Vision-Based Multi-Robot Control

Mark Lee
Weier Mi
Andrew Palmer
Motivation for Multi-Robot Control

Autonomous navigation: (HW) camera, encoders, IMU, (SW) high processing

Robust, wider application vs Cost

How can we make autonomous navigation scalable?

Processing Unit (vision processing) vs Robot Platform (L,R,Straight,Stop)
System Overview

Processing Unit Benchmarks:
1. Laptop i5 (development OpenCV)
2. ARM (Baseline)
3. ARM + FPGA (Alternative)

Faster frame processing → higher frequency of robot commands (RT)
SDSoC Setup

ARM and FPGA interface (VDMA)

**Code Structure:**

**Src:** Top level code `main.c`
- Links virtual memory to physical memory space video
- Setup TCP network
- Calls series of image processing functions
- Receives Position, Orientation → determine robot commands
- Broadcast robot commands

**Hw:** Image functions to be synthesis for accelerator
- RGB2YCBCR
- Threshold by Color
- Median filter
- Center of Mass
- Corner Detection
Network

- Socket Programming
- TCP/IP
- Server and Client
- Functions

Picture credit: https://www.codeproject.com/articles/586000/networking-and-socket-programming-tutorial-in-c
Image Processing Pipeline

- RGB to YCbCr
- Color Threshold
- Median Filter
- Center of Mass
- Corner Detect
- YCbCr to RGB
RGB to YCbCr Conversion

- Convert each pixel to YCbCr format for later image processing stages
- Rationale: YCbCr more effective for color differentiation
  - RGB colors are affected by illumination variation
  - YCbCr separates luminance from colors
  - HD (1080p) is encoded in YCbCr

\[
\begin{bmatrix}
Y \\
Cb \\
Cr
\end{bmatrix} = \begin{bmatrix}
16 \\
128 \\
128
\end{bmatrix} + \begin{bmatrix}
0.183 & 0.614 & 0.062 \\
-0.101 & -0.339 & 0.439 \\
0.439 & -0.399 & -0.040
\end{bmatrix} \cdot \begin{bmatrix}
R \\
G \\
B
\end{bmatrix}
\]

Ranges:
- R/G/B [0 ... 255]
- Y [16 ... 235]
- Cb/Cr [16 ... 240]

RGB to YCbCr color conversion for HDTV
Color Threshold

- Goal: Differentiate between Red Robot, Blue Robot, and Green Goal
- Detect different colors based on Y, Cb, and Cr values
- Set each object’s pixels to be distinguishable colors
Median Filter

- Remove noise
  - Outliers
- Preserve Edges

\[
\begin{array}{ccc}
5 & 3 & 4 \\
3 & 10 & 5 \\
3 & 4 & 5 \\
\end{array}
\]

\[
\begin{array}{ccc}
5 & 3 & 4 \\
3 & 4 & 5 \\
3 & 4 & 5 \\
\end{array}
\]
Line Buffers in Median Filter

Pipeline II = 1

FIFO interface

ap_window (9x9) and 8 ap_linebuffer data types

Shift line buffer & window → read only the very bottom pixel
Center of Mass

- Locate objects within the scene
- Calculate image moments for pixel values of red, blue, and green objects
- Center of mass coordinates \( \bar{x}, \bar{y} \), calculated from moments for each object

\[
\begin{align*}
\bar{x} &= \frac{M_{10}}{M_{00}} \\
\bar{y} &= \frac{M_{01}}{M_{00}} \\
M_{p,q} &= \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} x^p y^q g(x, y)
\end{align*}
\]

Red Robot Location
Blue Robot Location
Green Goal Location
Corner Detection

- Finds coordinates with minimal/maximal x/y values that belong to an object
- Temporal median filtering
- Only works for objects with four corners (which is our case)!
Robot Commands
SW Baseline Demo

- Demo
# Evaluation

<table>
<thead>
<tr>
<th>Application</th>
<th>SW Time (cycles)</th>
<th>HW Time (cycles)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>COM</td>
<td>1.8B</td>
<td>0.1B</td>
<td>20x</td>
</tr>
<tr>
<td>COM + Corner</td>
<td>16B</td>
<td>0.16B</td>
<td>100x</td>
</tr>
</tbody>
</table>