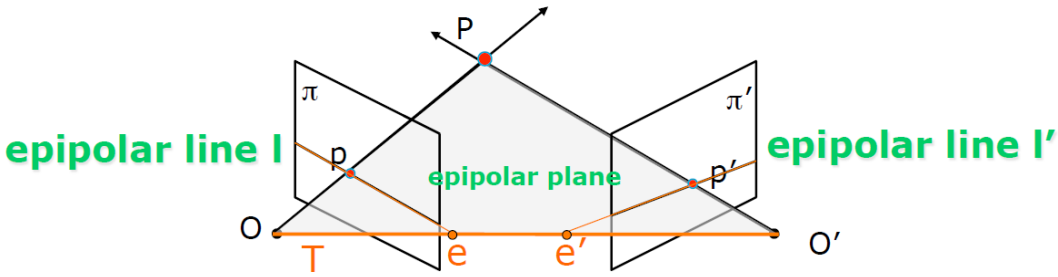


Abbildung 11.1.: The triangulation

image plane.

The epipole (e, e') is the intersection of the baseline with the respective image plan.

epipolar constraint:

If both p and p' are projections of the same point P, then p and p' *must* lie on the same epipolar plane. They must lie on epipolar lines l and l' respectively.

The impact of this constraint has a fundemental role in stereo and motion analysis. It reduces the correspondence problem to a 1D search along *conjungate epipolar lines*. Given an image point p, one needs to only search in the epipolar line l' for the corresponding point p'.

Required Knowledge

In order to know the epipolar geometry, we need:

- The location of the two Center of Projections
- the location of the two image planes
- The orientation of the image planes
- Intrinsic camera characteristics
  - Pixel size
  - Focal length
  - Principal point
- Extrinsic camera characteristics
  - The relative position of the 2 optical centers
  - The relative orientation of the two image planes

Buil up the Math

To develop a mathematical form of the epipolar constraint we assume that the intrinsic parameters of each camera are known. the goal is to express algebraically the epipolar constraint, so that it can be incorporated in our correspondence, stereo and motion algorithms.

1. Epipolar Plane Constraint

The vectors $\vec{Op}, \vec{O'p'}$ and $\vec{O'O}$ are all co.planar, this means they must satisfy the following equation:

$$\vec{Op} \cdot (\vec{O'O} \times \vec{O'p'}) = 0 \tag{11.1}$$

2. Relating the two Camera coordinate systems

Each image is unaware of the other camera, we need to express everything in terms of a single coordinate system. Without loss of generality we choose as the reference coordinate system the one of the camera with COP O.
The baseline T shows you how the COP O' can move to COP O (figure 11.2):

$$\vec{t} = \vec{O'O} \tag{11.2}$$

After applying the translation t to every point p' of the camera with COP O' the coordinate have the same origin (dashed orange rectangle π'). After the translation

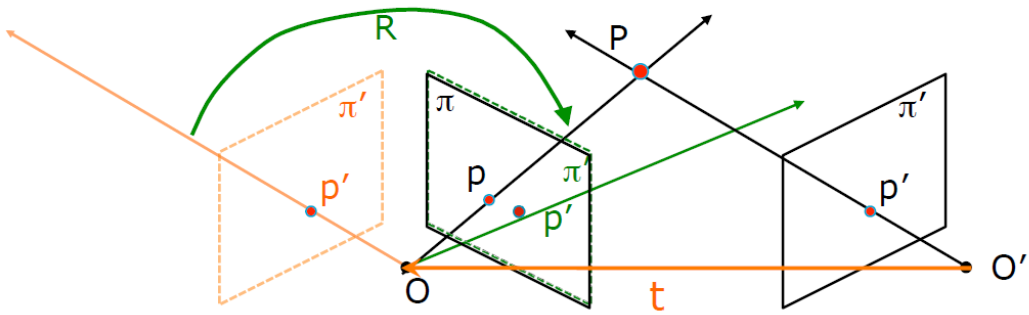


Abbildung 11.2.: Way to one coordinate System

the coordinate systems are not totally matched, we need to rotate the system of the camera with COP O' (green arrow in 11.2). We need a rotation Matrix R to align the two coordinate systems (dashed green rectangle π').
We can now rewrite the epipolar plane constraint in the coordinate system of camera O:

$$\vec{Op} \cdot (\vec{O'O} \times \vec{O'p'}) = 0 \Rightarrow \vec{p} \cdot (\vec{t} \times (\vec{Rp'})) = 0 \tag{11.3}$$

rewritten as a series of matrix multiplications:

$$\mathbf{p}^T(\mathbf{t} \times \mathbf{R})\mathbf{p'} = 0 \tag{11.4}$$

This will be represented more compactly as:

$$\mathbf{p}^T\mathbf{E}\mathbf{p'} = 0 \quad \text{where } \mathbf{E} = [\mathbf{t}_x]\mathbf{R} \quad \text{with } [\mathbf{t}_x] = \begin{Bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{Bmatrix} \tag{11.5}$$

where $\mathbf{E}$ is called the *essential matrix*(t_x is the matrix representation of the cross product with t).
The equation $\mathbf{p}^T\mathbf{Ep'} = 0$ is the algebraic representation of the epipolar constraint.
For the uncalibrated case, the matrices (rotation and translation) that express point p' in term of the coordinate system of camera O must also incorporate the intrinsic camera parameters:

$$\mathbf{p}^T\mathbf{K}^{-T}\mathbf{EK'}^{-1}\mathbf{p'} = 0 \quad \text{or} \quad \mathbf{p}^T\mathbf{Fp'} = 0 \tag{11.6}$$

where $\mathbf{F} = \mathbf{K}^{-T}\mathbf{EK'}^{-1}$ and K and K' are the intrinsic parameter matrices of cameras O and O'. $\mathbf{F}$ is called the fundamental matrix.

11.1.1. Key Points of Epipolar Geometry

For each pair of corresponding points p and p' in camera coordinates, the following relationship holds:

$$\mathbf{p}^T\mathbf{Ep'} = 0$$

$\mathbf{E}$ is the essential matrix

For each pair of corresponding points q and q' in pixel (image) coordinates the following relationship holds:

$$\mathbf{p}^T\mathbf{Fp'} = 0$$

$\mathbf{F}$ is the fundamental matrix

11.1.2. Estimation of E (or F)

Recover $\mathbf{E}$ (or $\mathbf{F}$) once, keep the camera setup stable and then reuse it for every scene point.
To estimate the matrices you can basically use SVD, to take into account that there exists noise, numerical errors, etc use the Longuet-Higgins Eight-Point Algorithm to estimate $\mathbf{E}$ or $\mathbf{F}$.

- you need 8 correspondece Points
- build up a linear equation system
- apply SVD
- Column of the singular value 0 is the solution for F, but cause of noise, numerical and other influences there exists in the most cases no 0 singular value

- apply SVD on $F = U_F D_F V_F^T$ and creating $D' = D_F$ with the smallest singular value of D_F replced by 0.
- get new estimate of F: $F' = U_F D' V_F^T$

12. Motion

In Computer Vision the Term motion is used to refer to images taken over time, the two main goals in the Topic of motion analysis are:

- Detect which objects are moving and in which direction
- extract shape information if possible

Motion analysis typically involves motion detection, the detection and localization (tracking) of moving-objects and derivation of 3d object properties.
An image sequence is a series of N images acquired at discrete time instants $t_k = t_0 + (k\delta t)$, where δt is a fixed time interval and $k = 0, 1, ..., N-1$. δt is very small as a consequence the apparent displacement between frames is at mist a few pixels, this observation simplifies the correspondence problem.
A very simple an basic Idea to detect motion is to subtract consecutive images from the frames. If there is a difference, then there is motion. But there are some effects that you should keep in mind.
You cannot distinguish between the object moved to the right or the camera moved to the left. Another effect is that you cannot detect the motion from a spinning sphere which have uniform color (figure 12.1 (a)). You can also misinterpret motion when the illumination change and you assign the movement to the object (figure 12.1). Another big Problem is the Aperture Problem (figure 12.2).

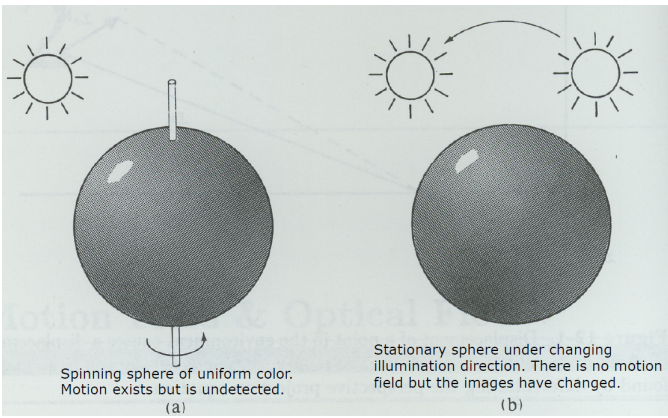


Abbildung 12.1.: Effects which lead to wrong conclusions by only subtract Images

The line, gray in the first Image, moves downright, black in the second image. If you

have the full field of view i can detect the motion of the line correctly: a movement to the bottom right. But if your camera only can see the inner circle of the blue donut then you would detect the motion if line wrong: a movement to the upper right.

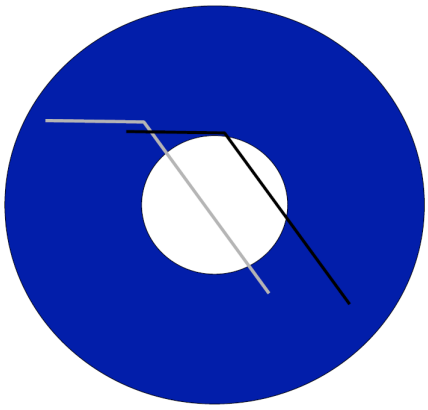


Abbildung 12.2.: Aperture Problem

Three important Points on Motion:

- Basically you can describe Motion as an observed change in intensity values at pixel p
- The change we associate with motion
- We try to infer which motion in 3D caused this motion in 2D

12.0.3. Optical Flow vs. Motion Field

Optical Flow

Optical flow is purely based on image the intensity and is a set of 2d vectors , each 2d vector shows you how the direction of motion of the intensity value is. Its purely measuring how the color respectively gray patterns has change in the image plane(figure 12.3).

Abbildung 12.3.: Optical Flow

Motion Field

The projection of the motion of the points in the scene. It is a collection of 2D vectors, each vector being the projection of the 3D velocity of a scene point on the image plane (figure 12.4).

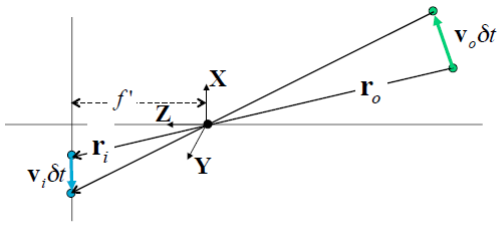


Abbildung 12.4.: Projection of the motion Vector on the image plane

Optical Flow $\neq$ Motion Field

The difference of these two key Words can be shown at the Barber's Pole Illusion (figure 12.5).
When the barbers pole rotate it looks like that the stripes goes up. But in reality each point on the barbers pole is just rotating clockwise, but not from the bottom up.
The motion Field shows the true motion in the 3D-World and the projection of these 3D motion is the motion Field. The optical flow is what you, the motion on your Image plane. Most of the time we have not a illusion like that and the optical flow

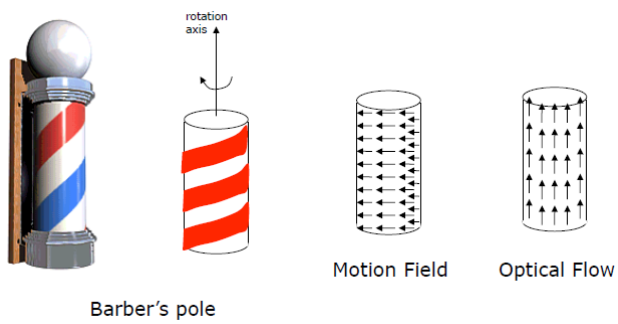


Abbildung 12.5.: The Barber's Pole Illusion

is a good approximation of the motion field.

Motion on a straight Line can be described as the distance traveled per unit time:

$$\vec{v} = \frac{ds}{dt} = \left(\frac{dx}{dt}, \frac{dy}{dt}\right) \tag{12.1}$$

If the point is moving on a circle then the best way to describe its speed, is by how many degrees it travels per unit time, i.e. its angular velocity:

$$\vec{v} = \frac{d\vartheta}{dt} \tag{12.2}$$

Motion Field Basics

Let $P = (X, Y, Z)$ point in scene and $p = (x, y, f)$ its projection: $\vec{p} = \vec{P}(\frac{f}{Z})$ The relative motion between the point P and the camera can be described as:

$$\vec{V} = -\vec{T} - \vec{\omega} \times \vec{P} \tag{12.3}$$

where $\vec{T}$ is the pure translation part of the motion and $\vec{\omega}$ is the angular velocity, where $\vec{V}$ is the vector in 3D, the projection give $\vec{v}$ which is the vector for the motion field.
Note that the rotational component of the motion field does not convey any information about depth (without translation you cannot use motion to reconstruct depth). In the case of pure Translation you can derive formals which tell you that that the length of $v(p)$ is proportional to the distance between p and $p0$ and inversely proportional to the depth of the 3D point P. The motion field of pure translation when there is a change in depth is radial, i.e. all vectors emanate/radiate from a common origin named the vanishing point of the translation direction.

Optical Flow Estimation

We compute the optical flow and hope this a good approximation of the motion field (being fully aware that the accuracy depends exactly on the direction of motion with respect to the patterns in the scene (barbers poll)).
There are two main strategies for computing the optical flow:

- Differential Methods: motion is computed at ever pixel (based on time derivatives)
- matching/prediction methods: motion is estimated only on selected features(make prediction about possible positions in the next frame)

12.0.4. Differential Methods

We need three assumptions for these methods:

- The image brightness is continuous and differentiable
- The image brightness value (more properly the image irradiance E) of objects doesn't change over ?t, in other words $\frac{dE}{dt} = 0$. This assumption is called the *image brightness constancy assumption*
- Points do not move very far

which leads to the Image Brightness Constancy Equation:

$$\vec{G}^T \vec{v} + E_t = 0 \tag{12.4}$$

where $\vec{G}$ (is the gradient(edge-detection)) and E (the pixel value) can directly extract from the Image.
When you have a sequence where the relative movement is very fast (which violates the third assumption) you can use the Gaussian pyramid to restore the small motion assumption.

13. Kalman Filter

The Kalmanm Filter is a prediction Methods for optical flow.
The image brightness equation does not explicitly incorporate previous knowledge. Such a method would work better:

- if we observe the scene for more than 2 or 3 frames
- there specific objects or regions whose motion is analyzed instead of estimating the motion of every pixel that has changed.

13.1. Dynamic System

Motion is now analyzed in the context of a dynamic system. Attributes for such a system are:

- We are dealing with a system that is changing over time
- we have sensors observing the dynamic scene. The measurements of compute from them are noisy
- there is an uncertainty about how the system is changing. In other words we have an uncertain model of the system's dynamic
- we want to produce the best possible estimates of what is moving in which direction and at what speed. We want optimal estimates of the state of a dynamic system

optimal estimates means that the methodology tries to minimize the error between what we estimates will be happening in the next time instance versus what is really happening in the next time instance.

The Kalman Filter was designed as an *optimal Bayesian technique* to estimate state variables at time t based on:

- the previous state of the dynamic system, i.e. at time $t-1$
- indirect and noisy measurements at time t

- known statistical correlations between variables and time

Consider motion as a problem where we have to estimate the values of the variables of some dynamic system.

Motion and Kalman Filter

A dynamic system is often described via:

- a *state vector*$\vec{x}$
- a set of equations called the *system model*, which captures the evolution of the state vectors over time

State Vector $\vec{x}$

The state vectors describes what is the current status of the dynamic system. A very important aspect is to choose what your state vector should be. For example you could be interested in the speed of a Tennis ball, then your state vector should be the velocity of the ball. But if you are interested in the position of the ball to measure if the ball is still in the court, your state vector should be the position of the center of the Ball. It depends in what you are interested in.
Mathematically the state vector is described as a set of variables at some time instance t:

$$\vec{x}(t) = (q_1(t), q_2(t), ..., q_n(t)) \tag{13.1}$$

where q are the variables of the dynamic system. In case of motion it look like:

$$\text{2D: } \vec{x}(t) = (v_x(t), v_y(t)) \qquad \text{3D: } \vec{x}(t) = (v_x(t), v_y(t), v_z(t)) \tag{13.2}$$

There is no limitation on the dimension of the state vector, for example you can track the velocity of four different objects:

$$\vec{x}(t) = (\vec{x}_0(t), \vec{x}_1(t), \vec{x}_2(t), \vec{x}_3(t)) \tag{13.3}$$

System Model

The key Assumption is that the system is **linear**. That means that the relationship between consecutive state-changes is linear!
The system model can be written as:

$$\vec{x}_k = \Phi_{k-1}\vec{x}_{k-1} + \vec{w}_{k-1} \tag{13.4}$$

- $\vec{w}_{k-1}$ is a vector describing the random process noise. This noise does not describe the noise from the sensors ! This noise describes unknown effects that occurs during the transition from state $k-1$ to state k. For example if you want to track a tennis ball, instead of describing wind as an influence in the state vector you can interpret wind as a kind of noise. Which is a very important aspect of the Kalman Filter that you can describe circumstances from the environment as noise in the model.
- Φ_{k-1} is the state transition matrix that captures the relationship between the current state k and the previous state $k-1$ in the absence of noise.
- Φ_{k-1} is an $n \times n$ matrix, $\vec{w}_{k-1}$ is an n-dimensional vector

Measurements

At any time t_k we have a vector $\vec{z}_k \in R^m$ of measurements of the system. Also the sensors are noise, the vector $\vec{\mu}_k$ describes the uncertainty of each measurement $\vec{z}_k$. The relationship between the true system state $\vec{x}_k$ and the measurements is:

$$\vec{z}_k = \mathbf{H}_k \vec{x}_k + \vec{\mu}_k \tag{13.5}$$

where $\mathbf{H}_k$ is the measurement matrix that captures the relationship between our measurements and the real system variables in the absence of noise, $\vec{\mu}$ measure the noise.

13.1.0.1. Noise

There are two types of noise:

- Process noise $\vec{w}_k$
- Measurement noise $\vec{\mu}_k$

Both types of noise are assumed to be white, zero-mean Gaussian !

13.1.0.2. Kalman Filter Setup (summary)

- We are observing a dynamic system
- We have a linear system model, but there is uncertainty about the accuracy of the employed model

$$\vec{x}_k = \Phi_{k-1} \vec{x}_{k-1} + \vec{w}_{k-1} \tag{13.6}$$

- We have noisy sensors that measure how the dynamic system behaves

$$\vec{z}_k = \mathbf{H}_k \vec{x}_k + \vec{\mu}_k \tag{13.7}$$

- the noise (Process noise and sensor noise) is assumed to follow a white, zero mean, Gaussian distribution

13.1.1. KF Steps

An estimate of $\hat{\vec{x}}_k$ is obtained from $\hat{\vec{x}}_{k-1}$ and $\vec{z}_k$ in a 2-step process

- 1. *Prediction Step:* First, obtain an intermediate estimate, $\hat{\vec{x}}_k^-$, based on the previous estimates, but without using the newest measurements $\vec{z}_k$

$$\hat{\vec{x}}_k^- = \Phi_{k-1} \hat{\vec{x}}_{k-1} \tag{13.8}$$

- 2. *Update Step:* Use the intermediate estimate $\hat{\vec{x}}_k^-$ and combine it with the newest measurements $\vec{z}_k$, to get $\vec{x}_k$ (figure 13.1)

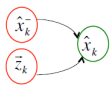


Abbildung 13.1.: Combine $\hat{\vec{x}}_k^-$ and $\vec{z}_k$

This 2-step process is performed as a series of 4 or 5 recursive equations which are characterized by:

- 1. The *state covariance Matrix* $\mathbf{P}_k$ is the covariance matrix of the estimate $\hat{\vec{x}}_k$. It is also known as the *covariance of the estimates*, it's a measurement of the uncertainty in $\hat{\vec{x}}_k$
- 2. The *state covariance Matrix* $\mathbf{P}_k^-$ is the covariance matrix of the estimate $\hat{\vec{x}}_k^-$. It is also known as the *covariance of the prediction error*, it's a measurement of the uncertainty in $\hat{\vec{x}}_k^-$.
- the *gain matrix* $\mathbf{K}_k$ which expresses the relative importance of the prediction $\hat{\vec{x}}_k^-$ and the measurement $\vec{z}_k$.

The gain matrix is useful to emphasize either the model or the measurements. For example if you know that you have a noisy sensor you should trust more your model.

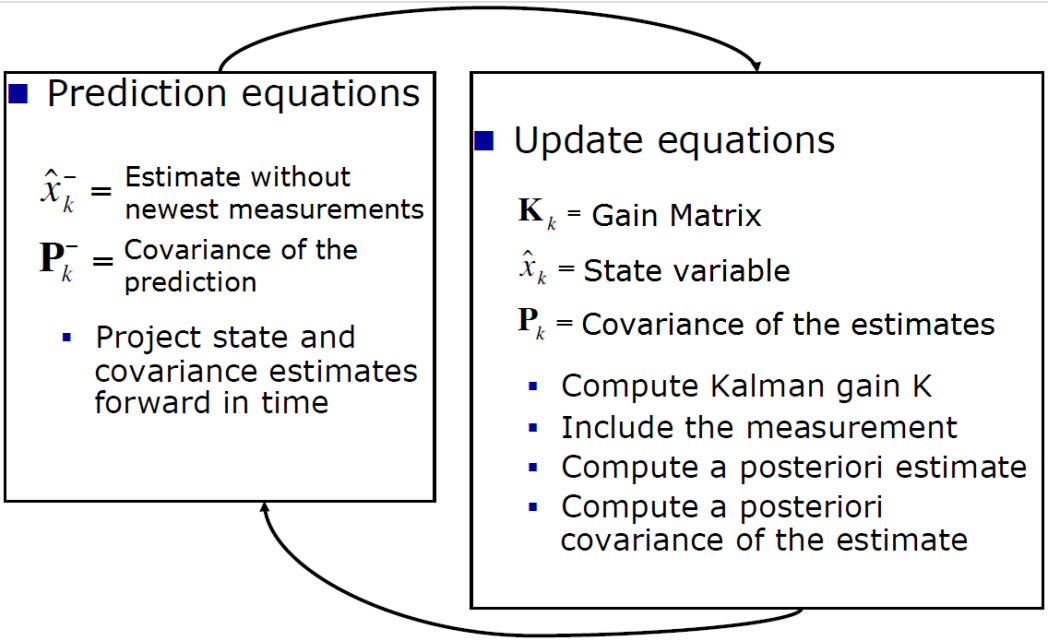


Abbildung 13.2.: Steps of the Kalman Filter

Optimality

A Kalman filter computes the optimal state estimate, as the maximum probability density of given the past estimates, the past measurements and the current measurement.

$$\hat{\vec{x}}_k = \max_{\hat{\vec{x}}_k} p(\vec{x}_k | \vec{x}_1, \vec{x}_2, ..., \vec{x}_{k-1}, \vec{z}_1, \vec{z}_2, ..., \vec{z}_{k-1}, \vec{z}_k) \tag{13.9}$$

From the assumption Kalman does at the beginning this probability density function is a Gaussian, so its maximum Value coincides with its mean:

$$\hat{\vec{x}}_k = \max_{\hat{\vec{x}}_k} p(\vec{x}_k | \vec{x}_1, \vec{x}_2, ..., \vec{x}_{k-1}, \vec{z}_1, \vec{z}_2, ..., \vec{z}_{k-1}, \vec{z}_k) \sim \mathcal{N}(\vec{x}_k, \mathbf{P}_k) \tag{13.10}$$

The optimal state estimate $\hat{\vec{x}}$ is the maximum of the PDF above, so it is the mean. The true state lies within an ellipse centered an $\hat{\vec{x}}_k$ where the axes of the ellipse are the eigenvectors of $\mathbf{P}_k$. In tracking features we use the uncertainty ellipses to reduce the search space for locating a feature in the next frame.

13.2. Conclusion

The Kalman Filter gives for linear system under white zero-mean Gaussian noise an optimal solution. Many systems exhibit Gaussian noise, it's a widely-used assumption. But most robotic systems and human motion are non-linear.

14. Particle Filter

Setup: Similar to Kalman Filtering, it explicitly *predicts an estimate* of the state of the dynamic system based on an *uncertain system model*. The prediction is then *updated* through the incorporation of information *by noisy measurements*.

Differences

The particle filter differ in the assumption that the Kalman filter does. The assumptions from Kalman that the system model is linear and the noise is a zero mean Gaussian are very restrictive.
The particle filter does not such assumptions ! The system model can be every thing, same counts for the noise!

14.1. Setup Particle Filters

- We are observing a dynamic system
- We have a system model, where f_k is a possibly non-linear transition function and $\vec{v}_{k-1}$ is a vector describing the random process noise. State transition formula:
$$\vec{x}_k = f_k(\vec{x}_{k-1}, v_{k-1}) \tag{14.1}$$
- We have noisy sensors that measure how the dynamic system behaves. Where h_k is a possibly non-linear measurement function and $\vec{n}_k$ is a vector describing the measurement noise.
$$\vec{z}_k = h_k(\vec{x}_k, \vec{n}_k) \tag{14.2}$$

There are no restrictions or assumptions regarding the dynamic system and the measurement model, other than they are Markovian.

PF Bayesian Framework

Goal: Estimate $\vec{x}_k$ given the previous states $(\vec{x}_1, ..., \vec{x}_{k-1})$ and the measurements up to time t_k, $(\vec{z}_1, ..., \vec{z}_{k-1}, \vec{z}_k)$.
In the Bayesian perspective: i want to compute how probable is it to have a particular value $\vec{x}_k$ occurring at time t_k given all this previous history $(\vec{z}_1, ..., \vec{z}_{k-1}, \vec{z}_k)$.
This means compute the pdf:

$$p(\vec{x}_k|\vec{z}_1, ..., \vec{z}_k) \tag{14.3}$$

and pick the value of $\vec{x}_k$ with the highest probability.

Markovian Assumption

The assumption is that only the state respectively the measurement before is relevant:

$$p(\vec{x}_k|\vec{x}_1, ...\vec{x}_{k-1}) = p(\vec{x}_k|\vec{x}_{k-1}) \tag{14.4}$$

$$p(\vec{z}_k|\vec{x}_1, ...\vec{x}_k, \vec{z}_1, ...\vec{z}_{k-1}) = p(\vec{z}_k|\vec{x}_k) \tag{14.5}$$

PF - 2-step estimation Process

This lead us to the 2-step estimation process:

- Prediction Step:

$$p(\vec{x}_k|\vec{z}_1, ...\vec{z}_{k-1}) = \int p(\vec{x}_k|\vec{x}_{k-1})p(\vec{x}_{k-1}|\vec{z}_1, ...\vec{z}_{k-1})dx_{k-1} \tag{14.6}$$

- Update Step(once the measurement $\vec{z}_k$ is obtained):

$$p(\vec{x}_k|\vec{z}_1, ...\vec{z}_k) = \frac{p(\vec{z}_k|\vec{x}_k)p(\vec{x}_k|\vec{z}_1, ...\vec{z}_{k-1})}{p(\vec{z}_k|\vec{z}_1, ...\vec{z}_{k-1})} \tag{14.7}$$

So if you know the pdf $p(\vec{x}_k|\vec{z}_1, ...\vec{z}_k)$, then estimation $\vec{x}_k$ is straightforward. Just pick the max.
But this pdf can be highly complex and we may have no analytic description (In KF this problem does not occur cause of the Gaussian assumption).

14.2. PDF Estimation and Importance Sampling

The PDF may highly complex and we may have no analytic description. To come up with this Problem the idea is to approximate $p()$ by generating N random samples from $p()$:

$$\hat{x}^{(i)}, \quad 1 \leq i \leq N \tag{14.8}$$

and then use these discrete samples when computing properties of $p()$, like the expected value or its variance. This samples are called **particles**!.
If you would know $p()$ you could sample $p()$ correctly, but you want to sample $p()$ without knowing $p()$ that you get an estimate of $p()$. This Problem is solved with the method importance sampling.
The Basic idea is that you use a helper function, which is called *important distribution* $q()$. So take N random samples from $q()$ instead of $p()$:

$$\hat{x}^{(i)}, \quad 1 \leq i \leq N \qquad \text{drawn from } q() \tag{14.9}$$

This technique of using an *importance distribution* like $q()$, to obtain samples from another distribution like $p()$, is called *importance sampling*.
Since $q()$ is of course not $p()$, each sample must be corrected by being multiplied by an appropriate weight, so that one can obtain an unbiased estimation of the properties of $p()$. For each sample $\hat{x}^{(i)}$ there is a weight $\tilde{w}^{(i)}$ that handles the discrepancy between the two distributions. This weights will be normalize such that all weights sum up to 1, normalized weights: $w^{(i)}$. In figure 14.1 you can see an example, the green function g is the importance distribution q and the red function f is the unknown function p. The blue stripes at the bottom are the normalized weights. You can see that the density of the samples follows the function g but the values of the weight approximate the unknown function f.

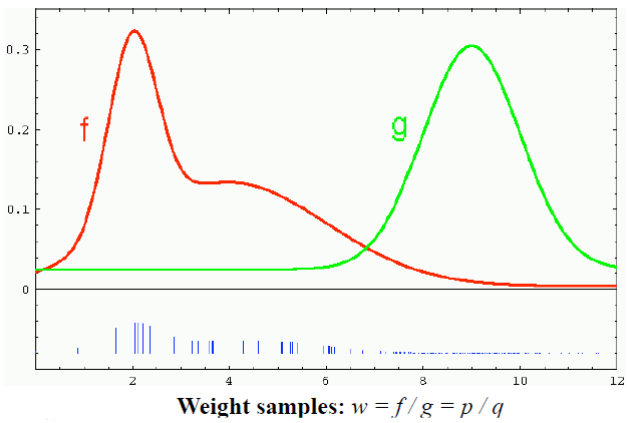


Abbildung 14.1.: importance Sampling

Basic Particle filter algorithm

- for each of the N particles do:
  - draw $\vec{x}_k^{(i)}$ from $q(\vec{x}_k^{(i)}|\vec{x}_{k-1}^{(i)}, \vec{z}_k)$
  - compute
$$\tilde{w}_k^{(i)} = \tilde{w}_{k-1}^{(i)} \cdot \frac{p(\vec{z}_k|\vec{x}_k^{(i)})p(\vec{x}_k^{(i)}|\vec{x}_{k-1}^{(i)})}{q(\vec{x}_k^{(i)}|\vec{x}_{k-1}^{(i)}, \vec{z}_k)} \tag{14.10}$$
- end
- For each particle $\vec{x}_k^{(i)}$, compute the normalized weight, $w^{(i)}$
- out of all particles, pick the $\vec{x}_k^{(i)}$ with the maximum weight, or set $\vec{x}_k$ to the expected value of posterior density $p()$.

Starving Samples - Resampling

There exists a point where the samples are starving and cant adapt to changes in the dynamic system. To avoid this a weight which gets a very high value will be spitted up in n numbers of new samples where the sum of these samples represents the old weight (figure 14.2). Starving samples (with very small values) will be deleted. This method is called resmapling and make sure that our particle filter can adapt to the dynamic system and can catch the object again if its lost.

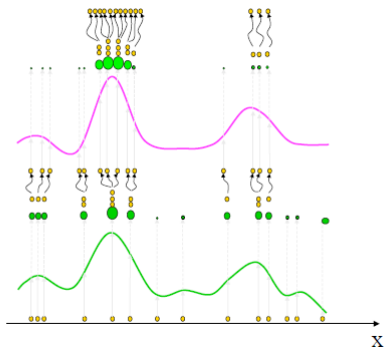


Abbildung 14.2.: Resampling Example

14.3. Advantages of Particle Filters

- Ability to represent arbitrary densities, not just Gaussians. This is particularly important for multimodal distributions

- Adaptive focusing on probable regions of statespace
- No Gaussian noise assumptions.
- General state and measurement models (no linear assumption)
- The framework allows the inclusion of multiple models. For example, simultaneously tracking multiple pedestrians, cars and bicycles

Particle Filter vs. Kalman Filter

- Same Bayesian framework and Markovian assumption
- Recursive formulation composed of a prediction and update step
- The Kalman Filter is fast, but is optimal only for linear systems with Gaussian distributions
- Particle Filters are slow, but place no limitations on the system model or the distributions

15. Remarks from the Exercises

15.1. Gaussian and Kernel size

A rule of thumb is to use the Kernel size for determining σ:

$$\sigma = \frac{w}{5} \quad \text{where w := size of Kernel H} \tag{15.1}$$

15.2. Perspective Projection

The Pinhole camera model implies the perspective Projection. A Point $P(X, Y, Z)$ in the world is projected as a point $p(x, y)$ on the Image plane. The perspective Projection results in a magnification factor m:

$$x = \frac{f}{Z} \cdot X \quad y = \frac{f}{Z} \cdot Y \qquad m = -\frac{f}{Z} \tag{15.2}$$

the equation shows us, that Objects near to the camera are shown bigger and objects far away from the camera are very small. The magnification m is inverse proportional to the distance.
A possible Solution is to assume $f \to \infty$, then the denominator gets negligible. This is the orthogonal projection !

15.3. Convolution

An 2D-Filter Kernel can be spitted into two 1D-Kernels. This is called separable Convolution. This brings advantages in computation time:

- 2D-Kernel $\sigma = 2$ with $(n \times n)$ need $n \times n$ multiplications for each Pixel $\Rightarrow$ $\mathcal{O}(n^2)$
- two 1D-Kernel $\sigma_x = \sqrt{2} \quad \sigma_y = \sqrt{2}$ with (n) needs n multiplication for x-direction and n multiplication for y-direction for each Pixel $\Rightarrow \mathcal{O}(n + n) = \mathcal{O}(2n) = \mathcal{O}(n)$

15.4. Corner Detection

A corner accents that there are significant changes in all directions in contrast to edges where you have no changes along the edge direction.
Corner detection is

- Invariant to intensity shifts. We look at difference/gradients
- Insensitive to intensity scaling. We look at differences/gradients → tendency persists.
- Rotation invariant. eigenvectors change, eigenvalues don't
- But not invariant to image scaling. A curved structure (edges) might become a corner.

15.5. Gabor Filters

Are used for edge detection. They are similar to the human visual system.
By combining the Gaussian kernel with sinusoidal functions, we get the Gabor filters

15.6. Biased Hough Transform

There is an bias in after the transformation into the have space caused by the finite size of an Image. In average, a line close to the image center are longer than lines far away from it. This leads to the problem that lines not in the Image center have lower values in the accumulator array.
A Possible solution either to use a wighted accumulation (less weight for center pixels) or use adaptable threshold when detecting the maxima (higher closer to center, lower in border region)

15.7. Hough Transform and tracking

If you want to track with the HT a circle its computational expensive due to 3-/4-D Hough space. This leads to problems in tracking:

- Stability (e.g. maxima less prominent, b/c less pixel vote for it)
- accuracy (depending on discretization):

- A high accuracy leads to a large hough space which leads to a computationally expensive search of the maxima
- A low accuracy leads to small hough space where you can find much faster the maxima, but you loose accuracy

A possible solution is to use alternative detection methods like: Fast Circle Detection (figure 15.2) another Method would be a size invariant circle detection.

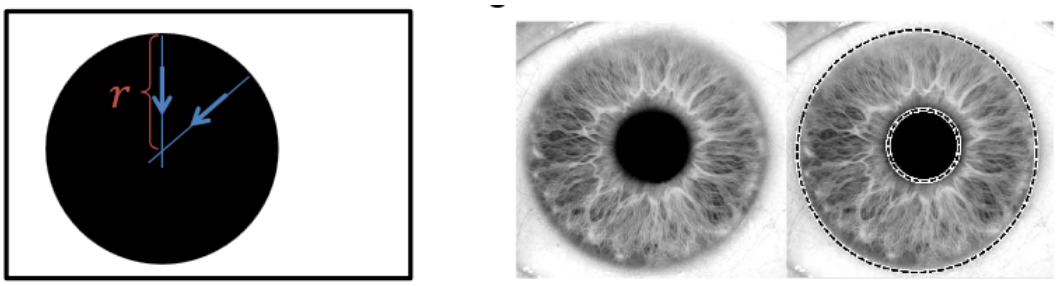


Abbildung 15.1.: Fast Circle Detection using gradient pair vectors

To detect a football during the match there is a complex pipeline process proposed:

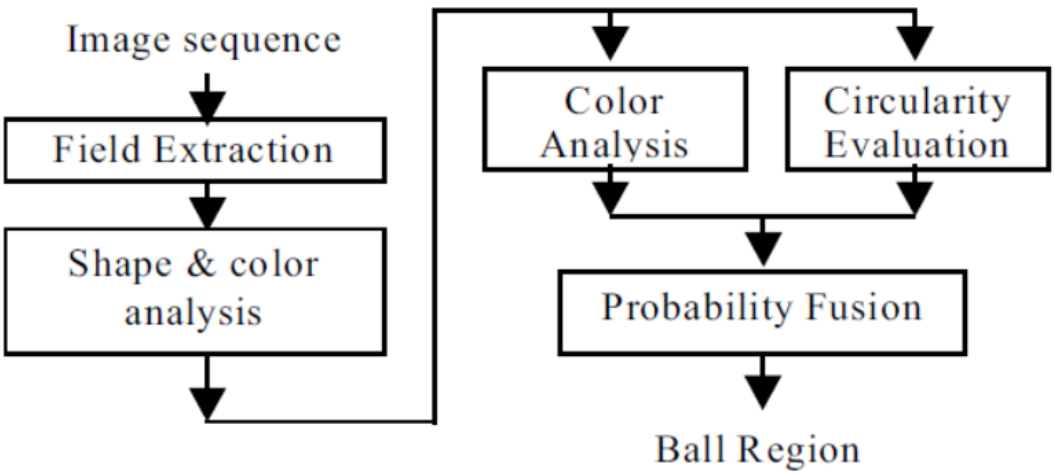


Abbildung 15.2.: An Effective and Fast Soccer Ball Detection and Tracking Method

15.8. Stereo

Another unmentioned effect on the reconstruction of the depth with triangulation is the focal length.:

- small focal Length: higher range of depth values mapped to one pixel ⇒ higher z error
- higher focal length: smaller field of view ⇒ harder correspondence search

15.9. Calibration

Standard calibration for cameras:

- intrinsic parameters:
  - f focal length
  - c_x, c_y principal points (COP on the image plane)
  - k_1, k_2 image resolution
  - distortion parameters, e.g. radial
- Extrinsic
  - R Rotation
  - t translation
- For stereo setup:
  - R Rotation between cameras
  - t translation between cameras

In 15.3 you can see two different calibration Patterns In the checkerboard pattern the

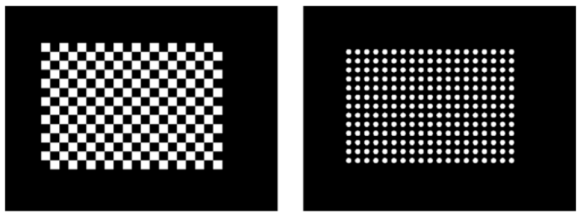


Abbildung 15.3.: Calibration Pattern

points where different patches contact each other are important (only inner points). For a dotted calibration pattern the central points of the dots are important (even blurred images can be used). The Detection can be done with the Hough transformation (lines might be distorted!) or with a direct gradient analysis (neighborhood; circle center).
The extrinsic parameters of a camera can not be calibrated because the world coordinate system is not known (they can be estimated by claiming the checkerboard to

be at [0 0 0]).
In Contrast the intrinsic camera parameters can be all estimated with a plane calibration pattern. The estimation can be done by minimizing the reprojection error of a set of known 3-D points in many different views.
If both camera in stereo setup are not calibrated then the fundamental matrix $F = K^{-T}EK^{-1}$, which is a 3×3 matrice with Rank 2, can be estimated with 7-/8-Points (Seven-point algorithm / eight-point-algorithm) or for an over-determined equation system use SVD.
To calibrate a time-of-flight system or a structured-light (e.g. KINECT) approach you need a calibration pattern with depth information(and for structured-light color information).

A. Anhang