A Subsumptive, Hierarchical, and Distributed Vision-Based Architecture for Smart Robotics

Vision-Guided and Intelligent Robotics Lab - ViGIR

Guilherme DeSouza
Introduction

- Vision and Automation
  - Automation vs. robotics constraints
  - Powerful computers vs. more demanding tasks
Introduction (cont.)

- Vision-Based Architectures
- Horizontal Layers (Brooks-86)
- Top-Down vs. Bottom-up

Hierarchical vs. "incremental" Subsumption
- Interaction between behaviors (Vectors, Fuzzy,...)
- TCA (Simmons-94), DAMN (Rosenblatt-95)

- Advantages and Disadvantages
  - "Intractable" (Tsotsos-95)
  - Fault tolerance vs. All behaviors must be active
Introduction (cont.)

- Traditional Control of Industrial Processes

SCADA Systems
Distribution => Redundancy => Fault Tolerance

- Vision-Based Control

Failures represent large financial costs
Fault Tolerance?
Introduction (cont.)

- Proposed Vision-based Control Architecture
  - Subsumption
  - Hierarchical
  - Distributed
    - CPU-Efficient
    - Redundant
    - Fault-tolerant
Introduction (cont.)

- Line Tracking
  - Visual servoing for Assembly of automotive parts on-the-fly
  - Subsumptive tasks

- Mobile Robotics
  - Visual servoing of the robot w.r.t. Multiple cues
  - Map-based with localization by tracking natural landmarks
Agenda

- Introduction
- Visual Servoing
- The New Software Architecture
- Implemented Visual Servos
  - Camera, Hand-Eye, and Head-Eye Calibrations
- Applications of the Software Architecture
  - Line Tracking
  - Mobile Robot Navigation
- Results and Conclusions
Agenda

- Introduction
- Visual Servoing
- The New Software Architecture
- Implemented Visual Servos
  - Camera, Hand-Eye, and Head-Eye Calibrations
- Applications of the Software Architecture
  - Line Tracking
  - Mobile Robot Navigation
- Results and Conclusions
Visual Servoing

- Closed-loop Control Systems that uses cameras as sensing devices. Its task is to position a robot with respect to a target object

- Task Space
  - 2DOF (bin-picking, pick-and-place, etc.)
  - 3DOF (gaze control, mobile robots)
  - 6DOF (assembly of parts)
Classification

- Position of the Cameras
  - Eye-in-hand (Calibration)
  - Fixed-mounted
  - Combination

- Extracted Feature Points
  - Endpoint Open-Loop (EOL) - target object
  - Endpoint Closed-Loop (ECL) - target and end-effector

- Hierarchy of the Control
  - Dynamic look-and-move
  - Direct servo

- Space of the Error Function
  - Position-based
  - Image-based
Position-Based Direct Servo

X* +
\[ X \]
- \[ \_ \] \[ \times \]
Cartesian Control Law

Power Amplifiers

Pose Estimation

Image Feature Extraction

X
Y
Position-Based Look-and-Move

Cartesian Control Law → Joint Control → Power Amplifiers → Image Feature Extraction

Pose Estimation

$X^*$ → $X$ → $Y$
Image-Based Direct Servo

\[ Y^* \rightarrow + \rightarrow \text{Feature space Control Law} \rightarrow X \rightarrow \text{Power Amplifiers} \rightarrow \text{Image Feature Extraction} \rightarrow Y \]
Image-Based Look-and-Move
Discussion

- **Position-Based**
  - Accuracy is highly dependent on Calibration
    - Hand-Eye and Head-Eye calibrations
  - Feature Extraction and 3D Reconstruction
    - Stereo, Model-based, Structure from Motion (Huang94)

- **Image-Based**
  - Non-linear Mapping from Task Space to Feature Space (Large dimension feature space)
    - Easy to find for Fixed-mounted cameras (Sanderson87)
    - Approximate using Jacobian
    - Difficult inverse, not stable, convergence.
Observations drawn from prior work

- Position-based is usually preferred
  - Specially for moving targets (same space)

- Visual-servoing systems are closed-loop, discrete time, dynamic systems
  - Sampling rate impacts the performance, convergence, and effectiveness
  - Time delays add even more latency
Observations drawn from prior work (cont.)

- Requisites for Visual-servoing
  - Fast sampling rates
  - Robust
  - Redundancy (multiple cues)
  - Accurate (Calibration)
  - CPU-efficient (minimize delays)
  - Fault-tolerant
  - Modular, Flexible, etc...
Agenda

- Introduction
- Visual Servoing
- The New Software Architecture
- Implemented Visual Servos
  - Camera, Hand-Eye, and Head-Eye Calibrations
- Applications of the Software Architecture
  - Line Tracking
  - Mobile Robot Navigation
- Results and Conclusions
Control Software Architecture

- Client/Server Infrastructure
  - Resources
  - Processes and Threads
  - Client/Server
  - Wrappers
  - Pipes/Sockets
  - Shared Memory
  - Modularity/Encapsulation
Wrappers in a Client-Server Infrastructure
Advantages of the Software Architecture

- Fault-tolerance
  - Redundant modules, subsumption
- CPU-Efficiency
  - Distribution
- Elasticity
  - Dynamic creation, deletion
- Flexibility
  - No effect of redesign of modules
Primary Characteristics of the Architecture

- Multi-loop Control Architecture
  - Independent parallel loops
  - Arbitration based on 'de-coupled' information from loops (confidence levels)
  - Modules can be subsumed or replicated
  - Slow modules do not affect others
  - Communication vs. Application threads
Notation

- Quaternions
  \[ q = (s, v) \text{ where } s = \cos(\theta/2), \quad v = \sin(\theta/2) \times e \]

- HTM
  - Denavit-Hartemberg Notation:

- Camera model
  - Pin-hole with Radial Distortion
  - \( Cal = Int \times Ext \) (\( Ext \) is a HTM)

- 3D Reconstruction
  - Stacking two calibration matrices (Least Square)
  - Object pose from cross-product of 3 feature points

- XYZOAT Representation

\[
^w H_b = \begin{bmatrix}
R_{3x3} & T^t \\
\tilde{0}^t & 1
\end{bmatrix}
\]
Position-Based Visual Servoing

- **Position task:** $E: T \rightarrow m$

- **Kinematics Error Function** = maps the task space $T$ (all poses of the end-effector) into an error vector (e.g. XYZOAT)

- Usually, one controller for each degree of freedom

- **Use HTMs**

- **Error Function:**

$$E \mathbf{T} e = e H_o - e H^*_o$$

$$E \mathbf{T} e = e H_c \mathbf{c} H_o - e H^*_o$$
Position-Based Visual Servoing

- Difference (?!):
  \[ E_T^e = ^eH_o ^oH^*_e \]

- Proportional Controller (Quaternions)
  \[ u = k_p ^eH^*_e \]
Image-Based Visual Servoing

- **Position Task:** $E: f(T) \rightarrow \mathbb{R}^k$
  
  - Non-linear mapping from task space $T$ into the $k$-dimensional feature vector $Y$
  
  $$Y = f(T)$$

- **Linearization**
  
  $$\partial Y = J \partial T$$
Image-Based Visual Servoing

- Error Function
  \[ E \cdot T_e = J^{-p} \cdot \nabla Y \]

- Proportional Controller
  \[ u = k_p \cdot J^{-p} \cdot \nabla Y \]

- Update Jacobian
  \[ J_{k+1} = J_k \cdot \frac{\nabla Y - J_k \cdot \nabla T \cdot \nabla T^t}{\nabla T^t \cdot \nabla T} \]
Agenda

- Introduction
- Visual Servoing
- The New Software Architecture
- Implemented Visual Servos
  - Camera, Hand-Eye, and Head-Eye Calibrations
- Applications of the Software Architecture
  - Line Tracking
  - Mobile Robot Navigation
- Results and Conclusions
Agenda

- Introduction
- Visual Servoing
- The New Software Architecture
- Implemented Visual Servos
  - Camera, Hand-Eye, and Head-Eye Calibrations
- Applications of the Software Architecture
  - Line Tracking
  - Mobile Robot Navigation
- Results and Conclusions
Recall

- **Position-Based**
  - Accuracy is highly dependent on Calibration
  - Hand-Eye and Head-Eye calibrations
  - Feature Extraction and 3D Reconstruction
  - Stereo, Model-based, Structure from Motion (Huang94)
Camera Calibration

- Intrinsic and Extrinsic parameters
  - Intrinsic parameters relate the *standard coordinate system* and the image coordinate system (pixels)
  - Extrinsic parameters relate a world coordinate system and the standard coordinate system

\[
\begin{bmatrix}
U \\
V \\
S
\end{bmatrix}
= \begin{bmatrix}
\uparrow fk_u & 0 & u_0 & 0 \\
0 & \uparrow fk_v & v_0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
\sim R \\
\sim T \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\]
Other Calibrations

Hand-eye and Head-eye Calibrations
Necessary for eye-in-hand configurations

No “world” is available:

\[ AX = XB \] Problem (Shiu & Ahmad)

- \( X \) is the desired calibration matrix
- and \( A, B \) are known matrices: extrinsic parameters
  of the camera and kinematics of the end-effector

(DeSouza et al, ICRA2001 & ICRA2002)
Hand-Eye Calibration

(ICRA2001)
Hand-Eye Calibration

Calibrate Hand-Eye and World-Base

\[(AX = YB)\]

Iterative Algorithm:

Find n equations such as:

\[A_i * e_i H_{c_i} = b H_w * B_i\]

Solve for: \[e_i H_{c_i}\]

"Average" the n solutions to find: \[e H_c\]

Use to solve:

\[A_i * e H_c = b_i H_{w_i} * B_i\]

"Average" the n solutions to find: \[b H_w\]

Repeat until it converges
Hand-eye Calibration

"Average" using quaternions for rotation part, and normal vector addition for the translation part.

Proof of convergence

At each step, the averaged "error quaternion" has a larger s-component

As the s-component grows towards 1 (unit vectors), the error quaternion becomes the "identity quaternion"
Hand-eye Calibration Test without Noise
Hand-eye Calibration
Test with Noise
Hand-eye Calibration

Results:

Less than 0.01 radians (0.3 degrees)

Less than 0.001 meters
Head-Eye Calibration (ICRA2002)

Goal: to calibrate mobile robot's stereo head without worrying about its pose

Again, to calibrate means:

- camera calibration (two pairs of Intrinsic and Extrinsic matrices)
- head-eye/neck-eye calibrations (four HTM matrices)
Stereo Head

3 DOF: pan, tilt, and vergence
Stereo Head - HTMs

One extra HTM: \( ^cH_e \) and \( ^E H_b \)

Independently of the world coordinate system
Calibration Procedure

Fix pan and tilt
Choose values for vergence $v_i$
Obtain corresponding $c_i H_w$
Equate and solve

$$A_{ij} c H_e = c H_e B_{ij}$$
Test A
Results for Test A

Error in XYZ-Distance for objects @1,200mm

Various pan/tilt combinations (-10°, 10°)

<table>
<thead>
<tr>
<th></th>
<th>dX</th>
<th>dY</th>
<th>dZ</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>0.4530</td>
<td>1.4530</td>
<td>2.4530</td>
<td>3.4530</td>
</tr>
<tr>
<td>Std Deviation</td>
<td>1.8849</td>
<td>3.9256</td>
<td>1.6090</td>
<td>1.5473</td>
</tr>
</tbody>
</table>
Results for Test A (cont.)

Error in XYZ-Distance for objects @2,100m

Various pan/tilt combinations (-15°,15°)

<table>
<thead>
<tr>
<th></th>
<th>dX</th>
<th>dY</th>
<th>dZ</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>9.3548</td>
<td>-9.3320</td>
<td>0.1588</td>
<td>15.4897</td>
</tr>
<tr>
<td>Std Deviation</td>
<td>4.5064</td>
<td>11.2934</td>
<td>3.9168</td>
<td>6.9080</td>
</tr>
</tbody>
</table>
Test B
Results for Test B

Error in XYZ-Distance

<table>
<thead>
<tr>
<th>Actual Depth</th>
<th>Calculated Depth</th>
<th>DX</th>
<th>DY</th>
<th>DZ</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1602</td>
<td>1604</td>
<td>1.47</td>
<td>11.42</td>
<td>-2.12</td>
<td>12.75</td>
</tr>
<tr>
<td>7484</td>
<td>7467</td>
<td>-6.4</td>
<td>28.23</td>
<td>1.88</td>
<td>147.68</td>
</tr>
<tr>
<td>15030</td>
<td>15467</td>
<td>-22.33</td>
<td>-518.9</td>
<td>10.84</td>
<td>520.79</td>
</tr>
</tbody>
</table>

Pixel Accuracy

<table>
<thead>
<tr>
<th>Depth</th>
<th>Left</th>
<th>Right</th>
<th>Top</th>
<th>Bottom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1601.5</td>
<td>14.74</td>
<td>14.47</td>
<td>1.12</td>
<td>1.03</td>
</tr>
<tr>
<td>7483.6</td>
<td>337</td>
<td>309.3</td>
<td>48.7</td>
<td>21.6</td>
</tr>
<tr>
<td>15029.8</td>
<td>1409.3</td>
<td>1187.2</td>
<td>243.7</td>
<td>26.4</td>
</tr>
</tbody>
</table>
Agenda

- Introduction
- Visual Servoing
- The New Software Architecture
- Implemented Visual Servos
  - Camera, Hand-Eye, and Head-Eye Calibrations
- Applications of the Software Architecture
  - Line Tracking
  - Mobile Robot Navigation
- Results and Conclusions
Feature-based Tracking

Model composed of straight line features

Extended Kalman Filter for pose estimation
Kalman-Filtering

Current pose estimate

Uncertainty region

Update uncertainty using assembly line dynamics

State statistics update

Iterate for all features

Landmark prediction

Landmark measure

Kalman Filter
Feature-based tracking (cont.)

Predicted Pose (red)

Extracted Pose (yellow)
Feature-based Tracking
(ICRA2003)

Model composed of Elliptical shapes

Binocular stereo tracking

Object pose is calculated with simple object reference frame definition
Example of Stereo-vision Tracking
Object pose calculation
American Sign Language

Facial Expressions (Eyebrow)
Active Appearance Models - AAM


Shape Vector: \[ X = (x_1 \ x_2 \ \ldots \ x_n \ y_1 \ y_2 \ \ldots \ y_n)^T \]

Texture Vector: \[ g = (g_1 \ g_2 \ \ldots \ g_m)^T \]

PCA Analysis:

\[ X = \bar{X} + \phi_s b_s \]

\[ g = \bar{g} + \phi_g b_g \]

Or

\[ X = \bar{X} + P_s c \]

\[ g = \bar{g} + P_g c \]
AAM Intuitively...

Synthesize any appearance by changing a set of parameters:

\[ p = (t|c) \]

where \( c \) = appearance parameters
and \( t \) = 2D pose parameters \((t_x, t_y, s, \Theta)\)

Ex: 2-class & 2-sample case
Facial Expressions
AAM and Pose Estimation (ICRA2004)

Tracking the control points (shape) frame by frame

3D Reconstruction of the shape

Pose Estimation
AAM – Tracking

\[ p = [c^T \mid t^T] \]

- Sampling the pixels inside the current shape
- Image warping into the mean shape (\( \overline{X} \))

\[ x = \overline{x} + Q_s c \]
\[ X = T_t (x) \]

\[ g = \overline{g} + Q_g c \]

Parameter Updating

\[ \Delta p^* = -R(p)r(p) \]

\[ c = c + \Omega c \]
\[ t = t + \Omega t \]
AAM – Tracking (Iterations)

\[ p = [c^T | t^T] \]

\[ x = x + Q_s c \]
\[ X = T_t(x) \]

\[ g = g + Q_g c \]

Parameter Updating

\[ \Delta p^* = -R(p)r(p) \]

- Sampling the pixels inside the current shape
- Image warping into the mean shape (\( \bar{x} \))
First Method: Stereo AAM

- Left appearance model → AAM matching → 3-D reconstruction → 3D scene points → Pose estimation
- Right appearance model → AAM matching → 3D model points

Estimated Pose \( \{Rx, Ry, Rz, Tx, Ty, Tz\} \)
Second Method: 3D Parameterization

- Sampling the pixels inside the current shape
- Image warping into the mean shape

\[ p = p + \frac{\partial L}{\partial p} \]

\[ \Delta p^* = -R(p)r(p) \]
Final Result
## Comparative Results

<table>
<thead>
<tr>
<th></th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mean of absolute error</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X (meters)</td>
<td>0.0070</td>
<td>0.0097</td>
<td>0.0105</td>
</tr>
<tr>
<td>Y</td>
<td>0.0061</td>
<td>0.0076</td>
<td>0.0078</td>
</tr>
<tr>
<td>Z</td>
<td>0.0034</td>
<td>0.0025</td>
<td>0.0035</td>
</tr>
<tr>
<td>Yaw (degree)</td>
<td>0.3530</td>
<td>0.5880</td>
<td>0.4875</td>
</tr>
<tr>
<td>Pitch</td>
<td>0.5268</td>
<td>0.5239</td>
<td>0.7470</td>
</tr>
<tr>
<td>Roll</td>
<td>0.2541</td>
<td>0.4149</td>
<td>0.2467</td>
</tr>
<tr>
<td><strong>Std of absolute error</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X (meters)</td>
<td>0.0056</td>
<td>0.0030</td>
<td>0.0082</td>
</tr>
<tr>
<td>Y</td>
<td>0.0071</td>
<td>0.0040</td>
<td>0.0068</td>
</tr>
<tr>
<td>Z</td>
<td>0.0026</td>
<td>0.0028</td>
<td>0.0032</td>
</tr>
<tr>
<td>Yaw (degree)</td>
<td>0.4389</td>
<td>0.2556</td>
<td>0.4142</td>
</tr>
<tr>
<td>Pitch</td>
<td>0.3921</td>
<td>0.0726</td>
<td>0.5082</td>
</tr>
<tr>
<td>Roll</td>
<td>0.2311</td>
<td>0.0957</td>
<td>0.2031</td>
</tr>
<tr>
<td><strong>Max of absolute error</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X (meters)</td>
<td>0.0250</td>
<td>0.0183</td>
<td>0.0516</td>
</tr>
<tr>
<td>Y</td>
<td>0.0284</td>
<td>0.0178</td>
<td>0.0379</td>
</tr>
<tr>
<td>Z</td>
<td>0.0109</td>
<td>0.0099</td>
<td>0.0281</td>
</tr>
<tr>
<td>Yaw (degree)</td>
<td>1.7634</td>
<td>1.2527</td>
<td>2.5138</td>
</tr>
<tr>
<td>Pitch</td>
<td>2.1204</td>
<td>0.6439</td>
<td>3.0327</td>
</tr>
<tr>
<td>Roll</td>
<td>0.9914</td>
<td>0.6759</td>
<td>1.1188</td>
</tr>
</tbody>
</table>

Note
(a) The result of the first method (Stereo AAM)
(b) The result of the second method (3D Parameterization)
(c) The result of our previous approach (Feature-Based ICRA2003).
black=ground-truth
red=1st method
blue=2nd method
green=circular feature approach
black=ground-truth
red=1st method
blue=2nd method
green=circular feature approach
black=ground-truth
red=1st method
blue=2nd method
green=circular feature approach
black=ground-truth
red=1st method
blue=2nd method
green=circular feature approach
black=ground-truth
red=1st method
blue=2nd method
green=circular feature approach
black=ground-truth
red=1st method
blue=2nd method
green=circular feature approach
Real-Time Multi-AAM
(IEEE Motion 2005 - IEEE Trans. PAMI)
Assembly On-the-fly
Assembly On-the-fly - Accuracy
Agenda

- Introduction
- Visual Servoing
- The New Software Architecture
- Implemented Visual Servos
  - Camera, Hand-Eye, and Head-Eye Calibrations
- Applications of the Software Architecture
  - Line Tracking
  - Mobile Robot Navigation
- Results and Conclusions
Mobile Robot Navigation

Multiple cues
No maps, natural landmark tracking
“Object-driven”
Object-driven Navigation

“Follow directions”:

- Select multiple objects to be tracked
- Identify object attributes ("recognize")
- Track and determine pose of object
- Servo the robot w.r.t. objects
Control Software Architecture

Multiple Control Loops:
Differences:

- Dynamic priorities
- Dynamic creation/deletion of loops
Cartesian Control Law

Pose Estimation

Image Feature Extraction

Arbitrator

**X**

**Y**
Object-Oriented Implementation
Classes, Objects, and Methods

- **Meteor Class**
  - **Double Door**
    - **Got_Left**
    - **Got_Right**
    - **Processed Frame**
  - **Light**
    - **Got_Left**
    - **Got_Right**
    - **Processed Frame**
  - **ObtAvoid**
    - **Processed Sona/Lidar**
    - **Visual Servo Comm.**
    - **Update Control**
    - **Arbt. Comm.**
PathPlanner

Instantiate

appLight, appDoor, appObstAvoid, etc…

Assign priorities and pose (HTM)

Re-instantiate

appDoor and appLight

Assign new priorities and new poses
Hallway Follower

Uses \textit{appLight}

Processing Steps

- Build X-histogram and Y-histogram
- Segment images based on brightness
- Select largest blobs and apply heuristics
- Apply Sobel to extract lines
- Apply linear regression to find line equations
- Find corner points and attributes of light fixture
- Apply heuristics and send coordinate to Arbitrator
Image Processing
Door Tracker

Same as before except:

- Histograms and segmentation are based on color
- Different heuristics
- Only 2 corners are extracted
Image Processing
Obstacle Avoidance

Check Sonar (and Lidar)

If there is an obstacle, circumnavigate.
Servo Control Module

- Different for Position and Image-based
- Calculates XYZOAT (or delta XYZOAT)
- Send commands to Arbitrator with Confidence level:
  - image processing
  - reconstruction
Agenda

- Introduction
- Visual Servoing
- The New Software Architecture
- Implemented Visual Servos
  - Camera, Hand-Eye, and Head-Eye Calibrations
- Applications of the Software Architecture
  - Line Tracking
  - Mobile Robot Navigation
- Results and Conclusions
Results

- Line Tracking running in five computers
- Simulating failures in zero, one, or more modules
- Results
  - Arbitrator and Fine Control: ~45fps
Contributions

- Multi-loop control architecture
- Distributed, fault-tolerant, modular, elastic, CPU-efficient, and flexible software architecture
- Efficient Client/Server infrastructure that satisfies the constraints of vision-based systems
- Simple control law using quaternions
- Accurate hand-eye and head-eye calibration algorithms
- General architecture: can be applied to visual servoing as well as to mobile robotics (etc.)
Conclusion

- Architecture reduces the delays by distributing modules over the network
- A modular design allows for redundancy, fault-tolerance, and subsumption of tasks

Future Work

- Better controllers:
  - PID using quaternion representation of the error function
  - Prediction using Kalman Filtering
- Geometric models of target (reduce error in posing)