www.FUmanoids.de

The goal is to have humanoid robots which could beat the men's World Cup soccer team by 2050.

RoboCup

- chess is boring
- Goal to have "**humanoid robots which could beat the men's World Cup soccer team by 2050.**"
  - RoboCup™ (Originally called as Robot World Cup Initiative) is an international research and education initiative. It is an attempt to foster AI and intelligent robotics research by providing a standard problem where wide range of technologies can be integrated and examined, as well as being used for integrated project-oriented education.

Schach vs Fußball

Schach

- Standardproblem der KI (seit 1950)
- 1996 mit Sieg von Deep Blue gegen Kasparow
- primär Speicher/Rechenaufwand (deterministisches Spiel, vollständige Information)

Fußball

- 1995 vorgeschlagen
- dynamische Umgebung
- unvollständige Information
- umfaßt großes Spektrum an Problemen:
  - Bilderkennung (Vision)
  - Selbstlokalisierung
  - Planen / Lernen
  - Sensorik und Motorik
  - Teamkooperation

RoboCup - Leagues

RoboCupSoccer

- Simulation League
- **Small Size** Robot League
- **Middle Size** Robot League
- Four-Legged Robot League
- Standard Plattform League
- Humanoid League
  - **Kid-size**
  - Teen-size
  - Adult-size

RoboCupRescue

- Rescue Simulation League
- Rescue Robot League

RoboCup@Home

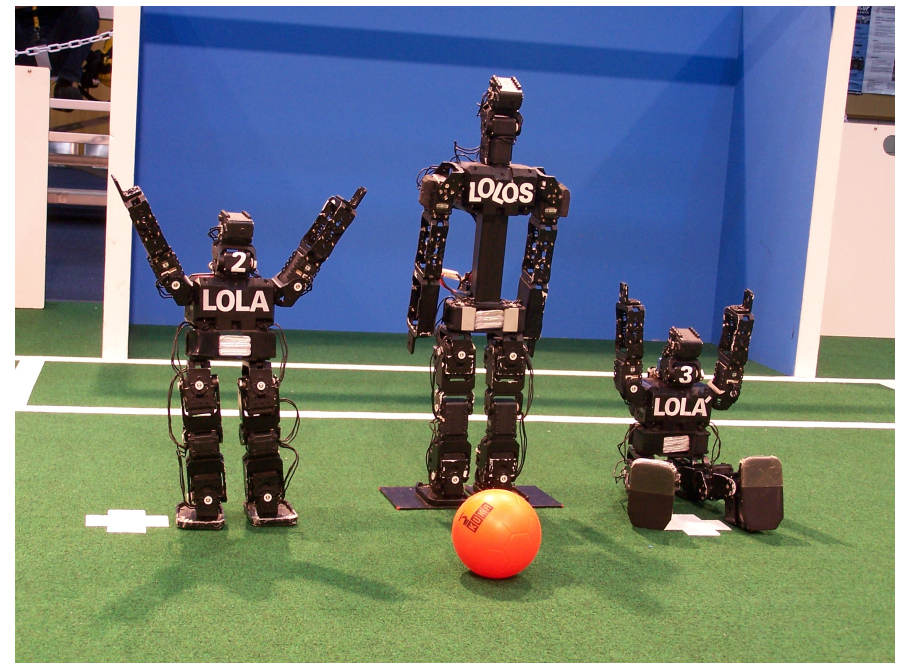
RoboCup DeMiner

RoboCupJunior

- Soccer Challenge
- Dance Challenge
- Rescue Challenge

FUmanoid History

- Successor of FU-Fighters
- FUmanoids started in 2006
- Sibling of AUTOnomous
- IranOpen: 2008: 1st, 2009: 2nd, 2010: 1st
- Worldcup: 2007: 3rd, 2009: 2nd, 2010: **???**



Team 2010

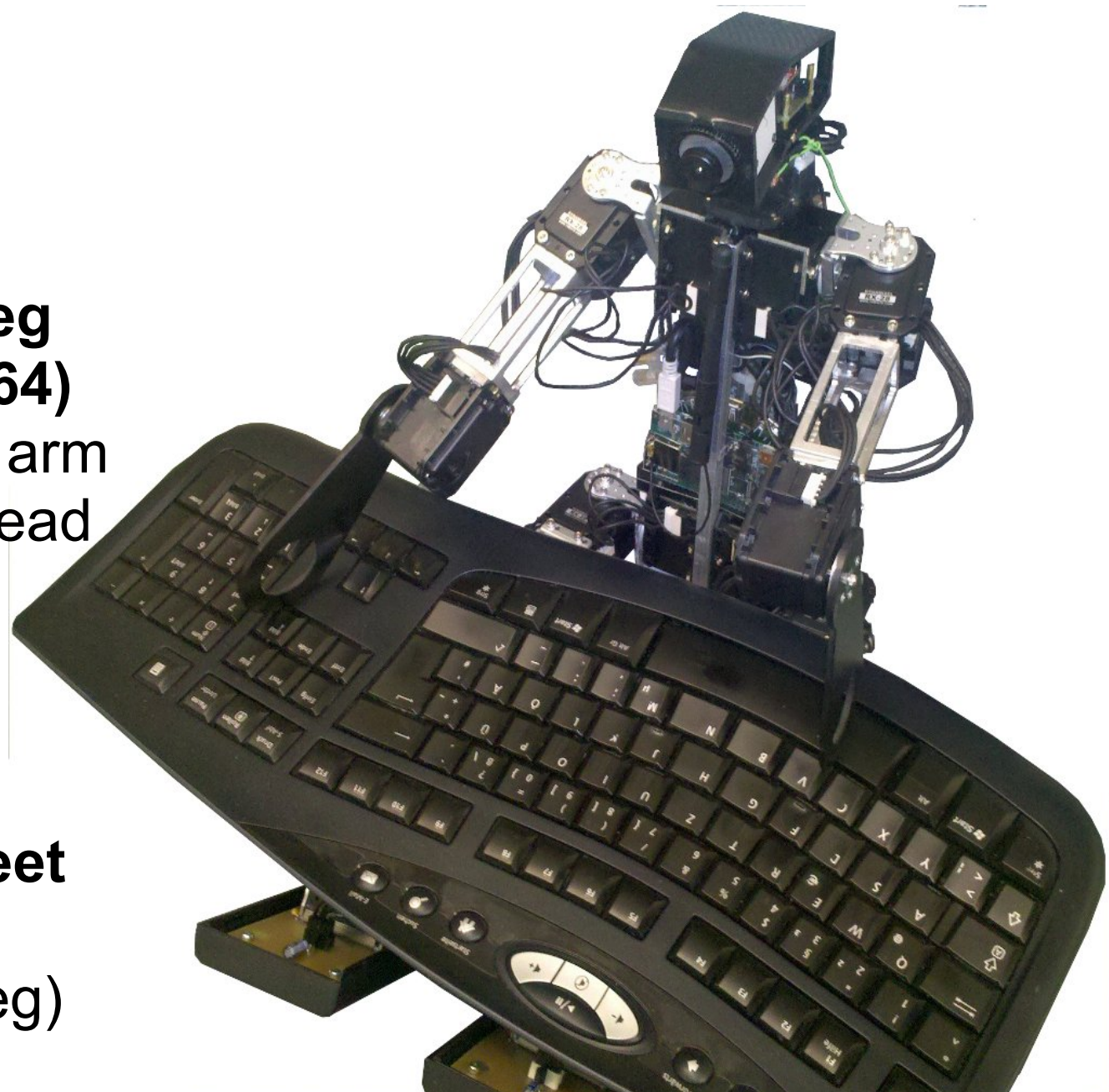
- mainly students (~10) + Hamid (teamleader)
- Bachelor, Master, Diplom, Projektarbeiten, Studienarbeiten & other stuff

Hardware 2009

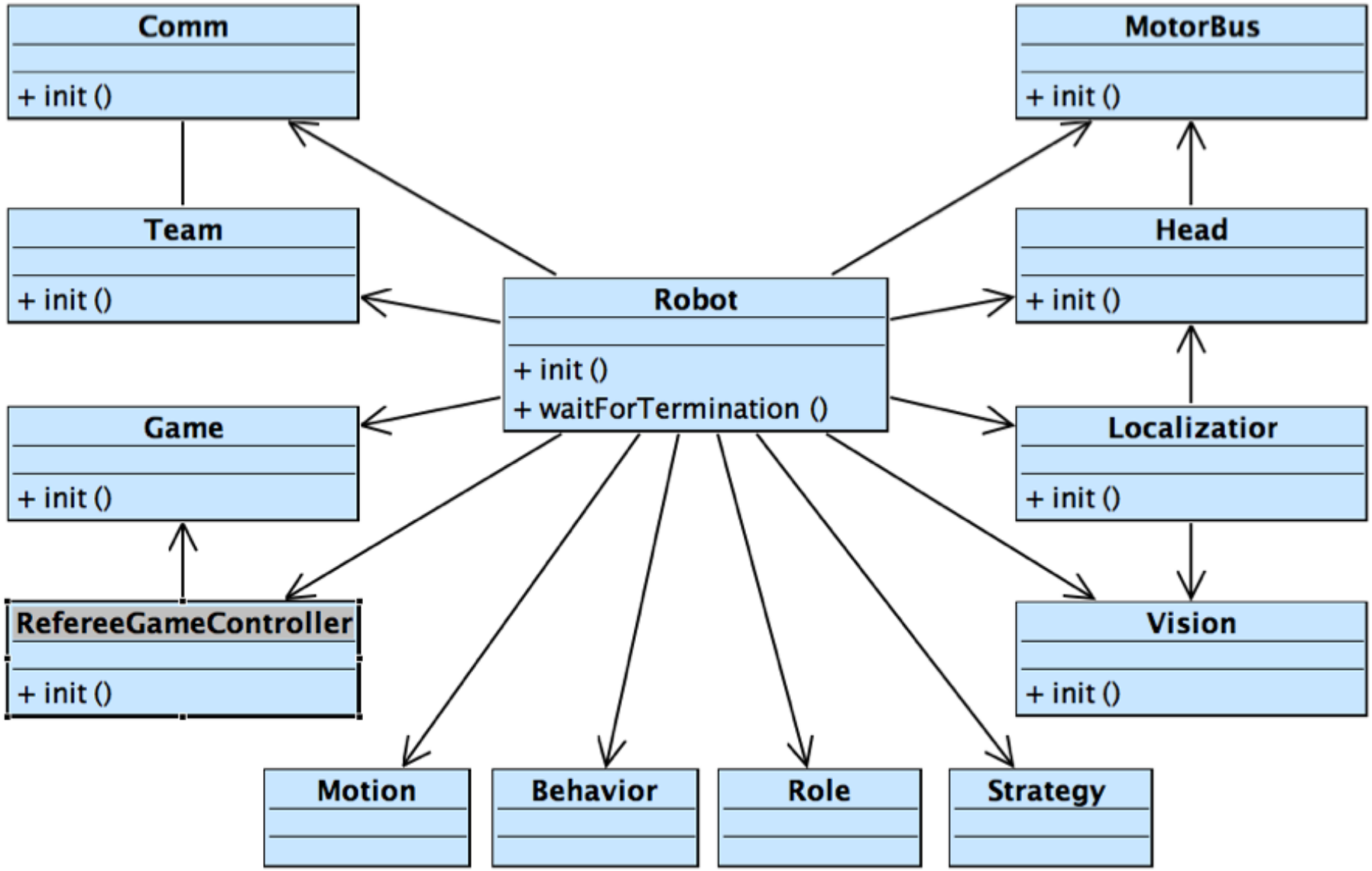
- < 60 cm
- 4.3 kg
- 21 DoF
  - 7 RX-64 in each leg
  - 3 RX-28 in each arm
  - 1 RX-28 in the head
- Gumstix Verdex
  - ARM XScale
  - 600 MHz
- Camera
  - 640x480
  - super wide angle
- (5 DoF IMU)

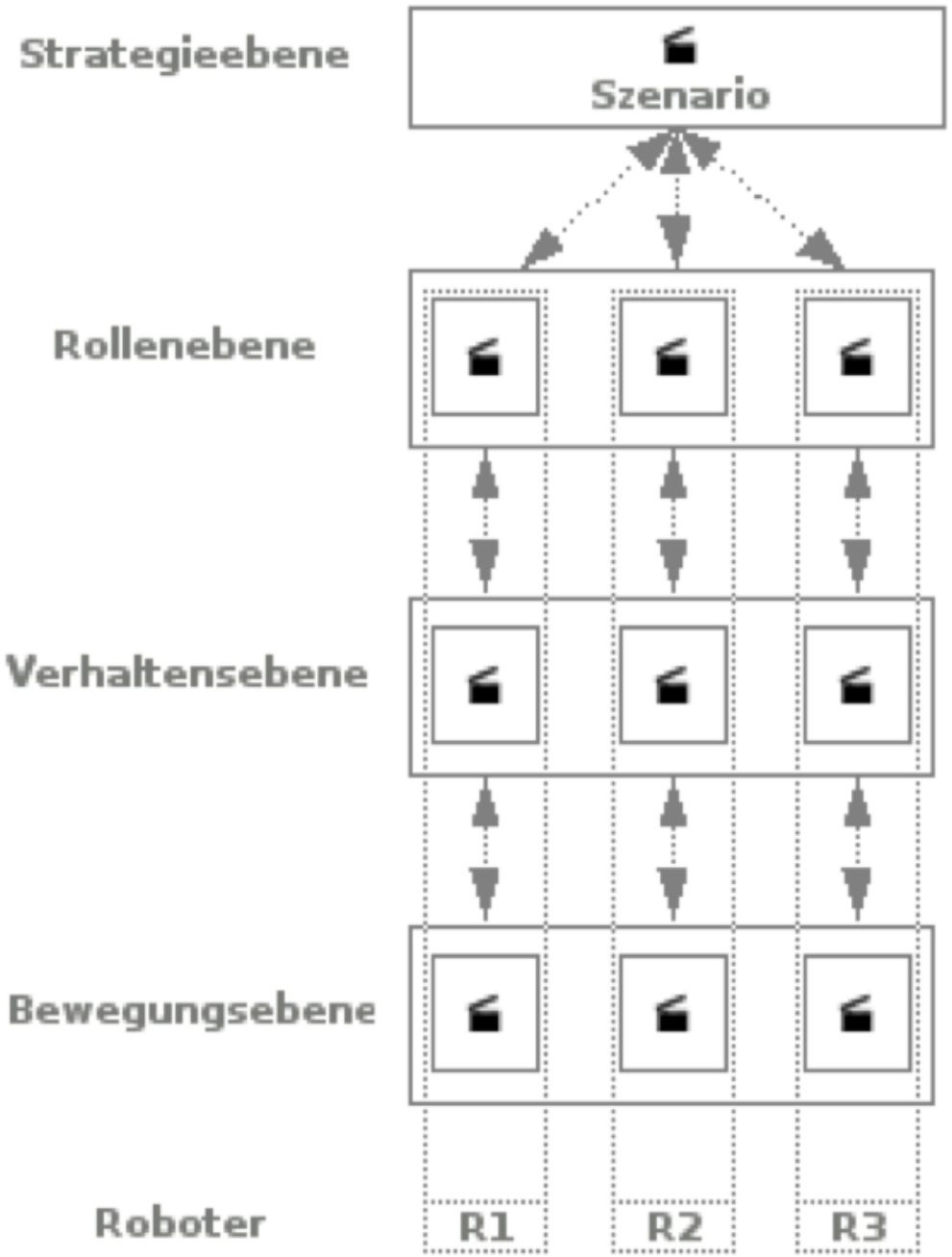
Hardware 2010

- **< 55 cm**
- **< 3 kg**
- **19 DoF**
  - **6 DoF in each leg (5 RX-28, 1 RX-64)**
  - 3 RX-28 in each arm
  - 1 RX-28 in the head
- Gumstix **Overo**
  - 600 MHz
  - ARM Cortex 8
- 5 DoF IMU
- **Tactile sensor in feet**
- **iCube Camera**
  - 640x480 (170 deg)



Architecture

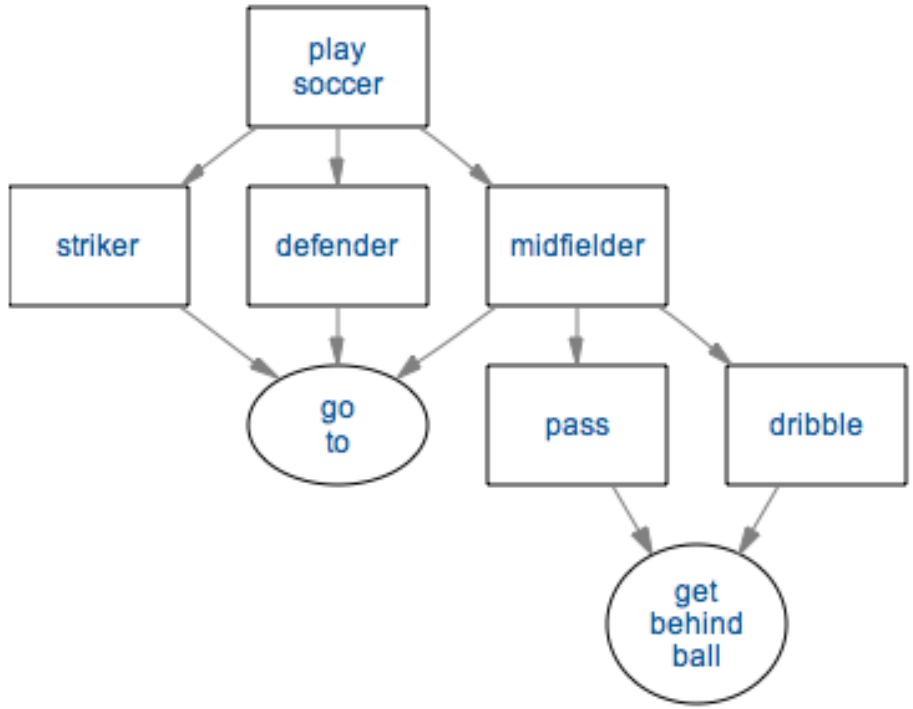




Concurrent Scenario-Based Planning (CSBP)

XABSL - Behavior Language

"The *Extensible Agent Behavior Specification Language* XABSL is a very simple language to describe behaviors for autonomous agents based on hierarchical finite state machines"

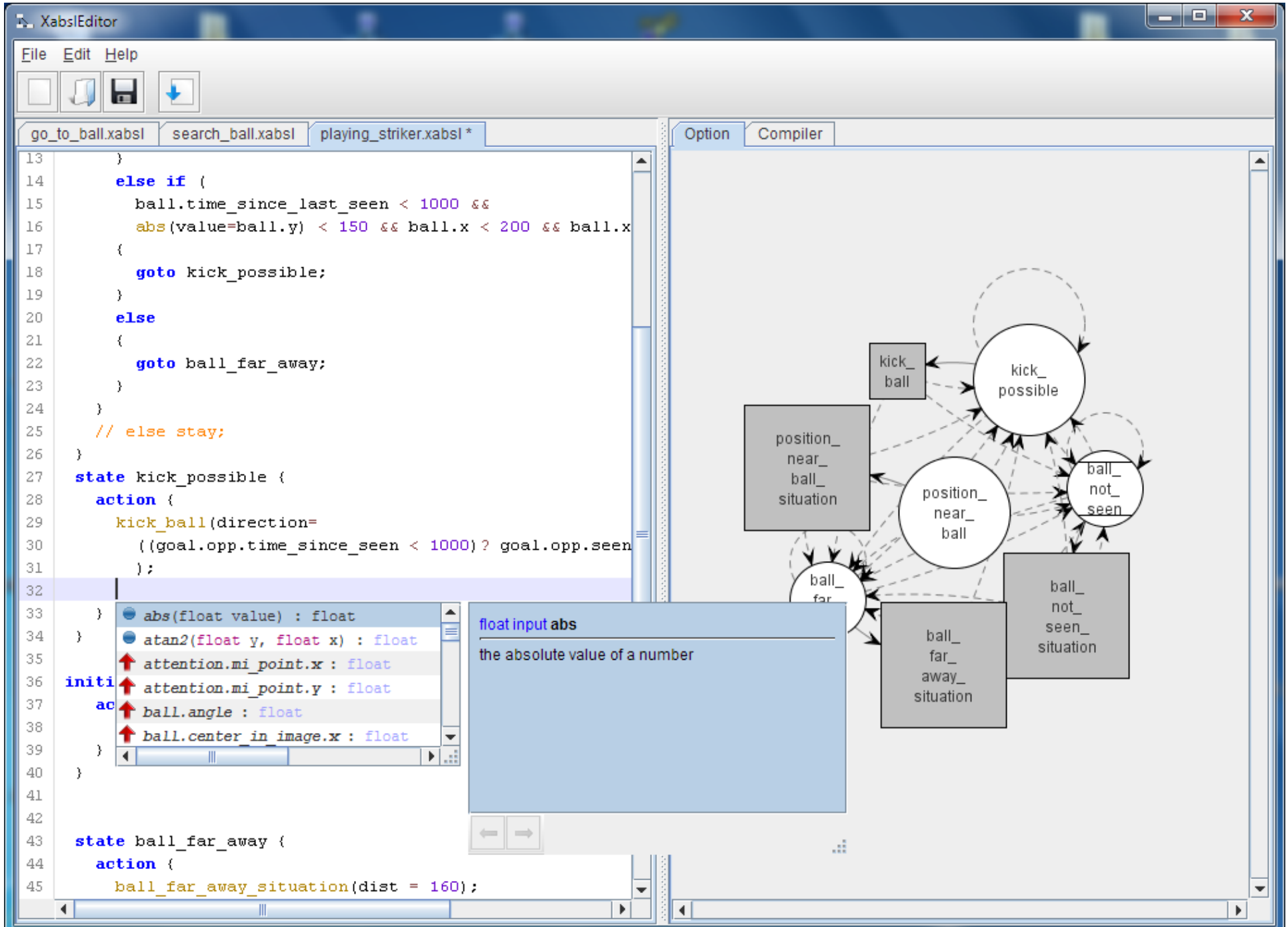


Main Parts

- Agents
- Options
- States
- Basic Behaviors
- Decision Trees

- simple behaviors possible
- more complex: http://www.robocup.informatik.tu-darmstadt.de/xabsl/examples/gt2008/options.html

XABSL - Behavior Language Editor

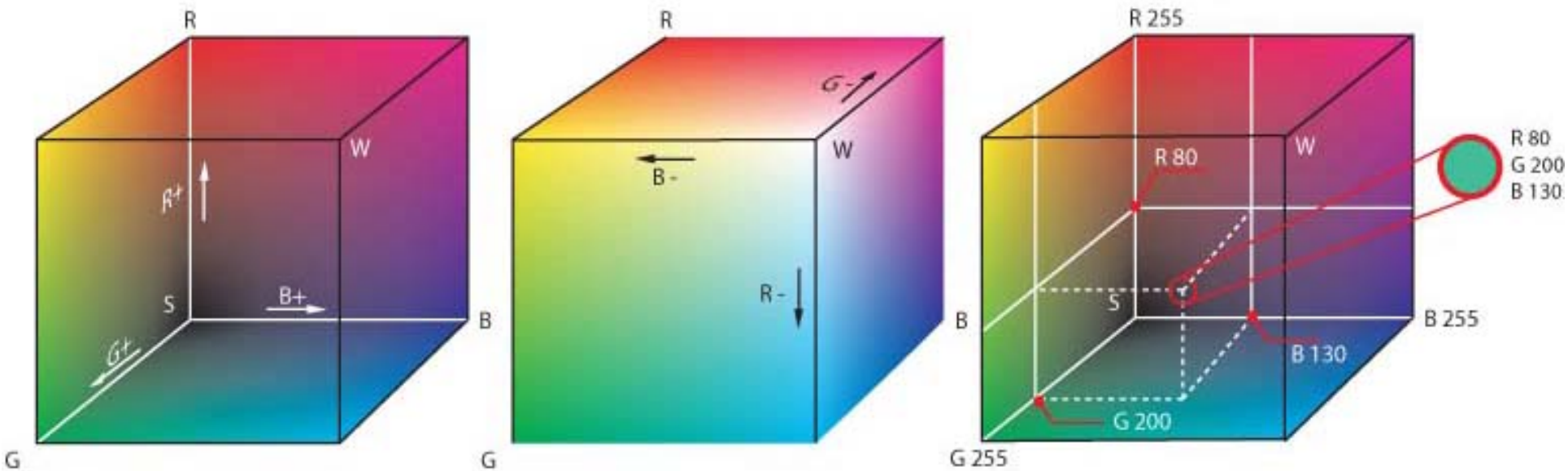


Vision

- Optische Erfassung des Spielgeschehens / Extraktion der für das Spiel relevanten Objekte (Ball, Tore, Spieler usw.)
- Verwendete Hardware
  - Eigenbau Kameraplatine mit 2 VGA-Kameras (=> Stereovision möglich, aber aktuell nicht verwendet)
  - Stattdessen eine Kamera mit ~180° Weitwinkel-Objektiv (Pro: Man sieht viel, Kontra: Linsenentzerrung notwendig)

Vision

- Für uns Menschen einfach - für die Roboter nicht. Wieso?
  - Farben sind nur Zahlen (z.B. RGB-Tupel) es gibt kein allgemein gültiges Verständnis für "grün", "blau" usw.
  - D.h.: Man muss festlegen, welche RGB-Werte als welche Farben interpretiert werden sollen (siehe auch Farbkalibrierung)



Vision

- Deshalb: Zur Vereinfachung haben alle Objekte spezifische Farben
  - Ball: Orange
  - Feld: Grün
  - Tore: Gelb und Blau
  - Seitenmarkierungen: Gelb-Blau-Gelb und Blau-Gelb-Blau
  - Teamfarben: Cyan und Magenta
  - Roboter müssen ansonsten schwarz sein
- Jedes Jahr werden allerdings die Regeln verschärft
  - von 2009 zu 2010 sind die Seitenmarkierungen verkleinert worden und Tore haben statt farbiger Rückwand nur noch farbige Pfosten und Netz

Vision

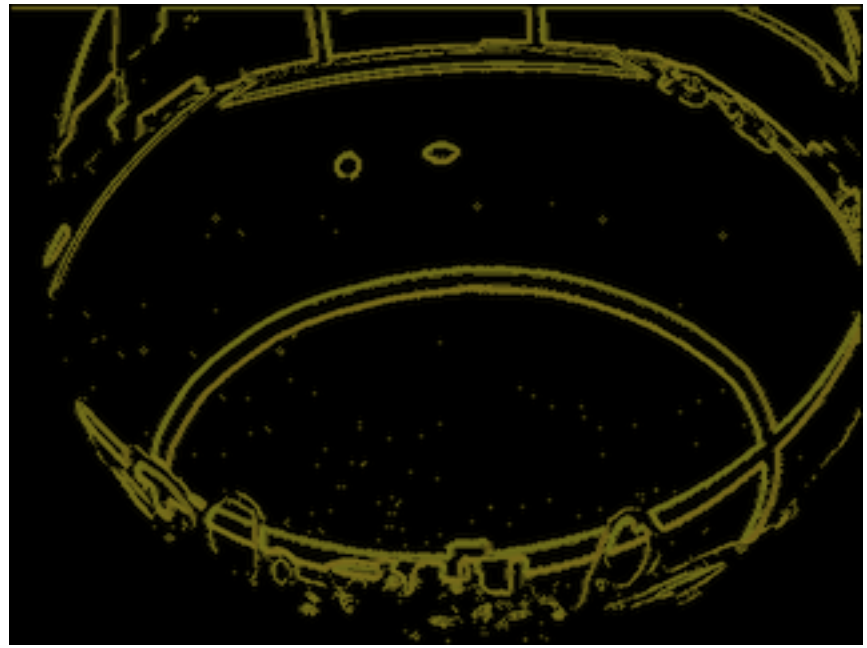
- Dieses Jahr ein kantenbasierter Ansatz gewählt, der durch Farbinformationen gestützt wird
- Was ist eine Kante?
  - Eine Kante ist genau da im Bild, wo die Differenz der Farbwerte von benachbarten Pixel hoch ist, z.B. an den Rändern der Feldlinien ist ein Grün-Weiß-Übergang
- Wozu Kanten finden?
  - Kanten entsprechen den Begrenzungen der einzelnen Objekte
  - Erster Schritt in Richtung kompletter Verzicht auf Farben und zur Objekterkennung rein durch Konturen

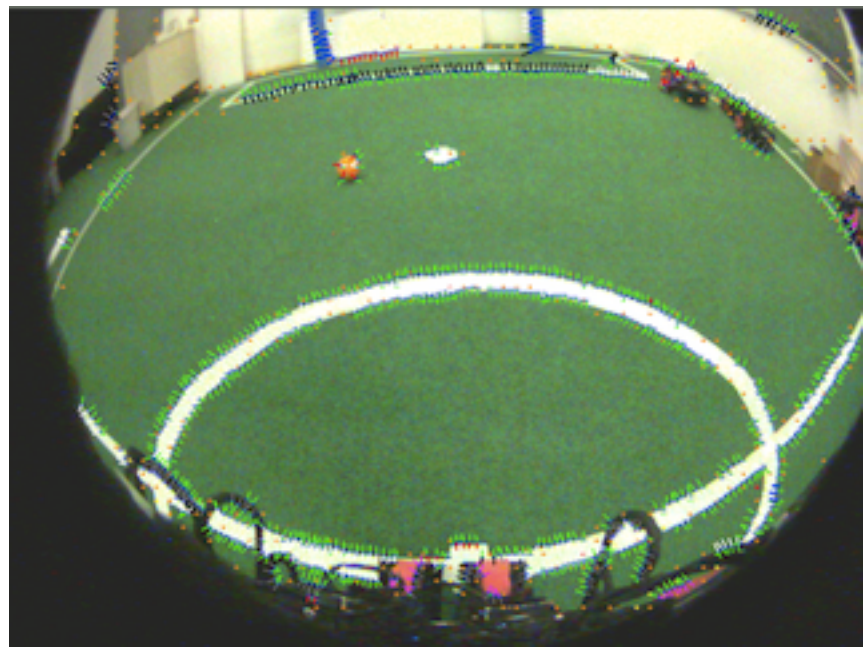
Vision

- Es gibt etliche bekannte Verfahren, um Kanten und Formen in Bildern zu Finden (z.B. Hough-Transformation,Canny-Algorithmus, Sobel-Filter usw.), teilweise in OpenCV schon vorgefertigt, warum also nicht gleich auf Farben verzichten?
- Kantenberechnungen sind *teuer* !
  - Beispiel: Der normale Sobelfilter muss zur Bestimmung der Gradientenrichtung eines Pixels acht benachbarten Pixel anfassen, eine Wurzel berechnen und atan2 berechnen
  - Nicht machbar bei 640 x 480 (~300.000) Pixeln pro Frame auf unserem System
  - Daher eigenes Verfahren entwickelt, dass nur ~10.000 Pixel anfasst

Arbeitsschritte der Vision

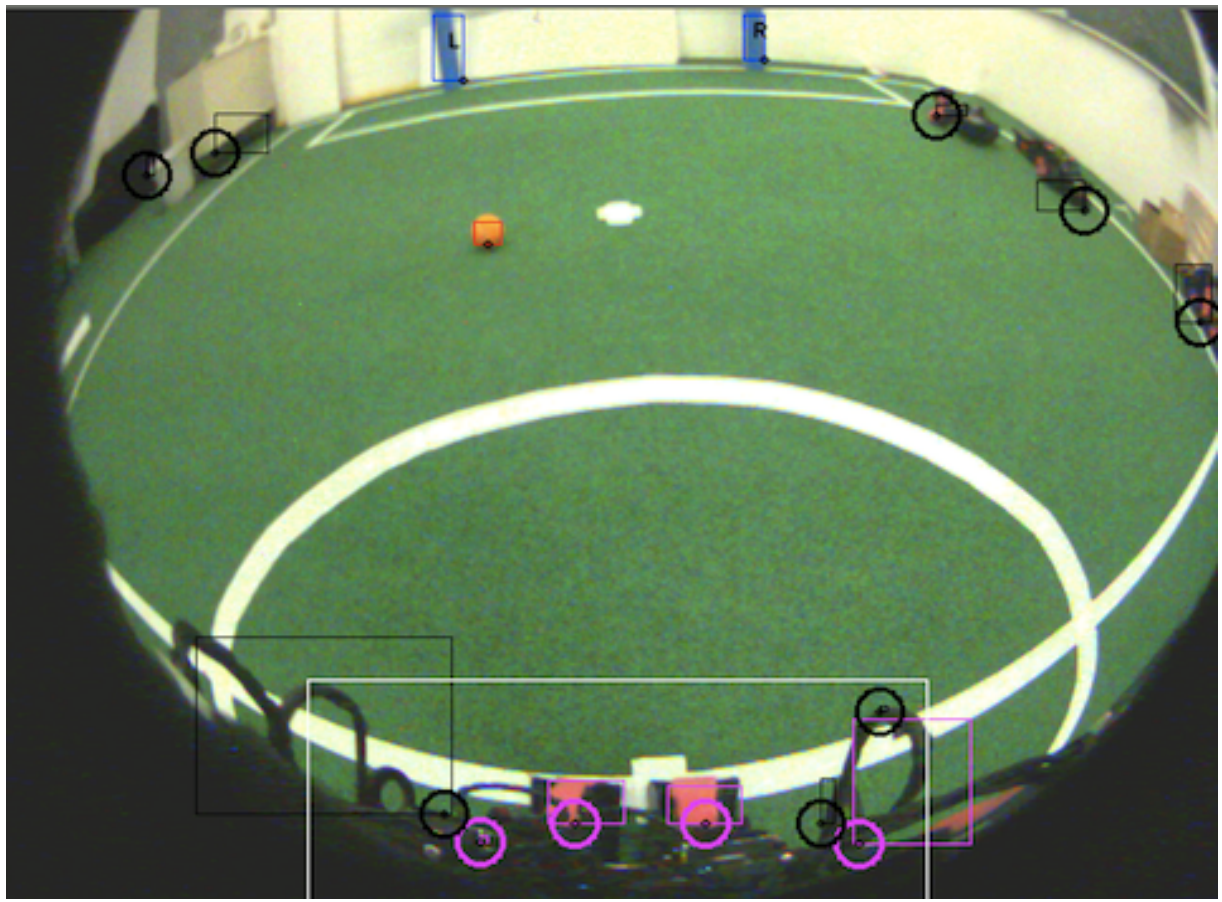
- In Miniaturbild (64 x 48) Kantenübergänge mit vereinfachtem Sobel-Filter finden ("Seedpoints")
- Im Originalbild von den gefundenen Seedpoints aus die Kante verfolgen und Linienverläufe extrahieren
- Farbinformationen auf beiden Seiten der Linienverläufe sammeln





Arbeitsschritte der Vision

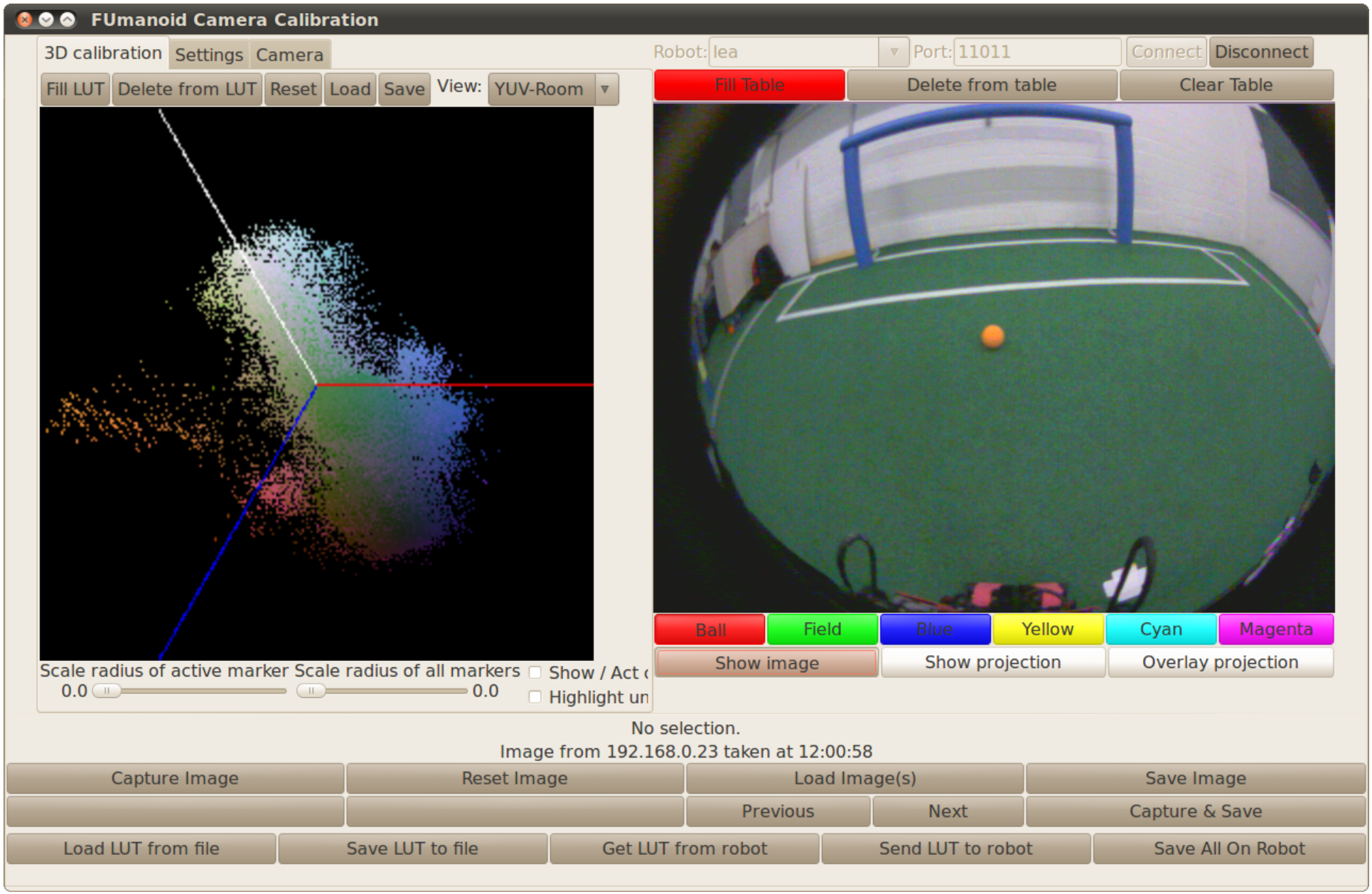
- Linien aufgrund von gesammelter Farbinformation klassifizieren (Ball-Kante, Tor-Kante usw.)
- Objekte aus den jeweils klassifizierten Kanten extrahieren



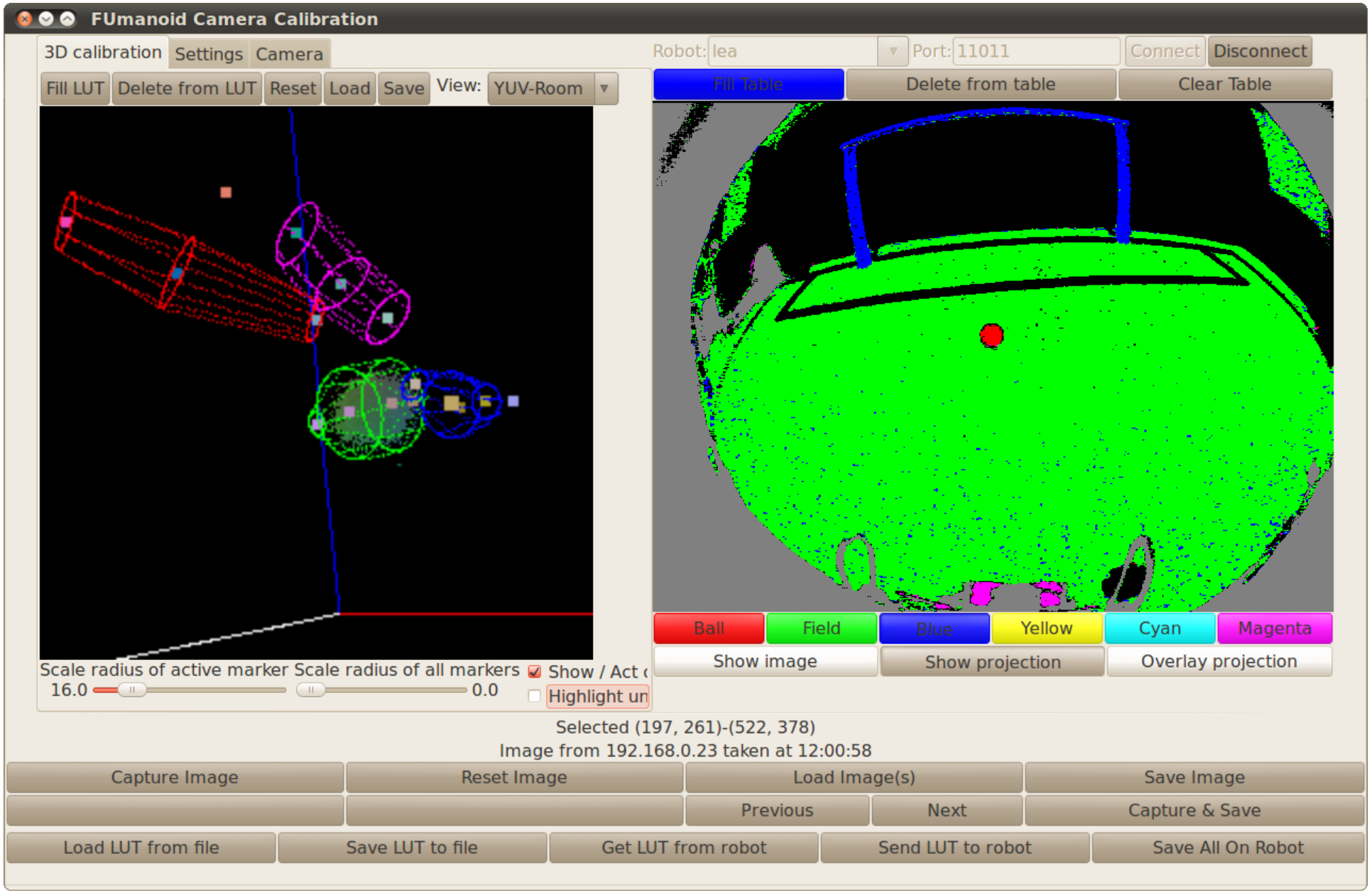
Farbkalibrierung

- Klassifizierung von Bildfarben zu Objektfarben
- nicht immer eindeutig
- lichtabhängig
- manuelle Kalibrierung z.B. pixelweise (mühsam)
- manuelle Kalibrierung über 3D-Visualisierung
  - Anzeige des Farbraumes (YUV)
  - Wahl zusammenhängender Farbwerte via Röhren

Farbkalibrierung (manuell)



Farbkalibrierung (manuell)



Farbkalibrierung (automatisch)

- Vorteil: zeitsparend, ohne menschliche Fehler
- Nachteil: Grenzfälle u.U. schwer zu entscheiden (überlappende Bereiche)
- diesjähriger Ansatz
  - Detektion von zusammenhängenden Regionen im Bild-Farbraum
  - probabilistische Zuweisung von Objektfarben
- andere Ansätze:
  - Objekterkennung und Klassifizierung anhand von Form und Lage

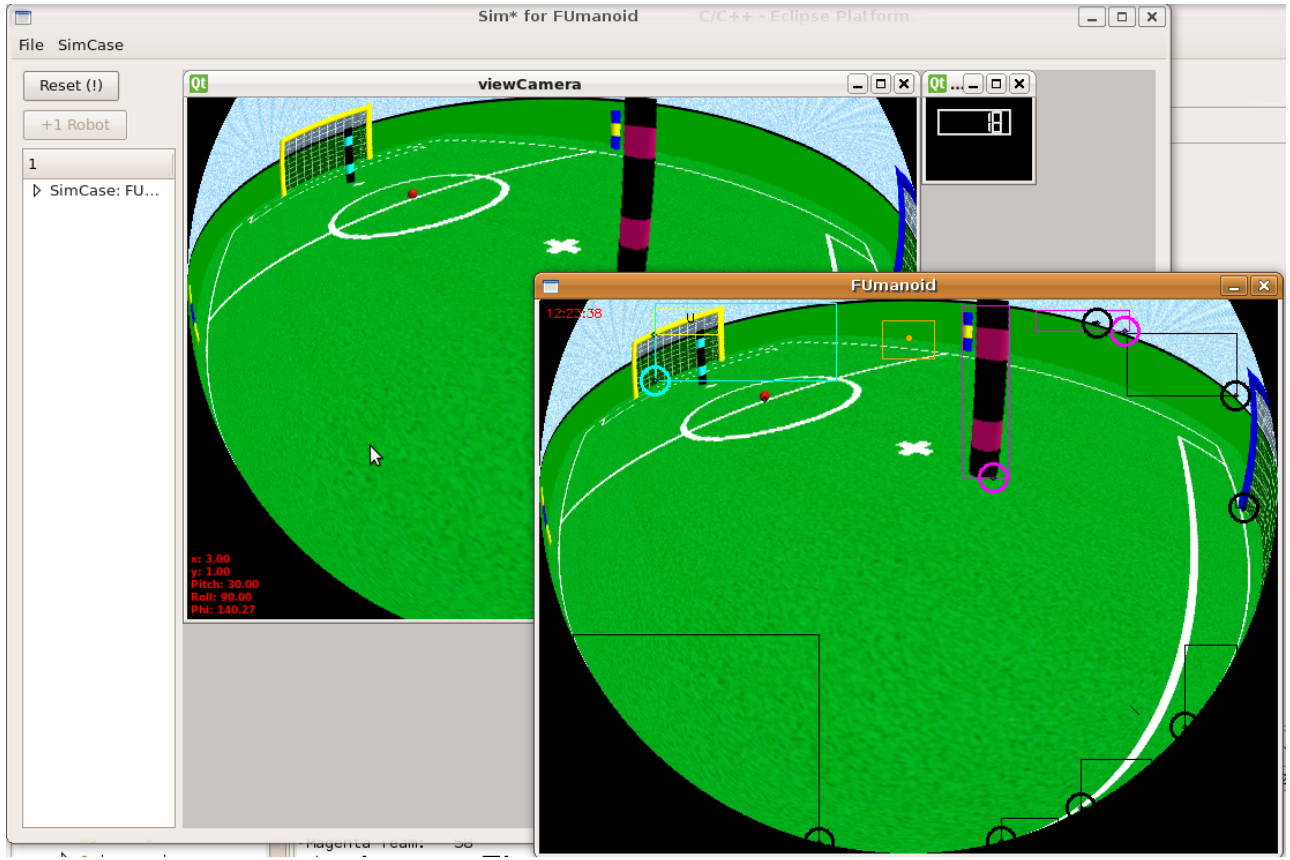
Selbstlokalisierung

- Roboter soll seine Position auf dem Spielfeld bestimmen
- notwendig z.B. für
  - Positionierung
  - Datenaustausch zu anderen Robotern ("wo ist der Ball?")
  - ...

Selbstlokalisierung

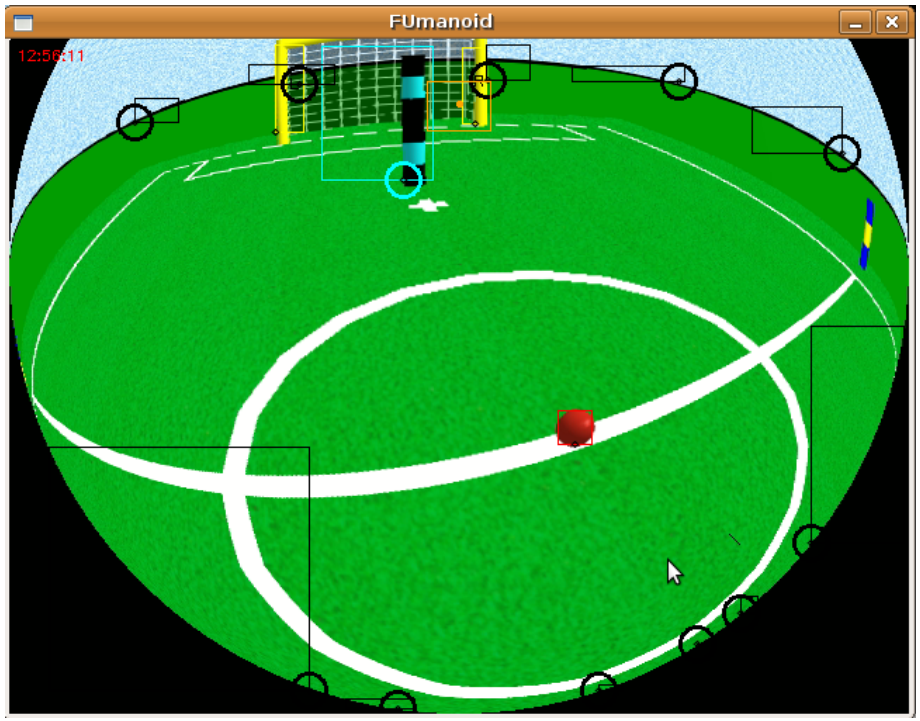
- Nutzung der gefundenen Kanten
- Bestimmung der Kreisposition im Bild
- Bestimmung gerader Linien
- Bildung von Hypothesen basierend auf gefundenem Kreis und Feldlinien
- Bewertung der Hypothesen
  - Abweichung gefundener Feldlinien
  - Abweichung gesehener Landmarken (Tore, Sidepoles)

Simulation



- **Vorteile:**
  - Vereinfachung von Problemstellungen
    - Die Realität bietet zuviele Störfaktoren für Testfälle
    - Ein Testszenario kann exakt vorbereitet werden
  - Entlastung der Hardware
    - Keine Schäden an den Robotern
    - Unbegrenzter Zugriff auf eine Testumgebung
      - sonst nur 4x Hardware für 15 Teammitglieder

Simulation



Was wird simuliert?

- Spielfeldumgebung des Roboters
- Feedback der Sensoranfragen
  - Kamerabild
  - Lage des Beschleunigungssensors
  - Servo-Motoren
- Veränderungen in der Welt
  - mit der Physikengine ODE

Was wird für die Simultion verwendet?

- OpenGL, Open Dynamic Engine, OpenCV, Qt-Framework

Heute nicht behandelt

- Walker
  - ...
- Kinematik
  - vorwärts
  - inverse
- Regelung: P, PI, PID, ...
- MicroProzessor fuer Motions
- IMU
  - Kalmanfilter
- *Linux on Gumstix*
- Reactive vs Deliberative
- Logging
- FUremote
  - Eclipse/RCP
  - Administrationstool
  - GameController / Referee
  - MotionEditor
  - usw.
- install.py
  - FU swiss army knife
- Zusammenarbeit als Team von Menschen :)
  - Redmine
  - SVN
  - Meetings

DANKE !

Mehr von uns:

- www.fumanoids.de
- twitter.com/fumanoids
- www.facebook.com/fumanoids

www.FUmanoids.de

Q&A