

Artificial Intelligence in Automotive Technology

Johannes Betz / Prof. Dr.-Ing. Markus Lienkamp /Prof. Dr.-Ing. Boris Lohmann



тт

Lecture Overview

1 Introduction: Artificial Intelligence	6 Pathfinding: From British Museum to A*	11 Reinforcement Learning
18.10.2018 – Betz Johannes	29.11.2018 – Lennart Adenaw	17.01.2019 – Christian Dengler
Practice 1	Practice 6	Practice 11
18.10.2018 – Betz Johannes	29.11.2018 – Lennart Adenaw	17.01.2019 – Christian Dengler
2 Perception	7 Introduction: Artificial Neural Networks	12 AI-Development
25.10.2018 – Betz Johannes	06.12.2018 – Lennart Adenaw	24.01.2019 – Johannes Betz
Practice 2	Practice 7	Practice 12
25.10.2018 – Betz Johannes	06.12.2018 – Lennart Adenaw	24.01.2019 – Johannes Betz
3 Supervised Learning: Regression	8 Deep Neural Networks	13 Free Discussion
08.11.2018 – Alexander Wischnewski	13.12.2018 – Jean-Michael Georg	31.01.2019 – Betz/Adenaw
Practice 3	Practice 8	
08.11.2018 – Alexander Wischnewski	13.12.2018 – Jean-Michael Georg	
4 Supervised Learning: Classification	9 Convolutional Neural Networks	
15.11.2018 – Jan-Cedric Mertens	20.12.2018 – Jean-Michael Georg	
Practice 4	Practice 9	
15.11.2018 – Jan-Cedric Mertens	20.12.2018 – Jean-Michael Georg	
5 Unsupervised Learning: Clustering	10 Recurrent Neural Networks	
22.11.2018 – Jan-Cedric Mertens	10.01.2019 – Christian Dengler	
Practice 5	Practice 10	
22.11.2018 – Jan-Cedric Mertens	10.01.2019 – Christian Dengler	



Feedback from last week

- Too much slides / Too much information in too little time
 - \rightarrow AI is a complex topic with different subjects
 - \rightarrow Overall lecture is not enough time (1.5h + Practice)

 \rightarrow We will define learning results for each lecture now: Each slide is just for you to understand everything in total, but the focus of what you should learn (especially for the exam) is more clear

- AI in the detail: First lecture was just there to give an overview and to define the methods and terms → Lecture 2-12 will go into more detail
- Too much typos or spelling mistakes: Yes, shame on me. We will have additional corrections now.



Objectives for Lecture 2: Perception Depth of understanding Remember Understand Utilize Analyze Evaluate Develop

Perception Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Johannes Betz, M. Sc.)

Agenda

- 1. Chapter: Computer Vision
- 2. Chapter: Machine Vision
- 3. Chapter: Image Processing
- 4. Chapter: Features
- 5. Chapter: Feature Analysis
- 6. Chapter: Information Analysis
- 7. Chapter: Summary











What is Artificial Intelligence? - Perception



The world is a complex and dynamic place

What is Artificial Intelligence? - Perception



The world is a complex and dynamic place



What is Artificial Intelligence?

Breaking down the general problem of creating AI into 9 sub-problems:

- 1. Reasoning & Problem Solving
- 2. Knowledge Representation
- 3. Planning
- 4. Learning

- 5. Natural Language Processing (NLP)
- 6. Perception
- 7. Motion and Manipulation
- 8. Social Intelligence





Computer Vision - The Perception Problem

Problem Description:

- A computer is acquiring the ability to perceive the environment
- Sensors are used as input: Camera, Lidar, Ultrasonic, Radar, microphones, ...
- Machine Perception: Capability to interpret data related to the environment
- Computer Vision (CV): The camera input (images/videos) is analyzed and information is extracted

Focus in this lecture: Computer Vision (CV) and Camera

- Camera is the cheapest sensor of the car
- Camera is similar to "Human Eyes"
- Camera imagery includes almost every information about the environment
- Camera images can be process by classical CV-Algorithms and Deep Learning-Algorithms





Computer Vision - Why it matters



Autonomous Driving



Health



Security



Comfort





Access

Source: http://www.magazine.unimore.it/site/home/international-news/articolo820043703.html /https://medium.com/health-ai/heat-screening-for-breast-cancer-computer-vision-for-dieting-brain-wave-passwords-c5294b5e73b / http://osxdaily.com/2017/11/10/can-use-iphone-x-without-face-id/ http://gamechanger.co.ke/product/xbox-360-kinect-sensor/ https://www.express.de/duesseldorf/aceseldorf-archiv/fahndungsfoto-hier-ueberfallen-zwei-raeuber-eine-tankstelle-in-duesseldorf--24421170



Computer Vision – Human Vision

Optic nerve

The optic nerve carries

and toward the brain.

signals away from the eye

Optic nerve

Information from the light-sensitive cells in the eyes is passed to the brain via the optic nerve

Focusing

The lens changes shape depending on the distance to the object, focusing the light onto the retina.

Object

As light hits an object, it is reflected, bouncing away from its surface in all directions.

Lens

As light passes through the lens, its path is bent, focusing the waves in toward the retina.

Optic tract

The optic nerve extends toward a region of the thalamus known as the lateral geniculate nucleus (LGN).

Optic chiasm The optic nerves from each eye cross over as they enter the

brain. The signals from the left side of each eye go to the left side of the brain, and vice versa.

How Humans see

https://www.howitworksdaily.com/science-of-vision-how-do-our-eyes-enable-us-to-see/

Thalamus

The thalamus is situated deep inside the brain, involved in relaying sensory information, including vision, hearing and touch.

Visual cortex

The visual cortex made up of six separate parts, located right at the back of the brain.

Primary visual cortex

Arranged like a map of the retina, it has a large area dedicated to the fovea - the region of the eye responsible for detailed colour vision.

Lateral geniculate nucleus (LGN)

There are LGNs, one on the left, and one on the right. They act as relays and send the information on to the visual cortex.



Computer Vision – Problem for computers

- Objects can be highly variable in shape e.g., trees, cars, animals, ...
- Loss of information in sensing process – 3D objects projected onto 2D images
- Missing data: Occlusions and hidden
 surfaces
- Shadows and noise obscure signal



Illumination Variability



Occlusion



Computer Vision – Problem for computers













Intra Class Variability



Computer Vision – History

- 1966: Minsky assigns computer vision as an undergrad summer project
- 1970's: Some progress on interpreting selected images
- 1980's: ANNs come and go; shift toward geometry and increased mathematical rigor
- 1990's: Face recognition; statistical analysis in vogue → Rule Based
- 2000's: Broader recognition; large annotated datasets available; video processing starts
- 2010's: Deep learning with ConvNets
 → Classification

Source: https://www.slideshare.net/tw_dsconf/computer-vision-crash-course-56934508 / https://ibug.doc.ic.ac.uk/research/detection-static-geometric-facial-features/ https://computervisionblog.wordpress.com/2018/03/25/deep-learning-approaches-for-object-detection/



"In 1966, Minsky hired a first-year undergraduate student and assigned him a problem to solve over the summer:

connect a camera to a computer and get the machine to describe what it sees." Crevier 1993, pg. 88







Computer Vision – The CV-Pipeline



Source: https://picclick.com/Point-Grey-Flea2-FL2-08S2M-IEEE-1394-Mono-Digital-Camera-252748955187.html / https://www.pexels.com/photo/man-holding-black-silver-bridge-camera-taking-photo-during-daytime-167571 / https://de.wikipedia.org/wiki/Datei:OpenCV_Logo_with_text.png /

ttps://www.researchgate.net/publication/323952018_Practices_and_pitfalls_in_inferring_neural_representations/figures?lo=1&utm_source=google&utm_medium=organic / https://stackoverflow.com/questions/35299878/probabilistic-hough-transform-line-segment-connection / https://www.autoscout24.de/auto/audi/audi-r8/

Additional Slide

Interactive Lecture Code: Access the camera

1. Choose the right hardware (camera and lens): For choosing the right camera, there is **no** general procedure. You have to choose regarding the application you want to achieve, the software you are using and on your budget. In the automotive sector the company *FLIR (https://eu.ptgrey.com/)* provides a good portfolio of different camera types.



In this lecture we are using the ZED Stereo Camera (<u>https://www.stereolabs.com/zed/</u>) which provides a dual 4MP Camera with high frame rates, an USB 3.0 connection and a 110°field of view. Stereolabs provides different SDK's for accessing the camera via C++ and Python Code

2. Choose the right software: In this lecture we are working with **OpenCV in Python3**. All examples we are display are written in Python with the ZED SDK. If you want to use the code with other images, the only thing you have to change **is the way of acquiring an image.**

Lecture Code which is displayed now: 01_Camera_input.py

Perception Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Johannes Betz, M. Sc.)

Agenda

- 1. Chapter: Computer Vision
- 2. Chapter: Machine Vision
- 3. Chapter: Image Processing
- 4. Chapter: Features
- 5. Chapter: Feature Analysis
- 6. Chapter: Information Analysis
- 7. Chapter: Summary











Machine Vision – Image Acquisition



Source: https://www.researchgate.net/publication/281476188_Smartphone_image_acquisition_forensics_using_sensor_fingerprint/figures?lo=1 / https://picclick.com/Point-Grey-Flea2-FL2-08S2M-IEEE-1394-Mono-Digital-Camera-252748955187.html / https://de.wikipedia.org/wiki/Datei:OpenCV_Logo_with_text.png / https://www.activesilicon.com/de/product-category/frame-grabbers-de/ http://home.bt.com/tech-gadgets/computing/give-your-pc-a-new-lease-of-life-5-upgrade-optionsyou-should-be-considering-11363940257353



Machine Vision – Image Acquisition

 Lights: The human eye can see well over a wide range of lighting conditions, but a machine vision system is not as capable. You must therefore carefully illuminate the scene under observation so that the machine vision system can "see" it clearly.

→ Problem for Autonomous Driving: Light changes during the day, weather etc.

Optics and lenses: The lens gathers the light reflected (or transmitted) from objects in the camera's field-of-view, and forms an image in the camera sensor. The proper lens allows you to adjust the field-of-view (FOV) and to place the camera at a convenient working distance from the scene. To pick the proper lens you will first need to know the FOV and the working distance. The FOV is the size of the area you want to capture.



Machine Vision – Image Acquisition

- Camera: The camera contains a sensor that converts light from the lens into electrical signals. These signals are digitized into an array of values called pixels. The resolution (precision) of the inspection depends upon the working distance, the field-of-view (FOV), and the number of physical pixels in the camera's sensor. A standard VGA camera has 640 x 480 physical pixels (= width x height), and each physical pixel is about 7.4 microns square
- Framegrabber and Software: Normally a digital Framegrabber sends the digitialzed image over a cable/Bussystem (FireWire, USB 2.0/3.0, Ethernet) into the computer. The grabbing can be done asynchronous or synchronous. After a frame was grabbed the image is put in the computer memory and can be processed by specific software e.g. OpenCV

Additional Slide

Image Resolution



Additional Slide

Interactive Lecture Code: Capture an Image

- 1. First of all we have to access the camera again.
- 2. In the second step, we have to parametrize the camera:
 - 2.1 Resolution of the Image: VGA, HD720, HD1080, HD2k
 - 2.2 Framerate per second: 15,30,60,100
 → Not every framerate works with every resolution
 - 2.3 Initialize the camera.
- 3. Chose how many Images should be taken.
- 4. Save the images to a folder.

Lecture Code which is displayed now: **02_Image_capture.py**



Machine Vision – Computer Vision Software

- Commercial Software (Free for students)
- Computer Vision System Toolbox
- Language: Matlab
- Operating System: Linux, macOS, Windows
- Includes:
 - Input/Output graphics, CV-basics, camera calibration, feature detection/extraction, deep learning
- Pros: Easy to use, good documentation, GPU boost
- Cons: Closed Environment, Performance







Machine Vision – Computer Vision Software

- Free software for **everyone**
- OpenCV (Open Source Computer Vision)
- Language: C++, Python
- Operating System: Linux, macOS, Windows
- Includes:
 - Input/Output Graphics, CV Basics, Camera Calibration, Feature Detection/Extraction, Deep Learning, Egomotion, Motion Tracking
- Pros: Everything you need, Good
 Documentation, powerful, GPU boost
- **Cons:** Installation (especially GPU)





Perception Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Johannes Betz, M. Sc.)

Agenda

- 1. Chapter: History, Pipeline, Definitions
- 2. Chapter: Machine Vision
- 3. Chapter: Image Processing
- 4. Chapter: Features
- 5. Chapter: Feature Analysis
- 6. Chapter: Information Analysis
- 7. Chapter: Summary



٦Π







Image Processing

Image processing is a method to perform some operations on an image in order to get an **enhanced image** or to **extract** useful information from it.

In Image processing different **computer vision algorithms/ mathematical operations** are used







Image Processing – The Pipeline





Image Processing – Camera Calibration





Estimate 3-D Structure from Camera Motion





Measure Planar Objects

- A camera projects 3D world-points onto a 2D image plane
- Todays cameras are cheap and produce distortion (radial, tangential)
- Camera Calibration: Finding the internal quantities of the camera and lens that affect the imaging processing:
 - Extrinsic camera parameters: Position of camera center, camera heading..

Estimate Depth Using a Stereo Camera

- Intrinsic camera parameters: Focal length, image sensor format,...
- Lens distortion parameters

Additional Slide

Types of Distortion

Real cameras use curved lenses to form an image, and light rays often bend a little too much or too little at the edges of these lenses. This creates an effect that distorts the edges of images, so that lines or objects appear more or less curved than they actually are. This is called **radial distortion**, and it's the most common type of distortion.

Another type of distortion, is **tangential distortion**. This occurs when a camera's lens is not aligned perfectly parallel to the imaging plane, where the camera film or sensor is. This makes an image look tilted so that some objects appear farther away or closer than they actually are.



Image Processing – Camera Calibration Process



1. Distorted Picture: Lines or objects appear more or less curved than they actually are.



2. Find and draw corners in the picture → Find the distortion coefficients and correction



3. Undistortion: Compute the distortion matrix and apply it to the picture

Additional Slide



Additional Slide

Interactive Lecture Code: Calibrate the Camera

1. Now we want to use our camera for further applications. To get undistorted images, we have to calibrate the camera. We can do this with different ways.

2. First of all we can use an internal calibration software. The ZED SDK is providing such an calibration software which makes it easy and fast to compute the distortion matrix. But be aware, not all camera SDK's provide such a software.

3. We can do the calibration on our own. To do this, we have to write specific code which finds the parameters for distortions of the camera, intrinsic and extrinsic parameters of camera etc. We are using the OpenCV Python Tutorial which can be found here: <u>https://opencv-python-</u> tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html

3.1 Print out a checkerboard : <u>https://www.mrpt.org/downloads/camera-calibration-checker-board_9x7.pdf</u>

3.2 Take at least 10 different pictures with the checkerboard and save them.

3.2 Calculate the corners and distances in each picture.

3.3 Calculate the camera matrix, distortion coefficients, rotation and translation vectors.

3.4 Undistort any image with these parameters.

Lecture Code which is displayed now: 03_Camera_calibration.py



Image Processing – What is an image?

- A **digital image** is a numeric representation, normally binary, of a twodimensional image.
- → Standard: A digital image is just a **matrix of values** based on color.



Scene element

ТШП

- A Digital image consists of a rectangular grid of evenly spaced pixels.
- Each pixel can be thought of as a measurement or sample of the light from a subject.
- Commonly, the original samples are obtained using a scanner or digital camera by **averaging the amount of red, green and blue light** that falls on the sensitive area of each of its CCD sensing elements.





Image Processing – Digital Transformation & Color Space

- Images represented as a matrix
- Suppose we have a NxM RGB image with a variable called im
 - -im(1,1,1) = top-left pixel value in R-channel
 - -im(y, x, b) = y pixels down, x pixels to right in the bth channel
 - -im(N, M, 3) = bottom-right pixel in B-channel





Image Processing – Color Spaces

- Reproducing human color perception/vision
- **Different color spaces for different problems** (monitor, printer, ...)
- Different color spaces for more exact color extraction
- Most common color spaces are RGB and HSV



Source: https://old.medialooks.com/mformats/docs/CK%20Advanced.html https://cs.wikipedia.org/wiki/Soubor:HSV_color_solid_cylinder.png https://design.tutsplus.com/articles/advanced-color-theory-what-is-color-management--cms-26307 https://www.codeproject.com/articles/243610/the-known-colors-palette-tool-final-revision-hopef


Image Processing – Color Spaces

RGB – Red Green Blue

- Values between 0 255 (8bit)
- 0,0,0: Black
- 255,255,255: White





R = 255



G = 255 (R=0,B=0)



B = 255 (R=0,G=0)

RGB

The most common is the RGB color space with 8 bits per channel, corresponding to $(2^8)^3 = 16,777,216$ (approx. 16.8 million) theoretically possible colors.

With 16 bits (per channel) this results in 281,474,976,710,656 (281 trillion) color options





Image Processing – Color Spaces

HSV – Hue Saturation Value

- Hue (0° Red, 120° Green, 240° Blue)
- Saturation (0% Gray 100% saturated Color)
- Value (0% dark 100% light)





Н (S=1,V=1)



S (H=1,V=1)



V (H=1,S=0)

Interactive Lecture Code: Live Color Change

1. Now we are showing how to extract different colors or color spaces from an image.

2. First we are displaying a live stream from the camera. The live stream is just displaying one frame/image after another. Each frame/image can be seen as a matrix of values.

3. We are changing the current live stream to a greyscale image.

4. We are splitting the live stream into three images. Each image is showing either the red, blue or green colors of the live stream.

5. At the end, we are changing the color space from RGB to HSV. In addition, we are splitting the HSV live stream into three images. Each image is showing the Hue, Saturation and Value of the live stream.

Lecture Code which is displayed now: **04_Change_color_spaces.py**



Image Processing – Filtering

- Image filters in spatial domain: Filter is a mathematical operation of a grid of numbers → Methods: Smoothing, sharpening, measuring texture
- Image filters in the frequency domain: Filtering is a way to modify the frequencies of images → Methods: Denoising, sampling, image compression
- Templates and Image Pyramids: Filtering is a way to match a template to the image → Methods: Detection, coarse-to-fine registration



Image Processing – Filtering

- Preprocessing image to **emphasize information** for next steps
- Mathematic multiplication with quadratic or odd kernel, convolution
 matrix or mask
- Convolution kernel depending on purpose/effect





Image Processing – Filtering Examples



Gaussian Filter

- Remove "high-frequency" components from the image (low-pass filter) Images become more smooth
- Convolution with itself is another Gaussian
 - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
 - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- Separable kernel
 - Factors into product of two 1D Gaussians





0.0030.0130.0220.0130.0030.0130.0590.0970.0590.0130.0220.0970.1590.0970.0220.0130.0590.0970.0590.0130.0030.0130.0220.0130.003

 $G_{\sigma} = \frac{1}{2z^{-2}}e^{-\frac{(x^2+y)}{2\sigma^2}}$

 $5 \times 5, \sigma = 1$

Interactive Lecture Code: Image Filter

1. Now we are showing how to apply a filter to a live image.

2. First we are displaying a live stream from the camera. The live stream is just displaying one frame/image after another. Each frame/image can be seen as a matrix of values.

- 3. We are changing the current live stream to a greyscale image.
- 4. In the next step we are applying a 2D Convolutional Filter to the live stream.
- 5. After that we are applying a Blurring/smoothing filter (Gaussian Blur) Filter to the live stream.

Lecture Code which is displayed now: 05_Image_filtering.py

Image Processing – Contrast Enhancement

Contrast: Contrast is the range of difference between different tones in a photograph. In black and white photography, contrast describes the difference between the darkest and lightest tones, but it also defines the grayscale. In color photography, contrast applies to how sharply colors stand out from one another.

- Low contrast → Image values concentrated near a narrow range (mostly dark, or bright or medium values).
- Contrast of an image can be revealed by its histogram: Representation of the tonal distribution in an digital image.
- Contrast enhancement → Change the image value distribution to cover a wide range.



P(f) = Number of Pixel f = tonal distribution



Image Processing – Contrast Enhancement READING HISTOGRAMS





Image Processing – Contrast Enhancement

Methods



Histogram Equalization

Adaptive Histogram Equalization: CLAHE (Contrast Limited Adaptive Histogram Equalization)

Interactive Lecture Code: Contrast Enhancement

- 1. Now we are showing how to do a contrast enhancement for an image
- 2. First we are taking an image with the camera
- 3. Then we are plotting a histogram, which gives us an overview over the pixel density
- 4. At the end we are applying a histogram equalization with the function cv2.equalizeHist

Lecture Code which is displayed now: 06_Contrast_Enhancement.py



Image Processing – Affine Transformations

- In geometry, an affine transformation, affine map or an affinity (from the Latin, affinis, "connected with") is a function between affine spaces which preserves points, straight lines and planes.
- Sets of parallel lines remain parallel after an affine transformation.
- An affine transformation does not necessarily preserve angles between lines or distances between points, though it does preserve ratios of distances between points lying on a straight line.







Image Processing – Re-Sampling

- Resampling is the mathematical technique used to create a new version of the image with a different width and/or height in pixels. Increasing the size of an image is called upsampling; reducing its size is called downsampling.
- A value for each cell in the new raster object must be computed by sampling or interpolating over some neighborhood of cells in the corresponding position in the original raster object.





Image Processing – Re-Sampling



Perception Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Johannes Betz, M. Sc.)

Agenda

- 1. Chapter: History, Pipeline, Definitions
- 2. Chapter: Machine Vision
- 3. Chapter: Image Processing
- 4. Chapter: Features
- 5. Chapter: Feature Analysis
- 6. Chapter: Information Analysis
- 7. Chapter: Summary



٦Π







Feature Extraction



Which of the following features could be useful in the identification of lane lines on the road?

ТШП

Feature Extraction

- A **Feature** is a piece of information which is relevant for solving the computational task related to a certain application
- Feature Detection includes methods for computing abstractions of image information and making local decisions at every image point whether there is an image feature of a given type at that point or not.
 - The resulting features will be subsets of the image domain, often in the form of isolated points, continuous curves or connected regions.
 - o Examples: Edges, Corners, Region of Interest, Ridges
- Feature Extraction: Once features have been detected, a local image patch around the feature can be extracted. For Computer Vision this means the isolation of various desired portions or shapes (features) of a digitized image or video stream



Feature Extraction

- Feature points are used for:
 - Image alignment
 - 3D reconstruction
 - Motion tracking
 - Robot navigation
 - Indexing and database retrieval
 - Object recognition









Feature Extraction – Edge Detection

- **Goal:** Identify sudden changes (discontinuities) in an image
 - Intuitively, most semantic and shape information from the image can be encoded in the edges
 - More compact than pixels
- Ideal: Artist's line drawing (but artist is also using object-level knowledge)





Feature Extraction – Edge Detection





Feature Extraction – Edge Detection





Feature Extraction – Canny Edge Detection

- Don't do Edge Detection by yourself: Use the Canny Edge Detection Algorithm
- This is probably the most widely used edge detector in Computer Vision
- Theoretical model: step-edges corrupted by additive Gaussian
 noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization

Paper:

J. Canny, <u>A Computational Approach To Edge Detection</u>, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

27,000 citations!



Feature Extraction – Canny Edge Detection



- 1. Filter image with x, y derivatives of Gaussian
- 2. Find magnitude and orientation of gradient
- 3. Non-maximum suppression: Thin multi-pixel wide "ridges" down to single pixel width
- 4. Thresholding and linking (hysteresis):
 - Define two thresholds: low and high
 - Use the high threshold to start edge curves and the low threshold to continue them
- Adjust Thresholds: Threshold at low/high levels to get weak/strong edge pixels
- Adjust Kernel-Size: Large kernel detects large scale, low kernel fine features

Interactive Lecture Code: Canny Edge Detection

1. Now we are showing how to apply a canny edge detection to a live image.

2. First we are displaying a live stream from the camera. The live stream is just displaying one frame/image after another. Each frame/image can be seen as a matrix of values.

- 3. We are changing the current live stream to a greyscale image.
- 4. After that we are applying a Blurring/smoothing filter (Gaussian Blur) filter to the live stream.

5. At the end, we are applying a Canny Edge Detection to the livestream. https://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html

Lecture Code which is displayed now: 07_Canny_Edge_Detection.py

Feature Extraction – Line, Circle,... Detection

The Problem

- Edge detection makes it possible to reduce the amount of data in an image considerably. However the output from an edge detector is still an image described by it's pixels.
- If lines, ellipses and so forth could be defined by their characteristic equations, the amount of data would be reduced even more.
- We can use the Hough Line Algorithm, which was originally developed to recognize lines and has later been generalized to cover arbitrary shapes e. g. circles



Feature Extraction – Hough Line



The algorithm for detecting straight lines can be divided into the following steps:

- 1. Edge detection, e.g. using the Canny edge detector
- 2. Mapping of edge points to the Hough space and storage in an accumulator.
- **3. Interpretation** of the accumulator to yield lines of infinite length. The interpretation is done by thresholding and possibly other constraints.
- 4. Conversion of infinite lines to finite lines.





Feature Extraction – Hough Line Example





Interactive Lecture Code: Hough Line Detection

1. Now we are showing how to apply a canny edge detection to a live image

2. First we are displaying a live stream from the camera. The live stream is just displaying one frame after another. Each frame can be seen as a matrix of values

- 3. We are changing the current live stream to a greyscale image
- 4. After that we are applying a Blurring/smoothing filter (Gaussian Blur) Filter to the live stream
- 5. After that we are applying a Canny Edge Detection to the livestream.

6. At the end we are applying a Hough line detection with the function cv2. <u>https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html</u> The function needs different parameters for input

Lecture Code which is displayed now: 08_Hough_Line_Detection.py



Feature Extraction – Corner Detection

- We should easily recognize the point by looking through a small window
- Shifting a window in **any direction** should give **a large change** in intensity



"flat" region: no change in all directions



"edge": no change along the edge direction



"corner": significant change in all directions



Feature Extraction – Corner Detection



1. Original Image



2. Compute Corner Response R



3. Find Points with larger corner Response



4. Points of local maxima of R



5. Show points in image

Feature Extraction – Depth Detection

■ Disparity between 2 images → disparity map



Disparity d = x1-x2



Feature Extraction – Depth map





Feature Extraction – Dilation, Erosion, Opening, Closing

- Morphology-based operations to connect points and edges and build regions/areas
- A set of operations that process images based on shapes. Morphological operations apply a **structuring element** to an input image and generate an output image.
- Dilation and Erosion are **minkowski addition and subtraction**
- Opening and Closing are **combined** Dilation and Erosion
Additional Slide

Feature Extraction – Dilation, Erosion, Opening, Closing

Dilation:

• The operations consists of convoluting an image A with some kernel (B), which can have any shape or size, usually a square or circle.

• Kernel B has a defined anchor point, usually being the center of the kernel.

• As kernel B is scanned over the image, we compute the maximal pixel value overlapped by B and replace the image pixel in the anchor point position with that maximal value. As you can deduce, this maximizing operation causes bright regions within an image to "grow" (therefore the name dilation). Take as an example the image above. Applying dilation we can get:



Feature Extraction – Dilation, Erosion, Opening, Closing

Erosion:

This operation is the sister of dilation. What this does is to compute a local minimum over the area of the kernel.

As kernel B is scanned over the image, we compute the minimal pixel value overlapped by B and replace the image pixel under the anchor point with that minimal value. Analogously to the example for dilation, we can apply the erosion operator to the original image (shown above). You can see in the result below that the bright areas of the image (the background, apparently), get thinner, whereas the dark zones (the "writing") gets bigger.



Perception Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Johannes Betz, M. Sc.)

Agenda

- 1. Chapter: History, Pipeline, Definitions
- 2. Chapter: Machine Vision
- 3. Chapter: Image Processing
- 4. Chapter: Features
- 5. Chapter: Feature Analysis
- 6. Chapter: Information Analysis
- 7. Chapter: Summary













Source: https://www.datacamp.com/community/tutorials/object-detection-guide

Additional Slide

Interactive Lecture Code: Image classification

The live software displayed now is based on software for running the live camera recognition demo with an artificial neural network based on the code from https://github.com/dusty-nv/jetson-inference

For running the code you need an Nvidia Jetson TX2

Lecture Code which is displayed now: 08_Hough_Line_Detection.py







Feature Techniques

Raw pixels

 Histograms of features (HOG)...











Random Forest

Feature Analysis – Classification

Methods for Classification:

1. Logic







3. Deep Learning

Source: https://de.mathworks.com/solutions/deep-learning/convolutional-neural-network.html?requestedDomain /https://commons.wikimedia.org/wiki/Logic_diagram







- Tested with
 - □ RGB
 - LAB Slightly better performance vs. grayscale
 - Grayscale
- Gamma Normalization and Compression
 - Square root_
 - Log Log Very slightly better performance vs. no adjustment



Source: Georgia Tech – Lecture CS 6476 Computer Vision – Lecture 18













Source: Georgia Tech - Lecture CS 6476 Computer Vision - Lecture 18







Perception Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Johannes Betz, M. Sc.)

Agenda

- 1. Chapter: History, Pipeline, Definitions
- 2. Chapter: Machine Vision
- 3. Chapter: Image Processing
- 4. Chapter: Features
- 5. Chapter: Feature Analysis
- 6. Chapter: Information Analysis
- 7. Chapter: Segmentation



٦Π







Information Analysis – Optical Flow

- A video is a sequence of frames captured over time
- Now our image data is a function of space
 (x, y) and time (t)



 Optical flow or optic flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene





Information Analysis – Egomotion Estimation

 In robotics and computer vision, visual odometry is the process of determining the position and orientation of a robot by analyzing the associated camera images.



Information Analysis – Egomotion Estimation

- 1. Acquire input images: using camera (Mono, Stereo,...)
- 2. Image correction: apply image processing techniques
- 3. Feature detection: define interest operators, and match features across frames and construct optical flow field.
 - 1. Use correlation to establish correspondence of two images, and no long term feature tracking.
 - 2. Feature extraction and correlation.
 - 3. Construct optical flow field (Lucas–Kanade method).
- 4. Check flow field vectors for potential tracking errors and remove outliers.
- 5. Estimation of the camera motion from the optical flow.
 - 1. Choice 1: Kalman filter for state estimate distribution maintenance.
 - 2. Choice 2: find the geometric and 3D properties of the features that minimize a cost function based on the re-projection error between two adjacent images. This can be done by mathematical minimization or random sampling.
- 6. Periodic repopulation of trackpoints to maintain coverage across the image.



Information Analysis – Object Tracking

Perception Johannes Betz / Prof. Dr. Markus Lienkamp / Prof. Dr. Boris Lohmann

(Johannes Betz, M. Sc.)

Agenda

- 1. Chapter: History, Pipeline, Definitions
- 2. Chapter: Machine Vision
- 3. Chapter: Image Processing
- 4. Chapter: Features
- 5. Chapter: Feature Analysis
- 6. Chapter: Information Analysis
- 7. Chapter: Summary



٦Π







Summary

What did we learn today:

- **Perception** is an Artificial Intelligence problem
- We can divide perception into Machine Vision and Computer Vision
- We focus on Machine Vision and Computer Vision → The ability to retrieve information out of a camera
- Reasons for using a camera in a vehicle: cameras are cheap, easy to access and provide a lot of information about the current environment
 → Similarity to human eye
- There are important differences between Machine Vison setups: We need to know what camera, lenses, computer setup and environment we are operating in when using machine vision



Summary

What did we learn today:

- We defined a computer vision pipeline
- Matlab/Simulik and the OpenCV Library are the first software tools to choose for processing digital images
- First of all we have to process the image for getting better information (features) out of one picture: Denoise, Contrast Enhancement, Color Space change...
- Then we have to extract the features like Edges, Corners,...
- We can then use the features for getting information out of the picture: Detect and Recognize objects, Classify objects,...
- We can use machine learning algorithms