“the tubes”
a tool for local video analysis

14–7–2010

Outline

The “tubes” data structure

Application: depth from motion

Appendix:Technicalities

The “tubes” data structure

- *Representation* of a piece of black-and-white video
- Based on a *given* spatio-temporal segmentation
- Allows efficient queries
- This queries are building blocks for high-level algorithms (e.g., depth from motion)

$$\text{raw video} \implies \text{tubes} \implies \text{fancy high-level algorithms}$$

The “tubes” data structure

Objects:

- Pixels
- Frames
- Tubes (spatiotemporal regions = moving objects)
- Regions (tubes intersected by frames)
- Graph of adjacent regions (neighboring objects)

small figure illustrating the RAG of a frame

The “tubes” data structure

Queries:

- Given a region:
  - List of its pixels
  - List of neighboring regions
  - Corresponding region on the next/previous frame
  - ...
- Given a pixel:
  - Tube it belongs to
  - Region it belongs to
  - ...
- Given a tube:
  - List of its regions
  - ...
- ...

Tubes: an ideal pipeline

1. Raw video frames
2. Tubes data structure
3. High level algorithms

Tubes: the real pipeline

The computation of the tubes data structure depends on

- A pre-computed optical flow
- A pre-defined spatiotemporal segmentation technique

The API `tubes.h`

blah blah blah

Applications of the data structure

- Tracking (immediate implementation)
- Motion statistics (immediate implementation)
- Blotch detection (immediate implementation)
- Depth from motion (semi-immediate implementation)

Note: The results depend on the optical flow computation and the given segmentation method.

Application: tracking

copypaste code for tracking

Application: motion statistics

copypaste code for motion statistics

Application: blotch detection

copypaste code for blotch detection

Application: depth from motion

Goal: estimate the relative ordering of each pair of neighboring regions.

Criterion: **The boundary between two regions follows the region which is closer to the camera.**

Implementation: Traversal of the “tubes” data structure. Each pixel votes for a depth order, comparing its simulated movement to the real segmentation on the next frame.

Criterion for depth ordering

The boundary between two regions follows the region which is closer to the camera.

figure of the three local cases

Hypotheses for “depth from motion”

Our algorithm assumes the following:

1. The segmentation is correct
2. The optical flow is correct
3. The boundary between two objects follows the region which is on top.

All these statements are false.
However, they are a reasonable starting point.

Counterexamples to the third hypothesis

Counterexamples: A paper sliding through a fold, a treadmill, an offset printing press

figure of a paper sliding through a fold, with optical flows

Note: The rarer the counterexamples, the stronger the hypothesis.

Examples

Tree sequence

Examples

Door sequence

Technicalities

- The representation is as useful as the given segmentation
- Sensibility with respect to the given optical flow

Possible choices of segmentation

- Spatiotemporal bi-level sets on a pre-set quantization
- Spatiotemporal bi-level sets on adaptive levels
- Spatiotemporal maximally stable extremal regions
- Spatiotemporal Mumford-Shah-Morel segmentation of the image
- Spatiotemporal Mumford-Shah-Morel segmentation of the flow

Tradeoff on the over-segmentations

many small regions imply:

- average flows less robust
- high computational cost
- lots of unused computation on “inner boundaries”

few large regions imply:

- danger of joining different objects
- low computational cost
- most computations are useful (among real boundaries)

conclusion: a segmentation as coarse as possible, without joining different objects.

Possible choices of optical flow

Flows we tried:

- Multigrid (llus)
- Graph cuts (nicolas)
- SIFTflow

General problem: The optical flow is most interesting at the boundary of occlusions. Unfortunately, most methods fail to compute the flow at occlusions, or smooth it out too much.

Our proposal: Ignore the computed flow near the boundaries of regions, and extrapolate the flow from the middle of the region up to its boundary.

Future work

Fugure goals

- draw conclusion from all the trials (flows $\times$ segmentations $\times$ parameters)
- use the tubes to compute or update the optical flow