Fahrraumdatenerfassung

Erkennung und Positionierung von Verkehrszeichen

Vortrag von Dominik Wetzel

zum Projekt

„Mess- und Testfahrzeug zur Aufnahme von Basisdaten für die Sicherheitsüberprüfung von Straßen und das autonome Fahren“

Westsächsische Hochschule Zwickau
University of Applied Sciences

Inhalt

- Projektüberblick
- Algorithmen
  - Segmentierung
  - Formerkennung
  - Stereo Vision
- Softwaresystem
- Diskussion

Projektüberblick

Fahrraumdatenerfassung

- Strecke abfahren
- Daten aufnehmen
- Daten auswerten
- 3D-Modell erstellen (für Simulator)

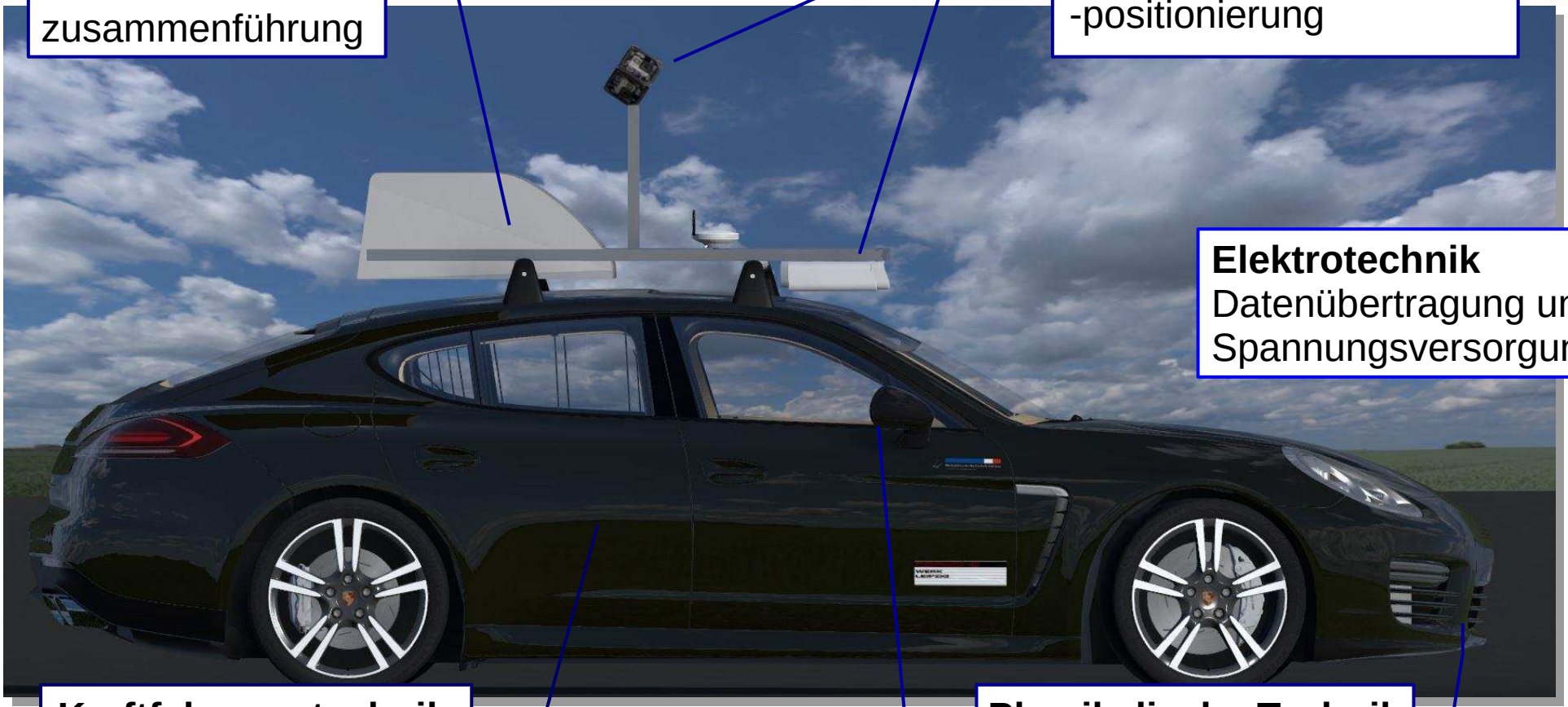
Quelle: Hendrik Weiß

Quelle: Fahrsimulator

Beteiligte

Informatik
Drohne und Daten-zusammenführung

Mathematik
Objekterkennung und -positionierung



Elektrotechnik
Datenübertragung und Spannungsversorgung

Kraftfahrzeugtechnik
Fahrzeugaufbau und Streckenmodellierung

Quelle: Ronny Häupl

Physikalische Technik
Querneigung und Straßenbreite

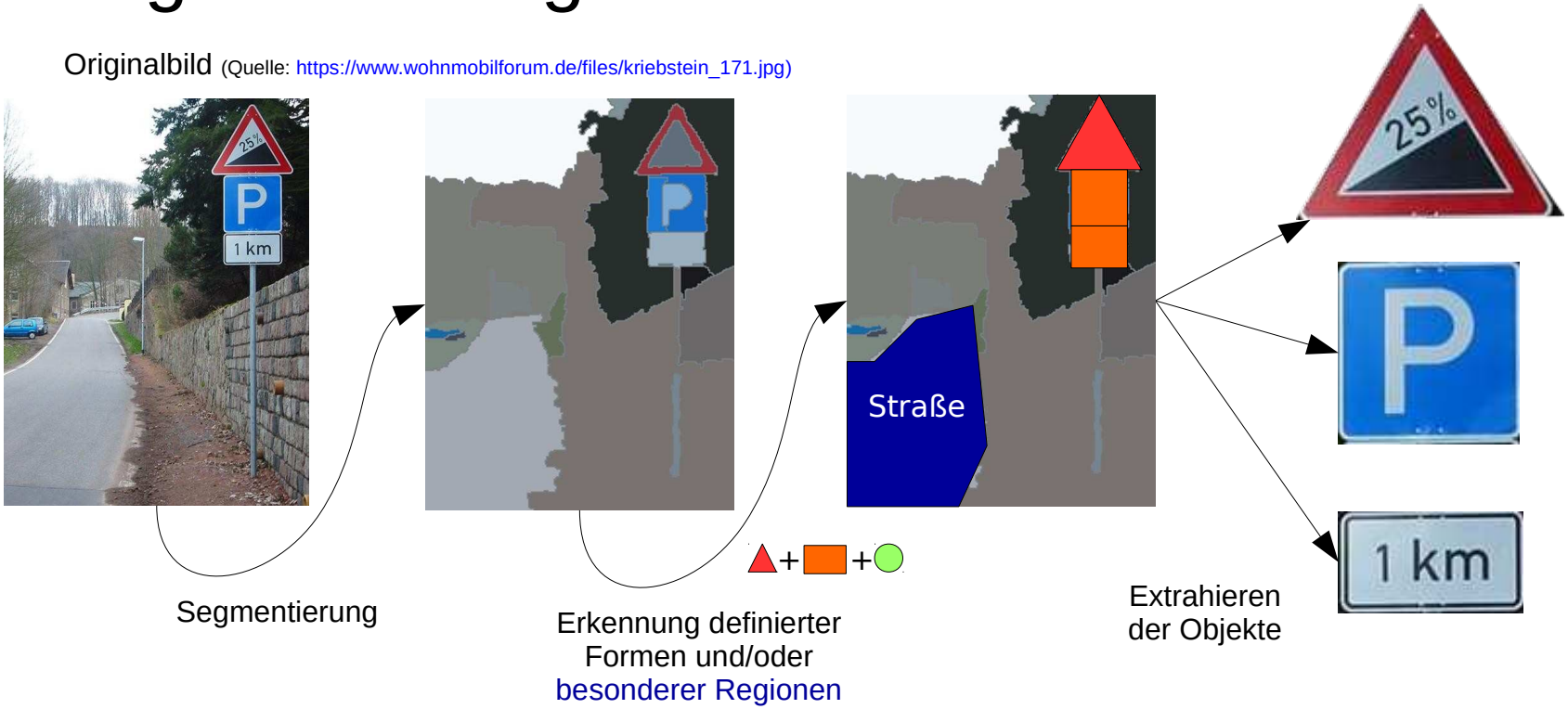
Aufgabe

- Erkennen fahrraumprägender Objekte
  - Verkehrszeichen
  - Leitplanken
  - Leitpfosten
- Positionierung relativ zum Fahrzeug
  - X,Y,Z – Koordinaten (in m)
- Extraktion Bilddaten

Algorithmen

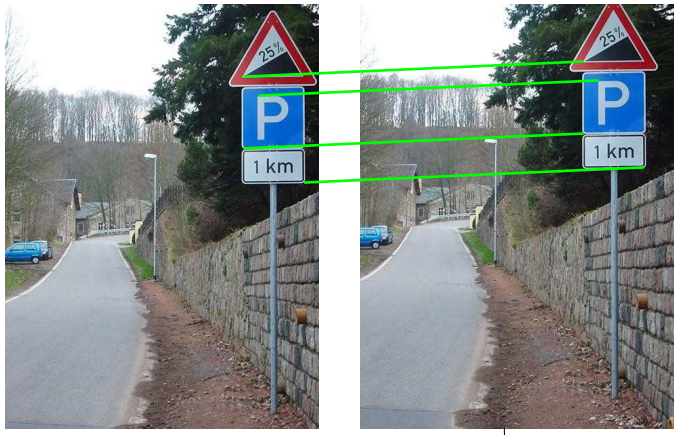
Geplanter Ablauf

- Kameraeinstellung & -kalibrierung → Aufnahme
- Segmentierung → Geometrie → Extraktion →

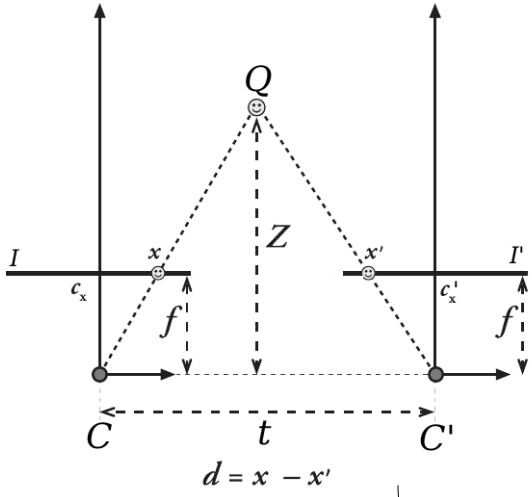


Geplanter Ablauf

Korrespondenzen → Triangulierung → Position



Markante Punkte und
Korrespondenzen finden



Quelle: Learning OpenCV, S. 208

Position mittels
Triangulation berechnen

oberster Punkt:
X = 1,2m
Y = 2,5m
Z = 2,0m

Segmentierung

- Allgemein
- Farbsegmentierung
- ROI-Erkennung

Allgemein

- komplettes Bild in Segmente aufteilen
  - SLIC, Superpixel, Liniendetektion
  - Problem: Laufzeit, unwichtige Bereiche, weiteres Vorgehen trotzdem nötig

- Farbsegmentierung
  - nur Bereiche von Interesse → Schwellwert
  - anschließend „Region of Interest“ (ROI) Erkennung

Farbsegmentierung

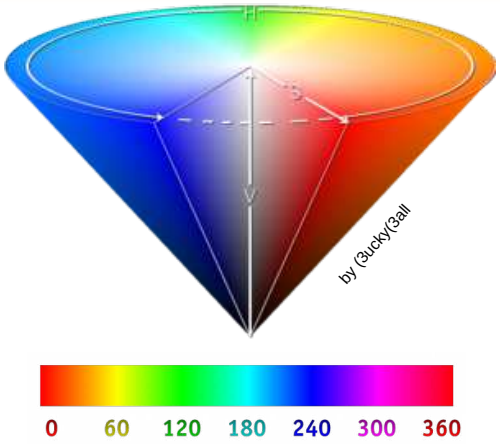
- RGB-Farbraum
  - visuelles Empfinden (Maschinentauglich)
  - Farbnuancen ändern alle Kanäle
  - Normierung:

$$\forall C_{x,y} \in I \wedge C \in \{R, G, B\} : \frac{C_{x,y}}{\sqrt{R_{x,y}^2 + G_{x,y}^2 + B_{x,y}^2}}$$

  - nicht erfolgreich

Farbsegmentierung

- HSV-Raum
  - visuellem Empfinden näher
  - Erfahrungswerte?



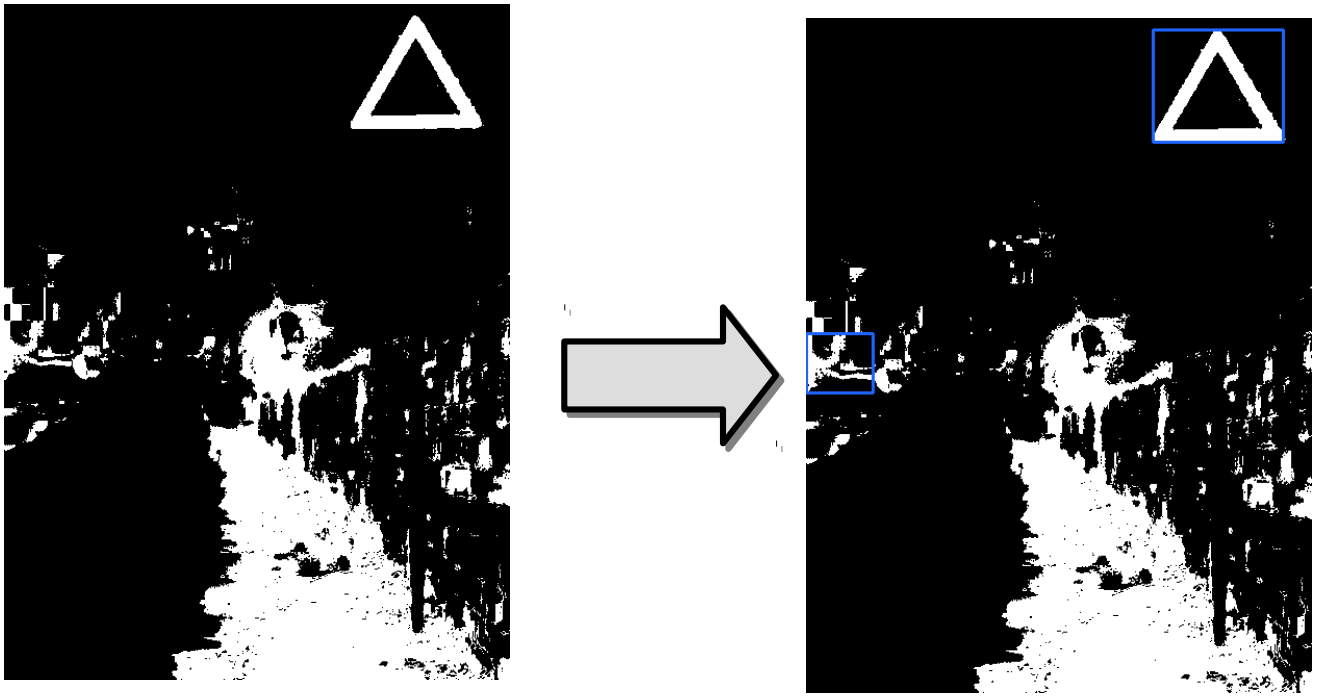
Quelle: Wikipedia

- Common Vision Blox Color
  - nutzt CIE-Lab-Raum
  - lernt Farbmodell mit ø, *min*- & *max*-Distanz
  - Filter → $Distanz < S \cdot (\max - \min) + \min$

    $(S = \text{Schwellwert})$

ROI-Erkennung

- nach Farbsegmentierung:



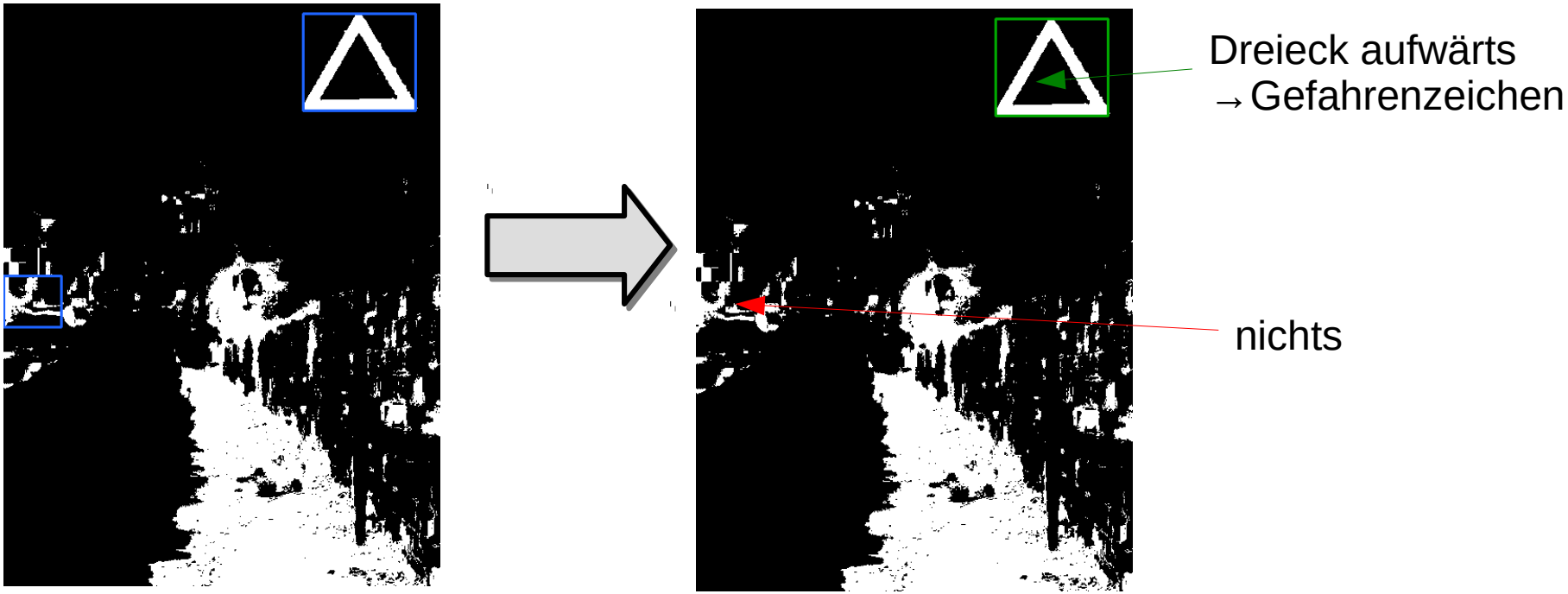
(Parameter: max Fläche, min H, min B, Verhältnis +- Abweichung)

Formerkennung

- Allgemein
- Korrelation
- Gradienten-Verfahren
- Kaskade

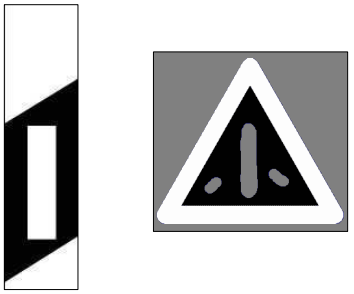
Allgemein

- ROIs klassifizieren



Korrelation (Template)

- Template über Bild schieben
- Template auf ROI anwenden
- Matlab (normxcorr2):



$$\gamma(u,v) = \frac{\sum_{x,y}[I(x,y) - \bar{I}_{u,v}][t(x-u,y-v) - \bar{t}]}{\sqrt{\sum_{x,y}[I(x,y) - \bar{I}_{u,v}]^2 \sum_{x,y}[t(x-u,y-v) - \bar{t}]^2}}$$

$I = \text{Bild}, \bar{t} = \text{Mittelwert Template},$

$\bar{I}_{u,v} = \text{Mittelwert von } I(x,y) \text{ in der Region unter Template}$

Quelle: http://de.mathworks.com/help/images/ref/normxcorr2.html

Korrelation (Template)

- eigene Implementierung:

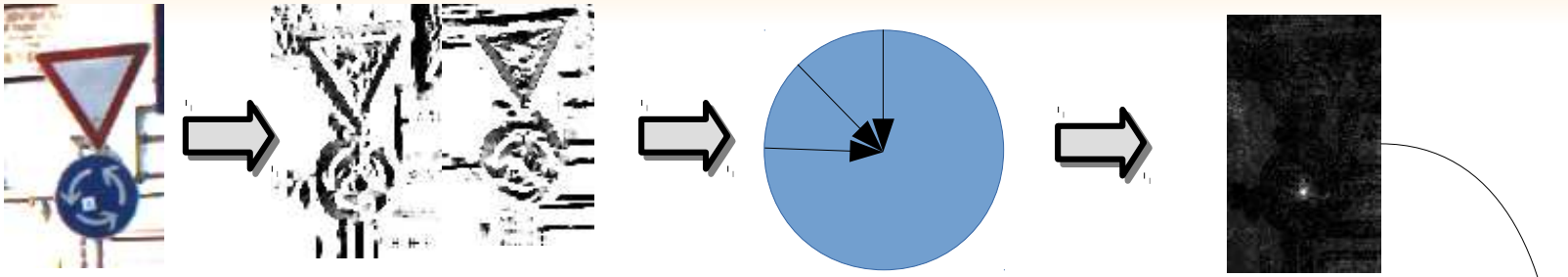
$$\gamma = \frac{\sum_{x,y}(I_{weiß}(x,y) \cdot \omega + I_{schwarz}(x,y))}{\sum_{x,y}(t_{weiß}(x,y) \cdot \omega + t_{schwarz}(x,y))}$$

$$I_c(x,y) = \begin{cases} 1 & falls\ I(x,y) = c \wedge t(x,y) = c \\ 0 & sonst \end{cases}$$

$$t_c(x,y) = \begin{cases} 1 & falls\ t(x,y) = c \\ 0 & sonst \end{cases}$$

$$c \in \{weiß, schwarz\}, \omega = \text{Gewicht}$$

Gradienten-Verfahren



X & Y
normierte Gradienten

Gradientvector * r =
Vote für einen Punkt

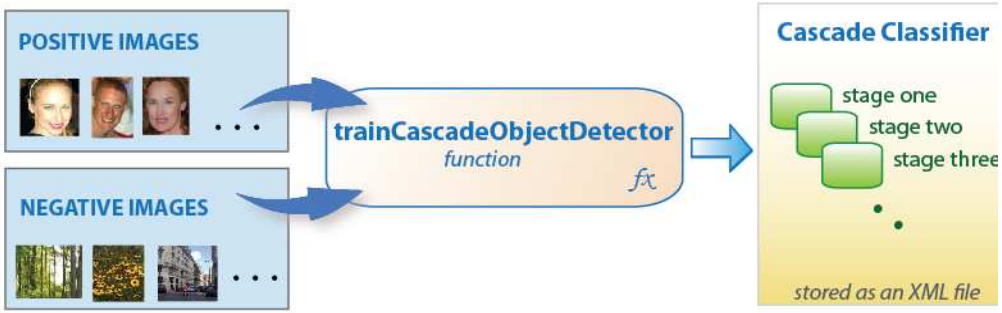
→ Peak im Zentrum
(Bild I_v)

- Erkennung Peak:
  - Skalieren auf 255 Grauwerte (I_{vs})
  - $(max(I_{vs}) - s(I_{vs}))/\bar{I}_{vs} > S$
  - Problem *S* nicht normiert.

- Erweiterbar für gleichmäßige Polygone

Kaskade

- Anlernen von Objekten
  - Positiv Bilder mit ROI von Objekt
  - Negative Bilder ohne Objekt
  - Berechnen von Features (LBP, HOG, Haar-Like)
- Verschiedene Schichten (Stages)
  - Äußere Schicht, grob
  - innere Schicht, fein



Quelle: http://de.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html

Kaskade

- Ausführung:
  - Descriptor für Stage läuft über Bild
  - Je Position:
    - Objekt nicht gefunden → nächste Position
    - Objekt gefunden → ROI in nächste Stage geben → nächste Position
  - Descriptor für nächste Stage verwenden → auf ROIs vorheriger Stage wiederholen
- Verwendung z.B. für Stoppschild

Stereo Vision

- Allgemein
- Intrinsische und exterinsische Parameter
- Kamerakalibrierung
- Punktkorrespondenzen
- Rektifizierung + Positionsbestimmung

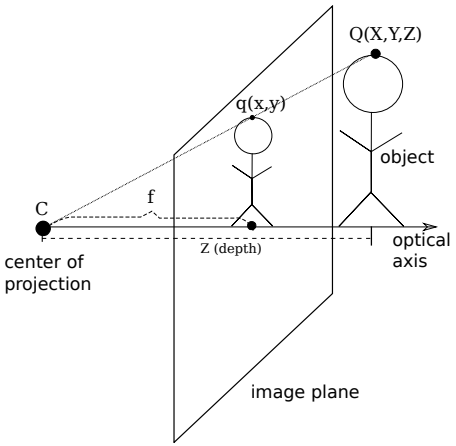
Allgemein

- 2 oder mehr Kameras
- Berechnung der Tiefeninformation
- dazu:
  - kalibrierte Kameras (Intrinsische Parameter) $K, dist$
  - Position Kameras zueinander (Extrinsische Parameter) $R|t$
  - Punktkorrespondenzen in Bildern p, p'

Quelle: http://www.gravitram.com/digital_stereo_rig.htm

Intrinsische Parameter

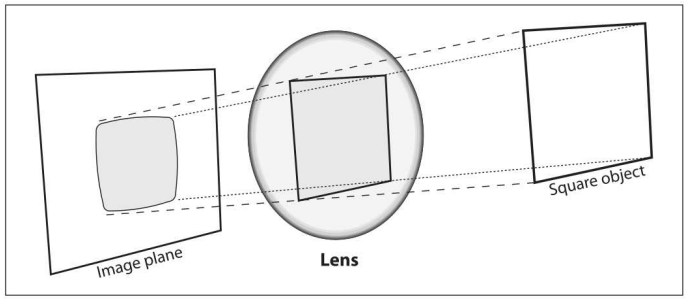
- Lochkameramodell erweitert um Verzerrungen
- K und Verzerrungskoeffizienten (dist)

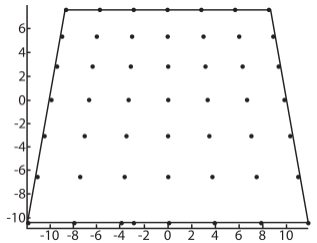


$$q = K \cdot \tilde{Q}, K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, q = \begin{bmatrix} x \\ y \\ w \end{bmatrix}, \tilde{Q} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Quelle:Masterarbeit, Wetzel S.5

$$dist = \{k_1, k_2, k_3, p_1, p_2\}$$

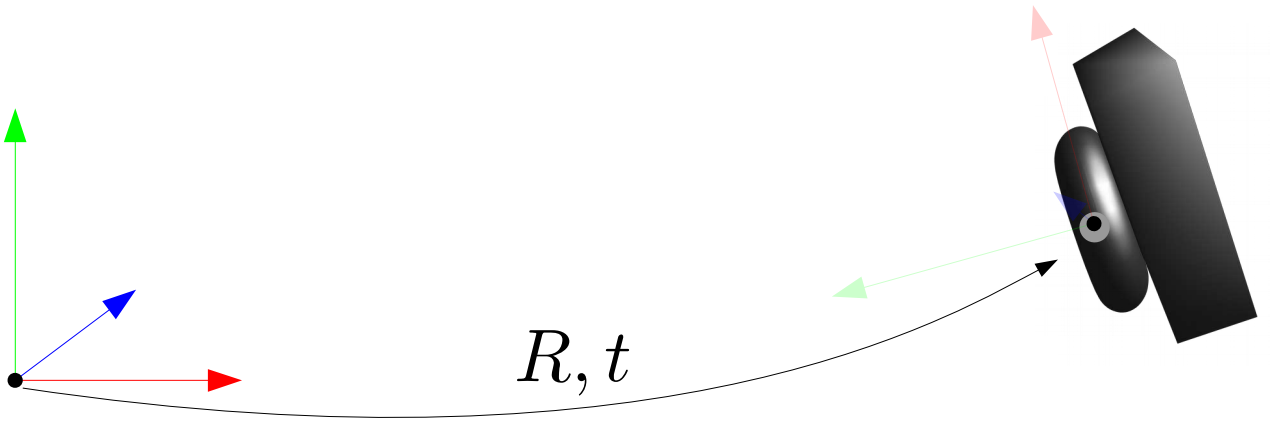




Quelle: Learning OpenCV, S. 376f

Extrinsische Parameter

- Rotation und Translation von Koordinaten-ursprung

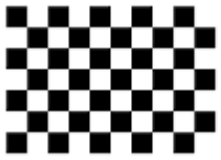


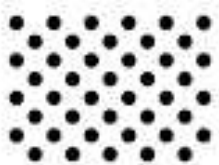
$$R = R_x R_y R_z, t = [t_x, t_y, t_z]^T$$

$$R = [r_1, r_2, r_3], r_{i \in \{1,2,3\}} = [r_{1,i}, r_{2,i}, r_{3,i}]^T$$

Kamerakalibrierung

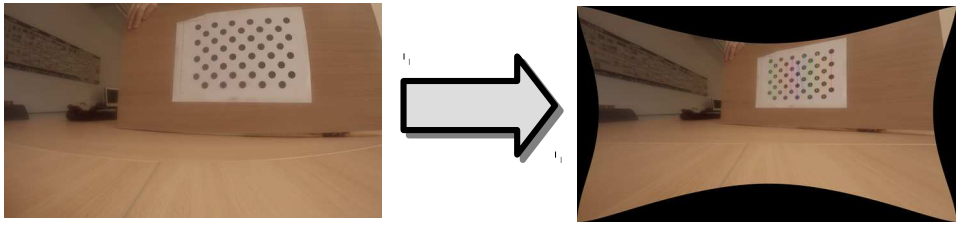
- Unbekannte:
  - 4 Intrinsisch (f_x, f_y, c_x, c_y)
  - 5 Verzerrung $(k_1, k_2, k_3, p_1, p_2)$
  - je Bild 6 Extrinsisch $(r_1, r_2, r_3, t_x, t_y, t_z)$
- Kalibrierungsmuster:
  - Bedingungen = 2 * Punkte * Bilder
  - möglichst viel Bewegung
  - kompletten Sichtbereich





Kamerakalibrierung

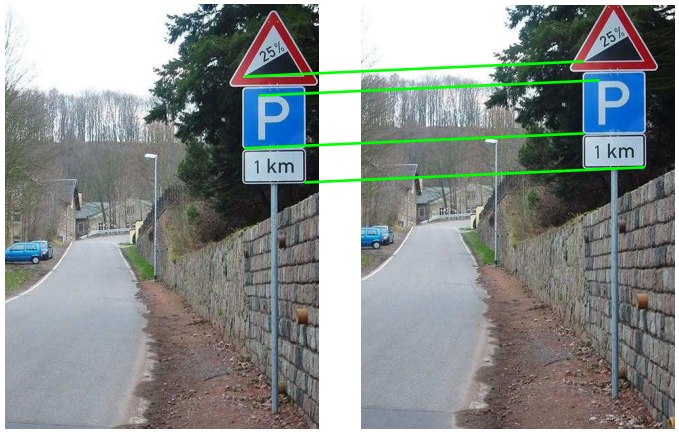
- Entzerrung



- Stereokalibrierung
  - Einzelkalibrierung
  - berechnet $R|t$ für Kameras untereinander
  - Neuberechnung nach Positionsänderung
  - Erlaubt metrische Positionsbestimmung

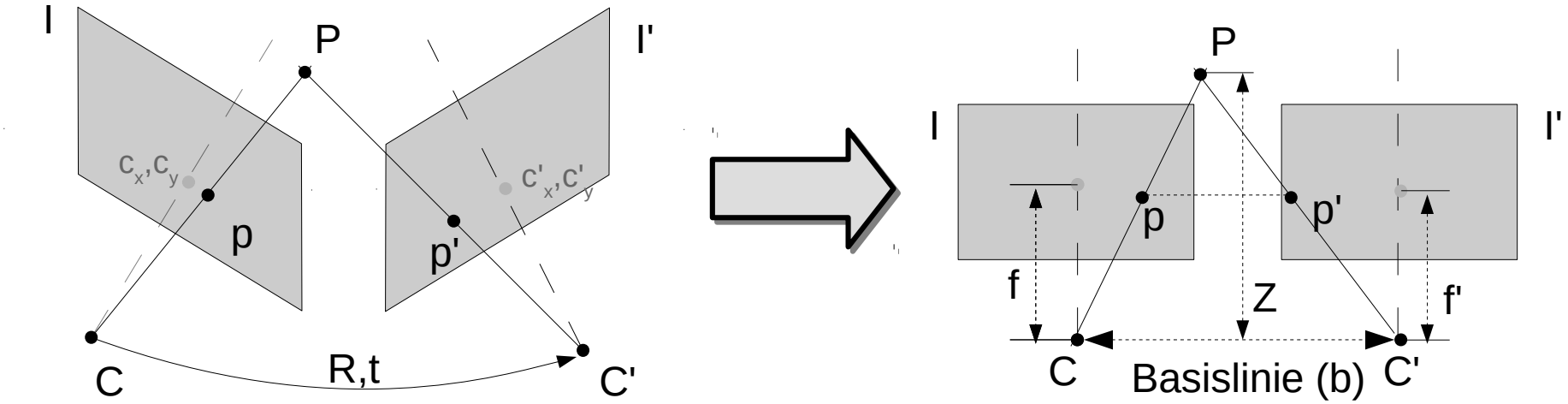
Punktkorrespondenzen

- Gleiche Punkte in 2 Bildern
- Zentrum der gefundenen Objekte
- SIFT, ASIFT, ORB ...



Rektifizierung

- Bildebenen transformieren
- Berechnung Reprojektionsmatrix Q



- Disparität $d = p_x - p'_x$

Positionsbestimmung

- Rektifiziert:

$$Z' = \frac{f \cdot b}{d}$$

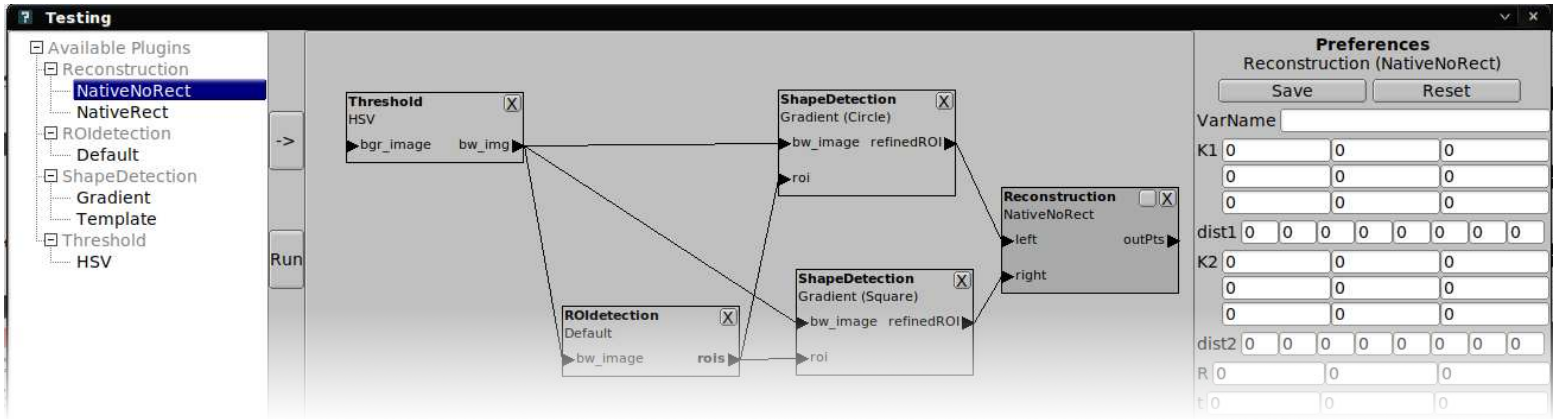
$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix}, Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & 1/b & -(c_x - c'_x)/b \end{bmatrix}$$

Quelle: Learning OpenCV, S. 435

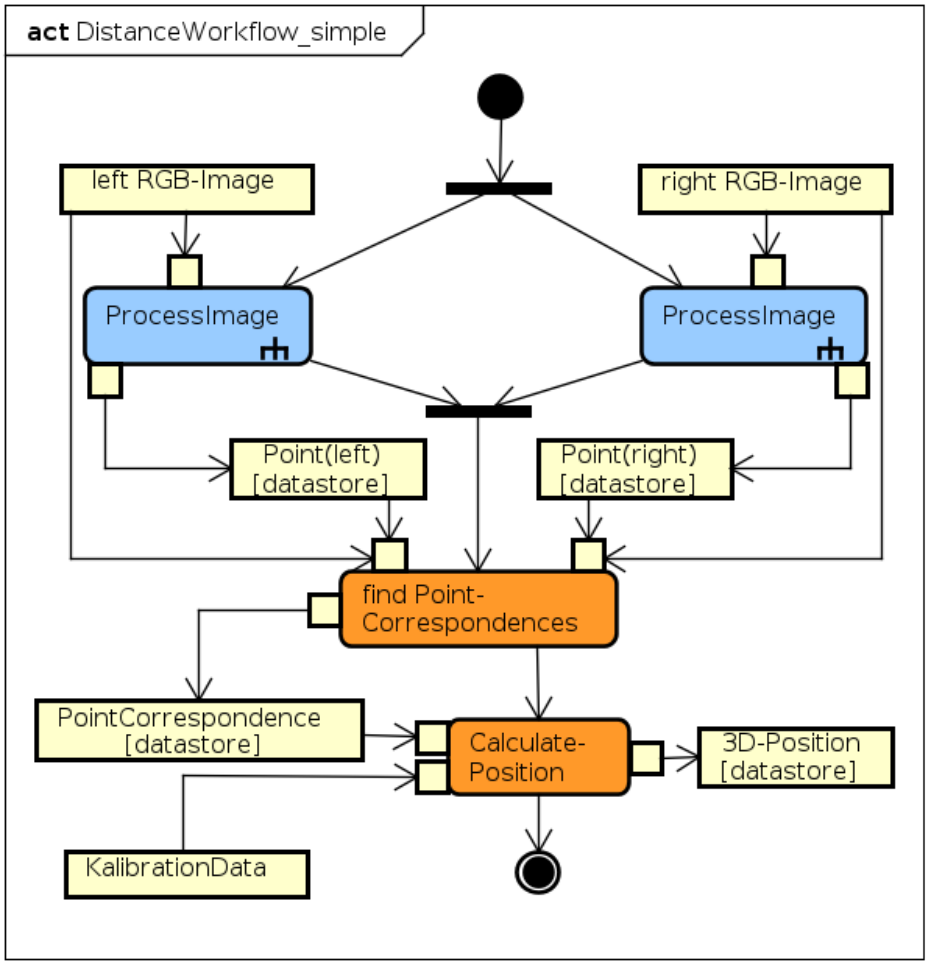
Softwaresystem

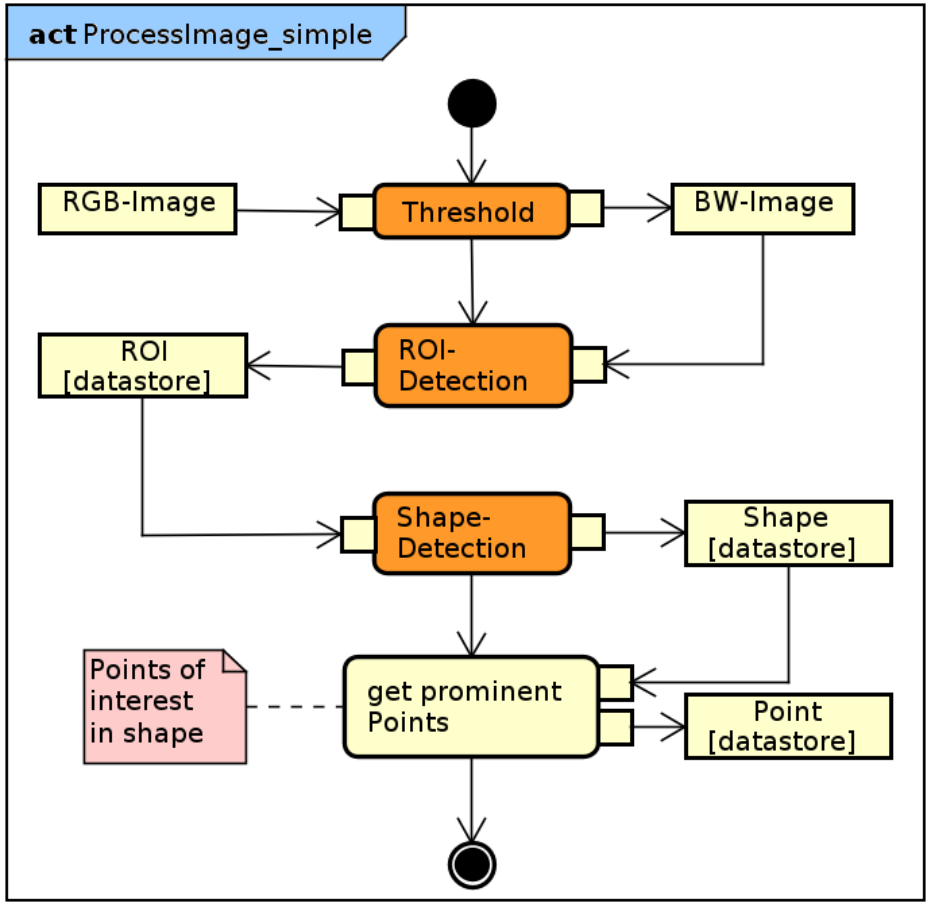
Modularisierung

- Einzelne Algorithmen als Module
- GUI-Frontend um Module einzustellen und zu verknüpfen (wie Simulink)
- Ermöglicht Erweiterung und Austausch von Algorithmen



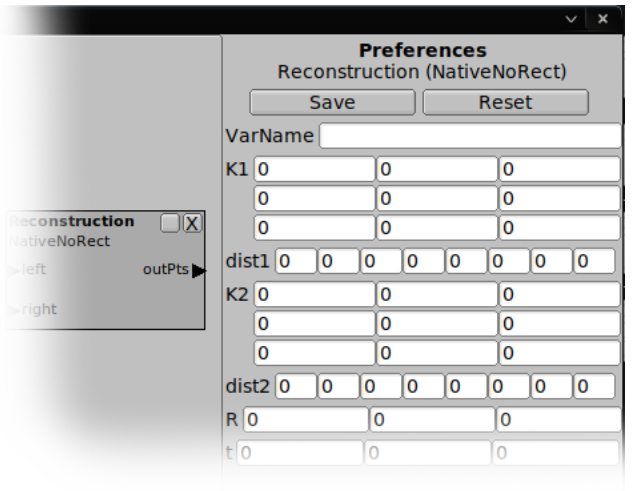
Module (orange)





Datenmodell

- einfaches Datenmodell zur Typübertragung für Input-, Output- und Konfigurationsparameter
  - F → Datei (File) bzw. Zeichenkette
  - M → Matrix oder Bild
  - D → Zahlen (double)
- Größenangabe möglich:
  - z. B.: D4, M3x3, M3
    → dynamisches Konfigurations-panel



Datenmodell

- Konfigurationen speichern/laden
- JSON-Dateien (Hauptdatei + Ordner)
- Format:

Hauptdatei.json:

```
{
  “Instances“: [
  [
    <PluginName>, <Konfig.json>,
    <Pos x>. <Pos y>
  ], …]
  ,
  “Connections“: [
  [
    [<InPluginNr>,<ParamNr>],
    [<OutPluginNr>,<ParamNr>]
  ], …]
}
```

Hauptdatei/Konfig.json

```
{
    "Params": [
        "z.B. String",
        [ [ <double>,<double> ] ],
        …
    ],
    "VarName": <Name>
}
```

Diskussion

Vielen Dank für Ihre Aufmerksamkeit.

Quellen

- G. Bradski, A. Kaehler, *“Learning OpenCV – Computer Vision with the OpenCV Library“*, 2008
- D. Wetzel, *“Master Thesis – Enhanced depth estimation for 'freehand stereo' using PatchMatch Stereo“*, 2015
- Internetquellen *(besucht 19.09.2016)*
  - https://de.wikipedia.org/wiki/HSV-Farbraum
  - http://de.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html
  - https://www.wohnmobilforum.de/files/kriebstein_171.jpg
  - http://de.mathworks.com/help/images/ref/normxcorr2.html
  - http://www.gravitram.com/digital_stereo_rig.htm