Gradient Vector Griding

An edge extraction and clustering method

Naja v. Schmude

Outline

Motivation

Idea of algorithm

Overview over steps of algorithm

- Calculation of *edge representers*
- Extraction of connection graph
- Extraction of edge traces

Optimizations

Results

Motivation

Currently most vision systems rely on the color of the objects, but rules will change to a more and more colorless environment.

- Therefore: Use *shape* of objects and not (only) the color

Real-time image processing necessary as camera is most important sensor

- Need of a fast and reliable image processor

CPU on humanoid robots is strongly limited (e.g. ARM processors)

- Need of a simple method without complex calculations

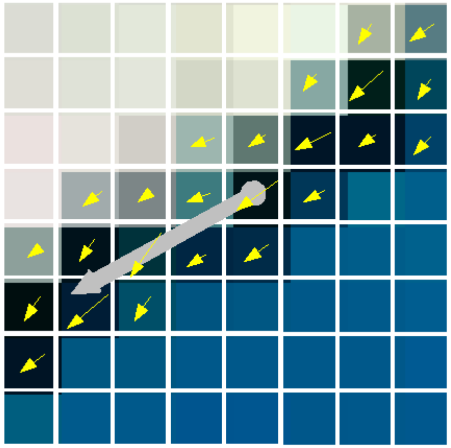
Idea behind Algorithm

Reduce complexity of the whole image to a much smaller *grid* formed from equal-sized cells

Only one pass over the whole image is needed to calculate the features for the cells, afterwards all operations are performed on the grid and its features

Steps of the Algorithm

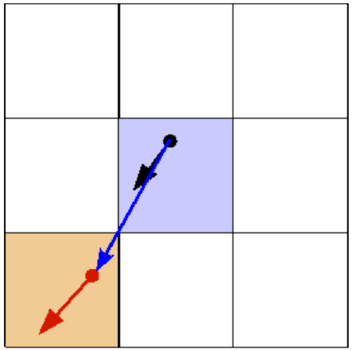
Overlay the image with a grid of equal-sized cells



Determine one or more significant *edge representers* (ER) per cell

- ER = feature, which describes the passage of an edge through the cell

Analyze 8-neighborhood of each cell to extract the connectivity

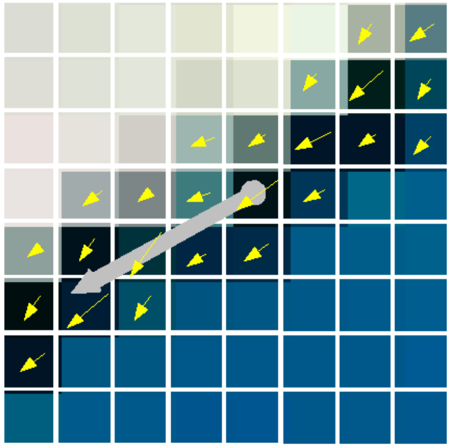


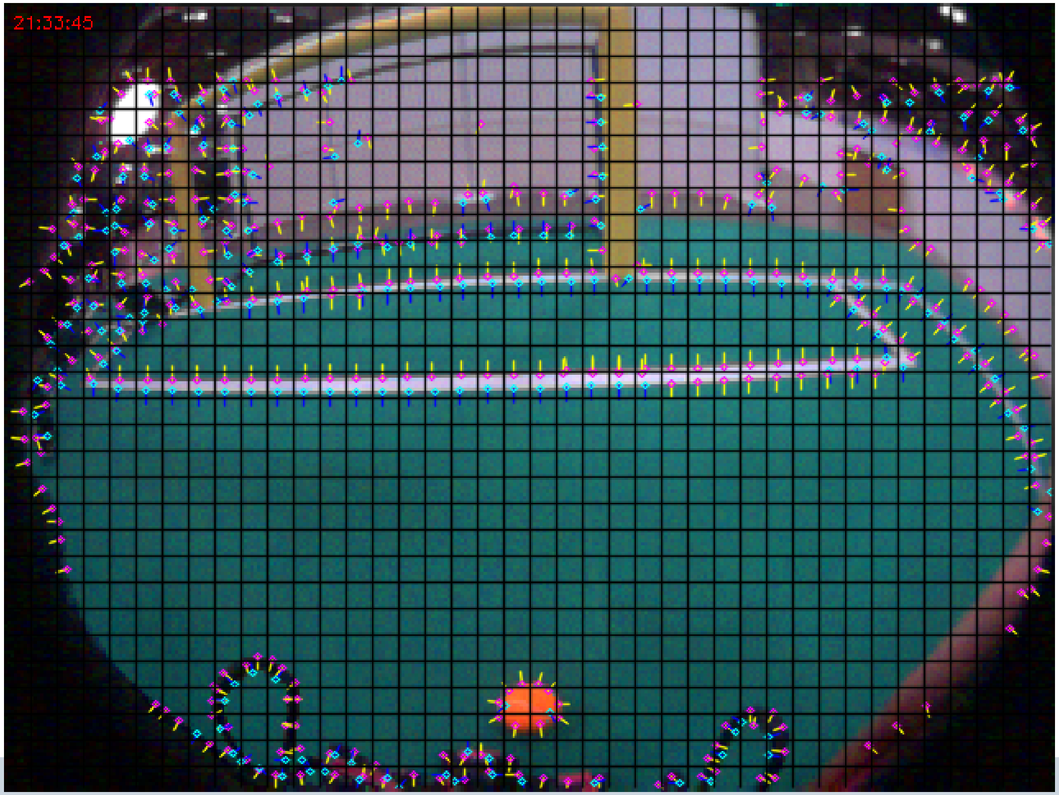
Extract edge traces out of the connectivity graph

Determination of ERs

Calculate gradient vector G foreach pixel in a cell

Accumulate each good edge pixel to that ER, where the average gradient orientation is closer then a certain distance to the current gradient orientation



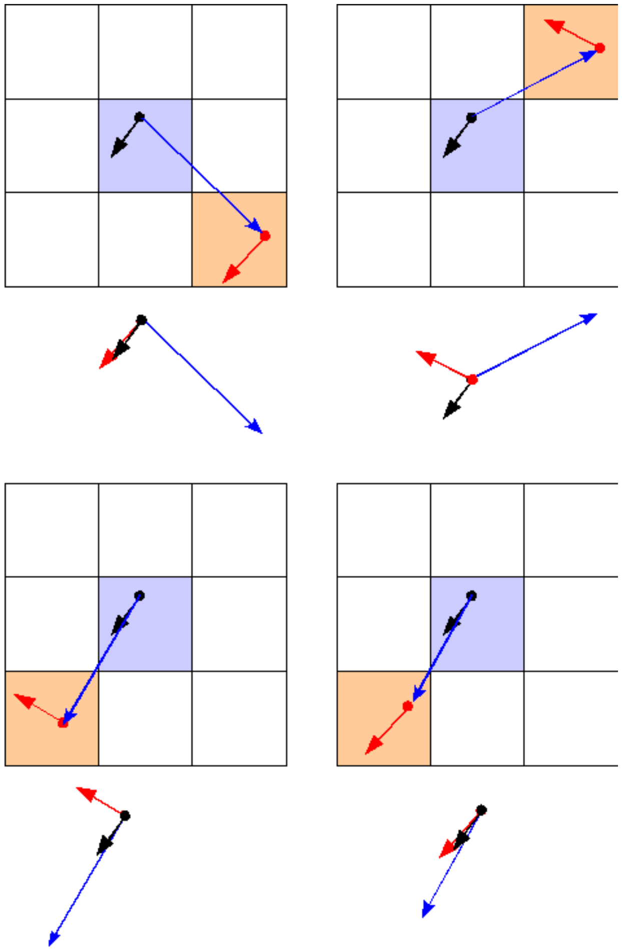


Extraction of the Connection Graph

Foreach ER (ER1):

- Foreach neighbor ER2 of ER1
  - Calculate connecting vector of ER1 and ER2
  - Connect ER1 and ER2, if orientation of connecting vector and orientation of both ERs are the same

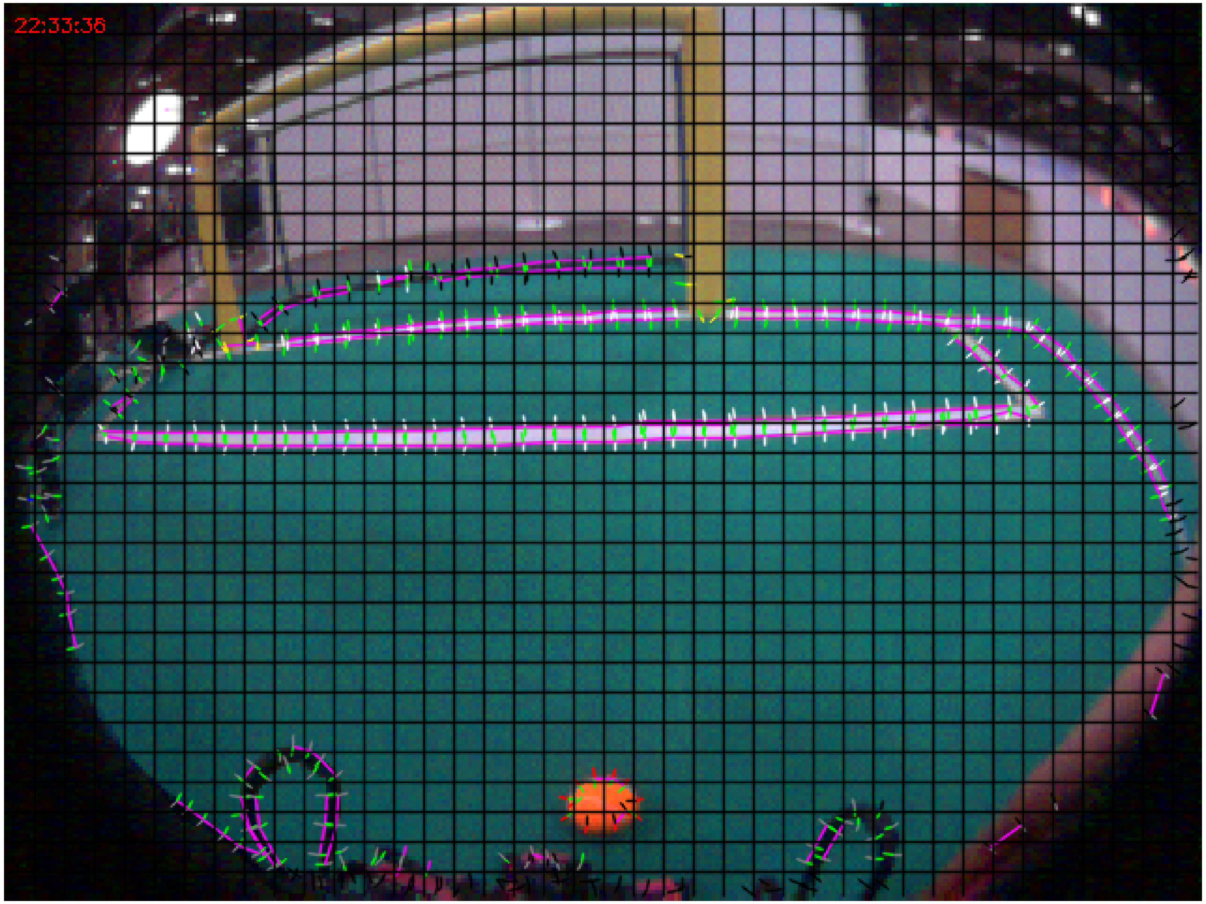
An ER can only be connected with just one neighboring ER!



Edge Trace Extraction

Iterate over all cells and search for ER with no incoming connection.

Follow the connection graph and attach all ERs on the way to one edge trace



Optimizations

Replace calculations for orientation thresholding in polar coordinates (uses atan2) with inner product thresholding in cartesian coordinates (only integer multliplications)

Skip pixels which lay above the horizon

Skip randomly some image rows

Results

Up to 17 fps on FUmanoids platform (Gumstix Verdex Pro with 600MHz PXA270)
(Compared to Canny edge detector with only 1.5 fps)

World vice-champion 2010 :-)

Thank you!