Neuromorphic Engines - project

Edge tracking using Silicon Retina sensor

presented by
Karol Hausman, Ross Kidson

Motivation

- Interactive perception
  - Robot interacts with the surroundings
  - Robot segments the objects like a child
  - Tracking
  - Clustering
  - Applications(video 1)
- Textureless objects
  - Lack of features(video 2)
  - Silicon Retina

Project Goal

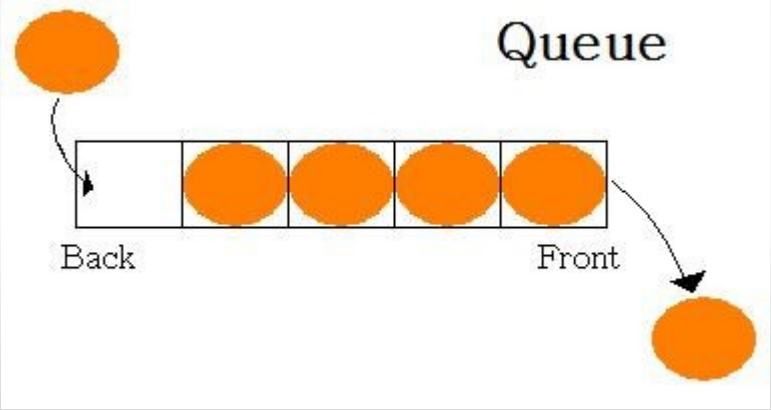
- Use Silicon Retina to track an edge of an object
- Result – one edge tracking
- Longterm result – clustering multiple edges

Project Preparation

- Line tracking is a very complex problem
- Many algorithms can be used to solve it
- How to take advantage of Silicon Retina sensor?
- How good are the results?

Dealing with Asynchronous Data

- Data is not actually asynchronous → USB polling
- One event is not enough data for known computer vision algorithms
- Small queue to process the data was implemented
- The queue is still much smaller data to process than one frame in a 'normal' camera



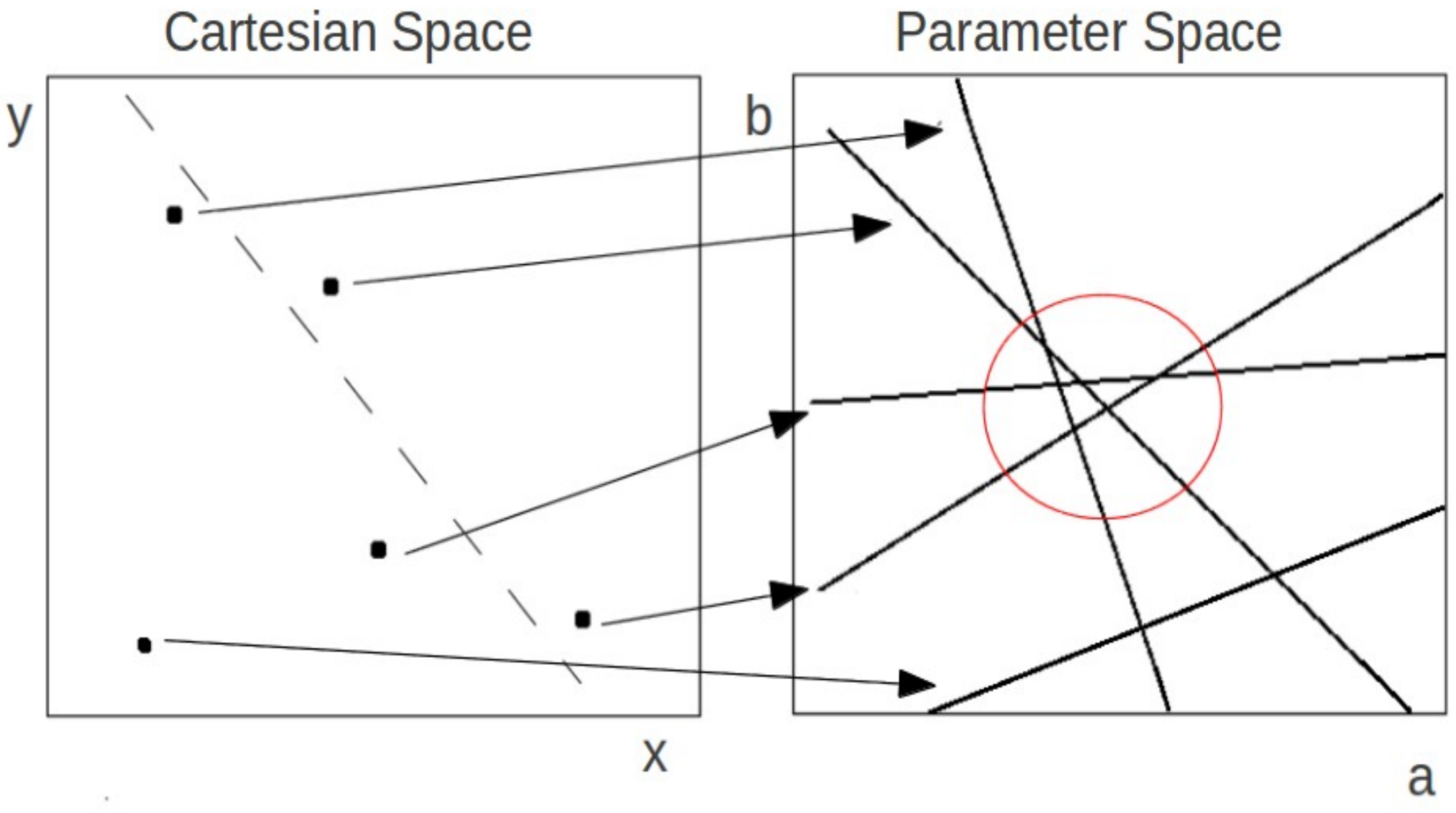
Algorithms

- Different algorithms:
  - The most obvious – calculate Hough transform for cartesian coordinates
  - The most popular – calculate Hough transform for polar coordinates
  - The most brutal – do the Ransac for polar coordinates

Hough transform - cartesian

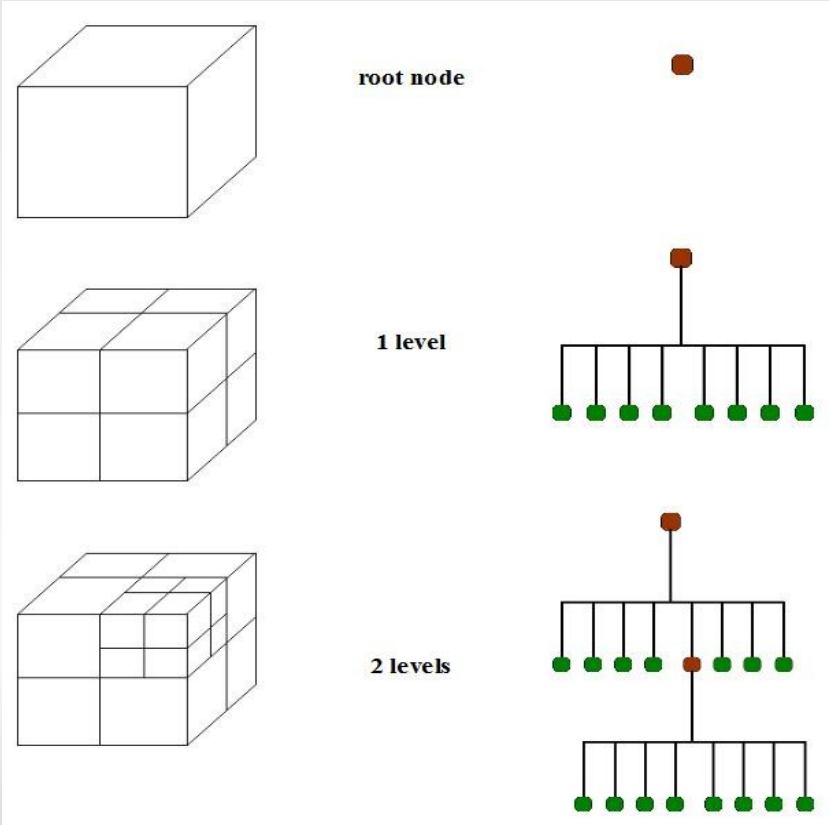
- Transfer all possible lines into parameter space (for ax+b it is a,b)
- One point in cartesian space results in one line in parameter space
- Find intersections of the resulting lines in the parameter space

Hough transform - cartesian



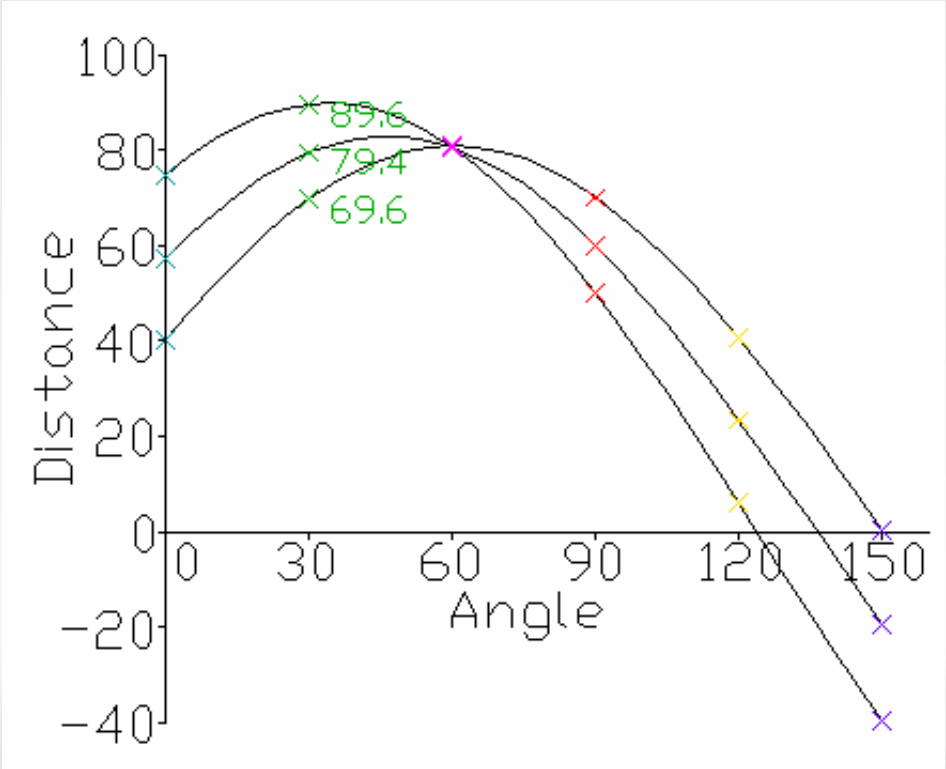
Hough transform - cartesian

- How to find the best intersection?
- Three implementations:
  - Brute force counting neighbors
  - Average intersection
  - Octree search



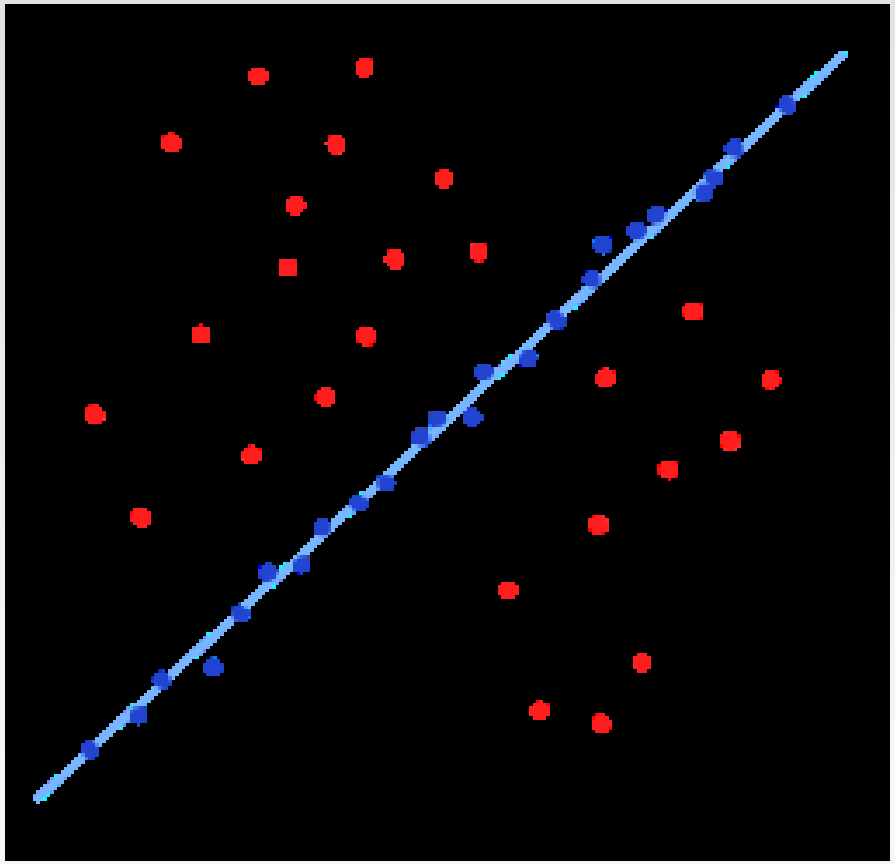
Hough transform - polar

- Transfer to polar coordinates
- Do not look for intersections → too complex
- Use a table → lines are discretized



RANSAC

- RANdom SAmple Consensus



Results

- Execution time comparison

Intersections	Hough Transform	RANSAC	*Conditions*
1255µs 35 events	**712µs** 30 events	**621µs** 150 events	*Optimal queue size*
884µs	**711µs**	**112µs**	*Same queue size* *30 events*

Results

- How does the time influence the result?
- Videos
  - Hough transform - cartesian
  - Hough transform - polar
  - RANSAC

Hough transform - cartesian

Hough transform - cartesian

- Cons:
  - Slow → number of intersections increases not linearly with the number of events
  - Cannot deal with vertical lines
  - Difficult to find intersections
- Pros:
  - :)

Hough Transform - polar

Hough Transform - polar

- Cons:
  - Accuracy depends on the resolution of the table
  - Not capable of tracking multiple lines → averaging
- Pros:
  - Problem with vertical lines solved

RANSAC

RANSAC

- Pros:
  - The fastest
  - Capable of multiple lines tracking
  - We are still in polar coordinates → vertical lines
  - Easily transferable for finding different shapes
- Cons:
  - The algorithm requires more events to work → more frame based

And the Winner is ...

- RANSAC is clearly the winner
- Is it possible to track multiple edges at the same time?
  - RANSAC modification

Multiple edges

Conclusions

- Dealing with the neuromorphic sensor and taking advantage of it is not easy
- Known algorithms have to be re-implemented
- Results are still worse than on the 'normal' camera
- Bright future?

Thank you

- Questions?