Generic Object and Action Detection with LARK (Locally Adaptive Regression Kernels)

Haejong Seo
University of California, Santa Cruz
Mentor: Gary Bradski
Motivation

Human-Robot Interaction
Robot-Robot Interaction

1. Where to look?
2. Is there any motion?
3. Where is a bottle of beer located?
   How big is this bottle?
   What pose is this?
4. Is he/she waving hands to me?
   Is other PR2 approaching to me?
Motivation

- DOT (dominant orientation templates) by Stefan
  - can handle object detection
  - run with a single web-cam
  - pretty fast

However,

- can not deal with

LARK can tackle all the problems like
Motivation & Goal

Develop fast and robust detection systems in open source

1. static saliency

2. space-time saliency

3. object detection

4. action detection
Outline

1. LARK Overview
2. Saliency Detection
3. Object Detection
4. Space-time Saliency Detection
5. Action Detection
6. Conclusion
LARK (locally adaptive regression kernels)

- Euclidean distance vs. Geodesic distance
Arclength on the surface

\[ ds^2 = dx_1^2 + dx_2^2 + dz^2, \]

Chain rule

\[ = dx_1^2 + dx_2^2 + (z_{x_1} dx_1 + z_{x_2} dx_2)^2, \]

\[ = (1 + z_{x_1}^2)dx_1^2 + 2z_{x_1} z_{x_2} dx_1 dx_2 + (1 + z_{x_2}^2)dx_2^2, \]

\[ = [dx_1 \, dx_2] \begin{bmatrix} z_{x_1}^2 + 1 & z_{x_1} \, z_{x_2} \\ z_{x_1} \, z_{x_2} & z_{x_2}^2 + 1 \end{bmatrix} \begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix}, \]

\[ = \Delta x^T \mathbf{C} \Delta x + \Delta x^T \Delta x, \]
LARK → self-similarity

\[ K(C_l, \Delta x_l) = \exp(-ds^2) = \exp\left\{-\Delta x_l^T C_l \Delta x_l\right\} \]

**Gray level patch at centered at** \( x_{13} \)

**Geodesic distances between** \( x_{13} \) (a center) and surroundings

**LARK values**

Haejong Seo (summer Internship) (6)
LARK (example)
LARK (speed-up)

Step 1: downsample by a factor of 4

Step 2: interpolate $C = [C_{11}, C_{12}, C_{22}]$ after computing in a lower scale

$C_{11}$  $C_{12}$  $C_{22}$

0.02 sec (70 times faster)
Saliency Detection

LARK

self-resemblance

\[
S_i = \frac{1}{\sum_{j=1}^{N} \exp \left( -\frac{-1 + \rho(F_i,F_j)}{\sigma^2} \right)}
\]

Saliency map

thresholding
Saliency Detection (video)
Object Detection

Stage 1: Compute LARK

Stage 2: PCA

Stage 3: 1) Resemblance Map (RM) using Matrix Cosine Similarity
          2) Significance Tests
          3) Non-maxima Suppression

Final result
Object Detection (speed-up)

Use saliency to reduce search space
Face Detection (video)

One template

Three templates
Object Detection (video)

Door knob

PR2

Drawing

Small robot

Three templates
3-D Object Detection (speed-up)

Tree structure for template

Pyramid search

Haejong Seo (summer Internship) (15)
3-D Object Detection (video)

CD case

mouse

naked

organizer
3-D Object Detection (video)

Two objects

Three objects
3-D LARK → self-similarity in 3-D

Space-time gradient vector field

3x3 local covariance matrices
Space-time Saliency Detection

\[ S_i = \frac{1}{\sum_{j=1}^{N} \exp \left( -\frac{1 + \rho(F_i, F_j)}{\sigma^2} \right)} \]
Space-time Saliency (video)
Action Detection

(30~35 frames) template

Input video

stage 1: compute 3-D LARKs

stage 2: compute feature volumes

stage 3:
1. generate resemblance map (RV) by matrix cosine similarity (MCS)
2. significance testing by controlling the false discovery rate (FDR)

Final result

Haejong Seo (summer Internship) (20)
Action Detection (speed-up)

7 frames of 3-D LARK
(3x3 (space)x5 (time))

35 frames

5 frames

5 frames

space-time saliency

pyramid search

Haejong Seo (summer Internship) (21)
Action Detection (video)

4 actions

- waving
- sitting down
- moving closer
- boxing
Code Availability

Package **larks**: service that trains object templates and detects objects locations and poses (available now)
→ stacks/object_recognition_experimental/larks

Package **saliency**: service that provides salient regions in images and videos (will be available)
→ cturtle/wg-ros-pkg-unreleased/sandbox/saliency

Package **actiondetection**: service that detect generic human actions in videos (will be available)
→ cturtle/wg-ros-pkg-unreleased/sandbox/actiondetection
Discussion & Future Work

- Use a **tracking** algorithm to avoid **blinking** effects
- Use **parts-based** detection to deal with **occlusion** (Steve Gould)
- Learn threshold values for each object and action
- Improve **scalability** → build a common tree for all the objects
- Use LARK as a **post filter for BiGGPy**
Thank you!

Any Questions?