Robotic Vision

What you see ≠ what you get
0. About the Presenter

Dr. Ying Wang
Computer Vision Algorithm Engineer

Research Interests:
Computer Vision, HRI, Robotics, Automation control

RWTH Aachen University, Germany
PhD Candidate in Mechanical Engineering, focusing on Robot Vision
Degree completed in 2016.01 with a mark of “sehr gut”
Topic: A Visual Servoing Approach to Human-robot Interactive Object Transfer

Harbin Institute of Technology, China
Master of Science in Mechatronics Engineering (top 5% of class)
Topic: Study of the DSP based Servo Controller of Die Bonding Machine

Wuhan University of Technology, China
Bachelor of Science in Mechanical Engineering and Automation (top 5%)
Topic: Design of Mechanical Structure of the Oil Tube marking Machine
Contents

1. Introduction

2. Modeling a Robotic Vision Problem

3. Solution Proposal

4. Summary
Robots are designed and built to complement human abilities.

**Introduction**

**Efficiency & Automation**
- Stacking boxes for shipping

**Skills & Accuracy**
- Medical operation
- Hazards Tolerance

**Mobility**
- Surveillance

**Spray painting**

**Robot Adaptability**
- Reaction
- Perception

**Demands for Robotic Vision**
Robotic Vision System

What you see (Scene)

Visual Sensor

Visual Data

What you get (visual measurement)

Introduction Modeling Solution Summary

Robot

Tasks

Algorithms & methods

Situational information

Visual Sensor

Processor

Dr. Ying Wang
Jul. 27th 2016
Robotic Vision System

Introduction

Modeling

Solution

Summary
Robotic Vision and Related Topics

Introduction       Modeling       Solution       Summary

© Sensopart

© Saab AB
Contents

1. Introduction
2. Modeling a Robotic Vision Problem
3. Solution Proposal
4. Summary
General Approach to Robotic Vision

Look-then-move

Initialization
Vision-based State Estimation
Robot Control

Look-and-move

Initialization
Vision-based State Estimation
Robot Control

Vision-motor model

Introduction  Modeling  Solution  Summary
Look-then-move

Visual sensing and manipulation are combined directly in an open-loop fashion.

The accuracy of the operation, in such a configuration, depends directly on the accuracy of the hardware, such as the visual sensors, the manipulator and the controller.
**Visual Servoing**

uses a visual-feedback control loop to increase the overall accuracy of the system - a principal concern in any application.

Visual servoing approaches broaden the application domain of robotic manipulation, as they do not need *a priori* knowledge of the workspace, that is, they are competent of visual control in an unmodeled environment.
State estimation, in the sensor space, mainly concerns the camera configuration and the image processing algorithms. Various visual information is extracted, such as color, pose, or features of objects, to describe the state of the target object.
Random variable $x$ denotes a quantity that is uncertain. This information is captured by the probability distribution $P_r(x)$ of the random variable. A random variable may be discrete or continuous.
State Estimation
Probability Model

Noise

Many-to-one mapping

©pudn

©PCL

Introduction Modeling Solution Summary
Maximum Likelihood

the maximum likelihood (ML) method finds the set of parameters $\hat{\theta}$ under which the data $\{x_i\}_{i=1}^I$ are most likely.

$$\hat{\theta} = \max_{\theta} [P_r(x_1 \ldots x_I | \theta)] = \max_{\theta} \left[ \prod_{i=1}^{I} P_r(x_i | \theta) \right]$$

Maximum a posteriori

maximum a posteriori estimation maximizes the posterior probability $[P_r(x_1 \ldots x_I | \theta)]$ of the parameters

$$\hat{\theta} = \max_{\theta} [P_r(\theta | x_1 \ldots x_I)] = \max_{\theta} \left[ \frac{P_r(x_1 \ldots x_I | \theta)P_r(\theta)}{P_r(x_1 \ldots x_I)} \right] = \max_{\theta} \left[ \frac{\prod_{i=1}^{I} P_r(x_i | \theta)P_r(\theta)}{P_r(x_1 \ldots x_I)} \right]$$

$$\hat{\theta} = \max_{\theta} \left[ \prod_{i=1}^{I} P_r(x_i | \theta)P_r(\theta) \right]$$
Bayesian approach

$$Pr(\theta \mid x_1 \ldots x_I) = \frac{\prod_{i=1}^I Pr(x_i \mid \theta)Pr(\theta)}{Pr(x_1 \ldots x_I)}$$

Evaluating the predictive distribution is more difficult for the Bayesian case since we have not estimated a single model but have instead found a probability distribution over possible models. Hence, we calculate

$$Pr(x^* \mid x_1 \ldots x_I) = \int Pr(x^* \mid \theta)Pr(\theta \mid x_1 \ldots x_I)d\theta$$

General Form

The predictive density calculations for the Bayesian, MAP and ML cases can be unified as

$$Pr(x^* \mid x_1 \ldots x_I) = \int Pr(x^* \mid \theta)\delta[\theta - \hat{\theta}]d\theta = Pr(x^* \mid \hat{\theta})$$
State Estimation
Machine Learning Solution

**model** to mathematically relate the visual data $x$ and the world state $w$. The model specifies a family of possible relationships between $x$ and $w$ and the particular relationship is determined by the model parameters $\theta$.

**Discriminative model**
$P_r(w|x)$
– model the contingency of the world state on the data

**Generative model**
$P_r(x|w)$
– model the contingency of the data on the world state

**Regression model**
To estimate a continuous quantity from continuous data. E.g. predicting the joint angles from an image of the human body.

**Classification model**
To predict a discrete quantity from continuous data. E.g. assigning a label to a region of the image to indicate whether or not a face is present.
**model** to mathematically relate the visual data $x$ and the world state $w$. The model specifies a family of possible relationships between $x$ and $w$ and the particular relationship is determined by the model parameters $\theta$.

**learning algorithm** to allow fitting the parameters $\theta$ using paired training examples $\{x_i, w_i\}$ where we know both the measurements and the underlying state.

**inference algorithm** to take a new observation $x$ and uses the model to return the posterior $P_{\tau}(w|x)$ over the world state $w$. 

---

**Introduction**

**Modeling**

**Solution**

**Summary**
**Vision-Motor Model**

- **Joint space**
- **Image space**
- **World space**

**Vision-based State Estimation** → **Robot Control**

**Vision-motor model**

**Estimated**

the model estimation can be conducted on- or off-line. The estimated image Jacobian relates the joint velocity directly to image space velocities, which can be estimated from previous measurements.

**Know a priori**

the forward or inverse kinematics of the robot (robot Jacobian) is available to deduce the differential changes between the joint and Cartesian space. e.g. PBVS, IBVS or 2 1/2D visual servoing.

---

**Introduction**  **Modeling**  **Solution**  **Summary**
Contents

1. Introduction
2. Modeling a Robotic Vision Problem
3. Solution Proposal
4. Summary
Solution Overview

Example: Object Following
**Task**: controlling the robot to follow the target

**Analysis Model**

- **Initialization**
- Vision-based State Estimation
- Vision-motor model
- Robot Control

**Design Model**

1. **Initialization**
   - Image formation
2. **Image processing**
3. **State Estimation**
   - Tracking
4. **Robot Control**
   - Robot Following
5. **Vision-motor model**
   - Know *a priori*

**Solution Overview**

*Example: Object Following*
Introduction  Modeling  Solution  Summary
Image Formation

Perspective Projection

Camera frame

Image frame

Projection line

Projection

\[
\begin{pmatrix}
X \\
Y \\
Z
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
f_x \frac{X}{Z} \\
f_y \frac{Y}{Z}
\end{pmatrix}
\]

Introduction  Modeling  Solution  Summary
**Image Formation**

**Perspective Projection**

**Correspondence Points**

\[
P = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}^T, \ p = \begin{pmatrix} x \\ y \end{pmatrix}^T
\]

**Camera Intrinsics**

\[
M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}
\]

**Perspective Projection**

\[
\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} [R_{CW} \ T] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}
\]
Task: controlling the robot to follow the target

What does the scene look like?
Which is the target object there?
Where is it located?
How should the robot be controlled?
Gaussian Filter \[ G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} \]
**Solution Overview**

Example: Object Following

**Task**: controlling the robot to follow the target

What does the scene look like?
Which is the target object there?
Where is it located?
How should the robot be controlled?

**Analysis Model**

- **Initialization**
  - Image formation
  - Image processing

- **State Estimation**
  - Tracking

- **Robot Control**
  - Robot Following

**Vision-motor model**
- Know *a priori*

**Design Model**

**Introduction**  |  **Modeling**  |  **Solution**  |  **Summary**
Tracking

Keypoints

(a) $x_t$, $x_t + u$

(b) $x_t$

(c) $x_t$, $x_t + u$

Introduction  Modeling  Solution  Summary
By including the surrounding neighbors, the underlying sampled surface geometry can be inferred and captured in the feature formulation, which contributes to solving the ambiguity comparison problem. Ideally, the resultant features would be very similar (with respect to some metric) for points residing on the same or similar surfaces, and different for points found on different surfaces, as shown in the figure below.
3D features are representations at a certain 3D point or position in space, which describe geometrical patterns based on the information available around the point. The data space selected around the query point is usually referred as the k-neighborhood.
Tracking
Local Descriptor - K-neighborhood

Introduction  Modeling  Solution  Summary
Tracking

Matching – Iterative Closest Point

- Initial pose guess
- Correspondence Searching
- Transform Computation
- Alignment Applying
- Converge?
  - Y
  - N

M (model)

S (scene)

Introduction  Modeling  Solution  Summary
**Tracking**

Matching – Iterative Closest Point

1. Initial pose guess
2. Correspondence Searching
3. Transform Computation
4. Alignment Applying
5. Converge?
   - Y: End
   - N: Go back to 1.

---

**Introduction**  **Modeling**  **Solution**  **Summary**
Tracking Workflow

Initialization
- Image formation
- Image processing

State Estimation
- Tracking

Robot Control
- Robot Following

Feature Extraction
- Feature Description
- Feature Matching
- Feature Tracking

Vision-motor model
Know a priori

Introduction  Modeling  Solution  Summary
Model
World state $w$ is continuous (3D pose) -> Regression model
Taking a generative approach, the likelihoods are described as

$$P_r(x | \omega = k)$$

Learning algorithm
the parameters from training data pairs $\{w_i, x_i\}_{i=1}^I$ where the pixels have been manually labeled. The prior parameter is learned from the world states $\{w_i\}_{i=1}^I$.

Inference algorithm aims to calculate the 3D pose of the object in the video stream.
**Solution Overview**

**Example: Object Following**

**Task**: controlling the robot to follow the target

What does the scene look like?
Which is the target object there?
Where is it located?
How should the robot be controlled?

**Design Model**

- **Initialization**
  - Image formation
  - Image processing
- **State Estimation**
  - Tracking
- **Robot Control**
  - Robot Following
- **Vision-motor model**
  - Know a priori

**Analysis Model**

- **Initialization**
- **Vision-based State Estimation**
- **Vision-motor model**
- **Robot Control**

**Solution Overview**

**Introduction**

**Modeling**

**Solution**

**Summary**
Vision-motor model: PBVS & IBVS
Pseudo codes - Initialization

1. set projModel ← perspectiveProjwithDistortion
2. set robot ← projModel
3. set point[4] //3D points
4. set dot[4]
5. compute cMo
6. set P ← (0, 0, 0)
7. set cdMo
8. compute pd ← cdMo, P
9. compute Zd from P
10. compute p ← cMo, P
11. compute Z from P
12. compute depth, tu
13. set task.addFeature ← (p, pd, depth, tu)
Solution Model of Machine learning
Visual Servoing

---

**Pseudo codes – Control design**

1. `set lamda ← (2.5, 0.2, 40)`
2. `set task.setServo ← EYEINHAND_L_cVe_eJe`
3. `set task.set_cVe(cVe) ← robot.set_cVe(cVe)`
4. `set task.set_eJe(eJe) ← robot.set_eJe(eJe)`
5. `set robot.setRobotState ← STATE VELOCITY_CONTROL`
Pseudo codes – Control loop

1. while true
2. for all feature points
3. get dot[i].x
4. get dot[i].y
5. compute & update cMo
6. Compute & update p
7. Compute & update tu
8. Compute & update depth
9. update task.set_cVe(cVe) ← robot.set_cVe(cVe)
10. update task.set_eJe(eJe) ← robot.set_eJe(eJe)
11. compute v
12. set robot.setVelocity
Solution Model of Machine Learning

Object Following

Introduction  Modeling  Solution  Summary
Robotic Vision by ROS

• Drivers
  – 2D/3D range finders
  – RGB-Depth cameras
  – monocular and stereo cameras

• API
  – Tools (pcl, visp, opencv with ros)
  – Support packages (calibration, recognition, image conversion, visualizer)
  – Messages
  – Topics
  – Services
  – parameters

• Tutorials & support
  www_ros wiki_com/
Solution Model of Machine learning

Abstraction

World model

Task-level reasoning

Scene interpretation

Feature extraction

Image Processing

Object motion planning

Trajectory generation

Joint control

Visual servoing

Perception

Reaction

Introduction  Modeling  Solution  Summary
谢谢！
Thank you for your attention!
Vielen Dank für Ihre Aufmerksamkeit!