Fusing points and lines for high performance real-time tracking

Ed Rosten, Tom Drummond

University of Cambridge
Model based tracking

- Edge tracking
- Point tracking
  - FAST features
  - Robust optimize

Combine for robust tracking

- Different failure modes
  - Combine for extra robustness
  - Combination is difficult
    - Statistics are non-Gaussian
Edge based tracking

- Start from position prior
Edge based tracking

- Start from position prior
- Search along edge-normal lines
Edge based tracking

- Start from position prior
- Search along edge-normal lines
- Adjust position to minimize errors
Edge based tracking

- Start from position prior
- Search along edge-normal lines
- Adjust position to minimize errors
- Gives drift free measurements
  - Model is static
Good prior needed

- Edges are a step change in intensity
- Correspondence is hard—pick closest edge
Good prior needed

- Edges are a step change in intensity
- Correspondence is hard—pick closest edge
- Prior must be good, or the wrong edge will be found
  - Correct edge might be nowhere near
Non Gaussian posterior

- Correct correspondences
  - Tracking is accurate
- Incorrect correspondences
  - Tracking is inaccurate—even if prior is good
Non Gaussian posterior

- Correct correspondences
  - Tracking is accurate
- Incorrect correspondences
  - Tracking is inaccurate—even if prior is good
Point based tracking

Detect features
Point based tracking

Detect features

Project features on to model.
Drift occurs here
Point based tracking

Detect features

Detect and match features in next frame

Project features on to model.
Drift occurs here
Point based tracking

1. Detect features
2. Project features on to model. Drift occurs here
3. Detect and match features in next frame
4. Alter pose to minimize reprojection error
Point based tracking

1. Detect features
2. Project features on to model.
3. Detect and match features in next frame
4. Alter pose to minimize reprojection error

Drift occurs here
The FAST feature detector

Contiguous pixels brighter than $p + \text{threshold}$

Rapid rejection by testing 1, 9, 5 then 13

1.59ms (Opteron 2.6GHz) - 8% of available CPU time

Source code available (see paper for URL)

16 test pixels used for feature vector

SSD used for matching between frames
The FAST feature detector

contiguous pixels brighter than $p \pm \text{threshold}$

Rapid rejection by testing 1, 9, 5 then 13

1.59ms (Opteron 2.6GHz) - 8% of available CPU time

Source code available (see paper for URL)

16 test pixels used for feature vector

SSD used for matching between frames
The FAST feature detector

- ≥ 12 contiguous pixels brighter than $p + \text{threshold}$

Rapid rejection by testing 1, 9, 5 then 13

1.59ms (Opteron 2.6GHz) - 8% of available CPU time

Source code available (see paper for URL)

16 test pixels used for feature vector

SSD used for matching between frames
The FAST feature detector

- $\geq 12$ contiguous pixels brighter than $p + \text{threshold}$
- Rapid rejection by testing 1, 9

Source code available (see paper for URL)
The FAST feature detector

- ≥ 12 contiguous pixels brighter than $p + \text{threshold}$
- Rapid rejection by testing 1, 9, 5
The FAST feature detector

- ≥ 12 contiguous pixels brighter than $p + \text{threshold}$
- Rapid rejection by testing 1, 9, 5 then 13

Source code available (see paper for URL)

Rapid rejection by testing 1, 9, 5 then 13
The FAST feature detector

- $\geq 12$ contiguous pixels brighter than $p + \text{threshold}$
- Rapid rejection by testing 1, 9, 5 then 13
- 1.59ms (Opteron 2.6GHz) - 8% of available CPU time
- Source code available (see paper for URL)
The FAST feature detector

- \( \geq 12 \) contiguous pixels brighter than \( p + \text{threshold} \)
- Rapid rejection by testing 1, 9, 5 then 13
- 1.59ms (Opteron 2.6GHz) - 8% of available CPU time
- Source code available (see paper for URL)
- 16 test pixels used for feature vector
- SSD used for matching between frames
Position optimization

- Sometimes > 90% outliers (even with SIFT!)
  - Robust optimize required
- Use EM
  - Mixture model is Gaussian (inliers) + uniform (outliers)
- SSD has some information about inlier probability
  - If only we knew the relationship...
EM provides probability that a match is correct
SSD for each match is known
Compute smooth function mapping SSD to probability
Use function to compute priors for each match next frame
Matching prior

- EM provides probability that a match is correct
- SSD for each match is known
- Compute smooth function mapping SSD to probability
- Use function to compute priors for each match next frame
Measurement Properties

- **Edge based tracking**
  - Requires
    - 3D geometric model
    - Good pose prior
  - Provides
    - Drift free measurements
    - Non Gaussian posterior

- **Point based tracking**
  - Requires
    - 3D point cloud
  - Provides
    - Robust differential measurements...
    - ...with approximately Gaussian posterior
Sensor fusion

- Either tracker can be wrong
  - Edge tracker can get correspondence wrong
  - Point based tracker can drift
- Posterior can be multimodal
  - Simple solutions do not work
Sensor fusion

- Either tracker can be wrong
  - Edge tracker can get correspondence wrong
  - Point based tracker can drift
- Posterior can be multimodal
- Evaluate modes next frame when more data arrives
Results - Strong unmodelled edges

Strong unmodelled edges frequently break the edge tracker
Results - Camera shake

- Pick up camera and shake *really* hard
- Can you follow the video? I can’t (but my tracker can)
Results - Camera shake

- 6Hz Camera shake
- Up to 204 pixels prediction error (89 average)
Results - Handheld camera

Pick up the camera and run around the lab
Summary

- A very fast feature detector
- An efficient, robust point based tracker
- Online modelling of match quality
- Careful modelling resulting in robust combination of trackers.

Any questions?
Efficient feature matching

Increasing mean

- Sort features by mean value of feature vectors
Efficient feature matching

- Sort features by mean value of feature vectors
- Find closest mean by binary search
Efficient feature matching

Increasing mean

- Sort features by mean value of feature vectors
- Find closest mean by binary search
- Search outwards
Efficient feature matching

Increasing mean

- Sort features by mean value of feature vectors
- Find closest mean by binary search
- Search outwards
- SSD between means bounds search
Efficient feature matching

- Sort features by mean value of feature vectors
- Find closest mean by binary search
- Search outwards
- SSD between means bounds search
- Best match has lowest SSD
# How FAST?

Percentage of available CPU time (typical video)

<table>
<thead>
<tr>
<th>Detector</th>
<th>2.6 GHz (%)</th>
<th>850 MHz (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>New FAST</td>
<td>5.4</td>
<td>21.7</td>
</tr>
<tr>
<td>FAST</td>
<td>7.45</td>
<td>48.5</td>
</tr>
<tr>
<td>DoG</td>
<td>301</td>
<td>1280</td>
</tr>
<tr>
<td>SUSAN</td>
<td>37.9</td>
<td>137.5</td>
</tr>
<tr>
<td>Harris</td>
<td>120</td>
<td>830</td>
</tr>
</tbody>
</table>