Fast Keypoint Detection in Video Sequences

Luca Baroffio, Matteo Cesana, Alessandro Redondi, Marco Tagliasacchi, Stefano Tubaro

Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano
Italy
Local Visual Features

- Starting point for many computer vision tasks
  - Object recognition
  - Content-based retrieval
  - Image registration

- Two-steps approach:
  - First step: keypoint detection (corners, blobs, etc.)
  - Second step: descriptor extraction (SIFT, SURF, BRISK, etc.)
Local features detection in video

- Most algorithms are tailored to still images

- For video, past literature targets the identification of keypoints that are **stable across time**
  - Stable features are key to object tracking, event identification and video calibration (main goal: application accuracy)
  - Stable features improve the efficiency of coding architectures exploiting the temporal redundancy (main goal: minimize bandwidth)

- We target computational complexity
  - Low power devices (smartphones, embedded systems, Visual Sensor Networks) require the process of features detection to be both **fast** and **accurate**
Fast extraction from video

- Baseline approach: apply a feature detector on each frame $\mathcal{I}_n$ of a video sequence
  - Inefficient from a computational point of view!
  - Temporal redundancy is not exploited!

- Our approach: apply the feature detector only in regions of $\mathcal{I}_n$ that are sufficiently different from $\mathcal{I}_{n-1}$
  - Compute a detection mask to identify such regions
  - Reuse keypoints from $\mathcal{I}_{n-1}$ outside those regions (keypoint propagation from $\mathcal{I}_{n-1}$ to $\mathcal{I}_n$)
Fast extraction from video

Formally:
- Let $\mathcal{D}_n$ be the set of features extracted from frame $\mathcal{I}_n$ (size $N_x \times N_y$)
- Let $d_{n,i} \in \mathcal{D}_n$ be the i-th features of the set, computed in keypoint location $p_{n,i}$
- Let $\mathcal{M}_n \in \{0, 1\}^{N_x \times N_y}$ be a binary detection mask defining the regions of the frame where the detector should be applied

\[
\mathcal{D}_n = \{d_{n,i} : \mathcal{M}_n(p_{n,i}) = 1 \} \cup \{d_{n-1,j} : \mathcal{M}_n(p_{n-1,j}) = 0\}
\]

New features

Propagated features
Detection Mask

- How to compute the detection mask $M_n$?
  - Need for a computationally efficient algorithm!

- We propose two alternatives:
  - Intensity Difference Detection mask
  - Keypoint Binning Detection Mask
Intensity Difference Detection Mask

- Idea: apply a detection only to regions that vary sufficiently across contiguous frames
- To this end, compute the absolute difference between downsampled representations of two consecutive frames
  - Already computed by the scale-space pyramid!
- If the difference in a given region is greater than a threshold, perform detection in such a region

\[
M'_{n,o}(k,l) = \begin{cases}
1 & \text{if } |L_{n,o}(k,l) - L_{n-1,o}(k,l)| \leq T_I \\
0 & \text{if } |L_{n,o}(k,l) - L_{n-1,o}(k,l)| > T_I,
\end{cases}
\]

- Final mask obtained through upsampling
Intensity Difference Detection Mask

Idea: apply a detector only to regions of the image that sufficiently vary across contiguous frames.
Keypoint Binning Detection Mask

- Idea: apply a detection only to regions where features have been found in previous frames
- To this end, compute a 2D spatial histogram of keypoints location
- If the number of keypoints in a spatial bin (of the previous frame) is greater than a threshold, perform detection in such a region

\[ \mathcal{M}'_n(k, l) = \begin{cases} 
1 & \text{if } \mathcal{M}''_n(k, l) \geq T_H \\
0 & \text{if } \mathcal{M}''_n(k, l) < T_H,
\end{cases} \]
Keypoint Binning Detection Mask

Idea: apply a detector only to regions of the image where features have been found in previous frames.
Idea: apply a detector only to regions of the image where at least N features have been found in previous frames.
**Idea:** apply a detector only to regions of the image where features have been found in previous frames.
**Keypoint Binning Detection Mask**

**Idea:** apply a detector only to regions of the image where features have been found in previous frames.
Experiments

- Datasets:
  - Stanford MAR dataset (4 sequences of cd covers under different imaging conditions)
  - Rome Landmark dataset (10 sequences of different landmarks in Rome)
  - Stanford MAR multiple object (4 sequences of different objects)

- Selected local features: BRISK (but our methods is generally appliable)

- Depending on the dataset, different accuracy measures:
  - Matches-post-Ransac (MPR) for Stanford MAR dataset
  - Mean of Average Precision (MAP) for Rome Landmark dataset
  - Combined detection and tracking accuracy for Stanford MAR multiple objects

- Complexity is measured by means of the required CPU time
Comparison with baselines

![Graph showing accuracy over time with different detection methods and their respective accuracy metrics.](image)

- **full detection**
- **ID detection mask**
- **KB detection mask**
- **patch propagation, \( \Delta = 10 \)**
Results – Stanford MAR

- Full detection
- ID detection mask
- KB detection mask

30% reduction

accuracy [MPR]

computational time [ms]
Results – Rome Landmark Dataset

35% reduction, 0.03% loss in MAP

- full detection
- ID detection mask
- KB detection mask

ICASSP 2016  March 2016
Results – Stanford Multiple Object

- Feature extraction time/frame [ms]
- Average tracking precision [pixel]

- Full detection
- ID detection mask
- KB detection mask

40% reduction
Conclusions

- **Up to 35/40 % reduction** in terms of computational complexity without significantly reducing visual task accuracy

- Higher frame rates / lower power consumption on low-power devices (smartphones, embedded systems)

Thank you!