Efficient Interest Point Detectors & Features

Instructor - Simon Lucey

16-423 - Designing Computer Vision Apps
Today

• Review.
• Efficient Interest Point Detectors.
• Efficient Descriptors.
Review

• In classical Structure from Motion (SfM) computer vision pipeline there are four steps,
  1. Locate interest points.
Make decision based on image structure tensor

\[
H = \sum_{i \in \mathcal{N}} \frac{\partial I(x_i)}{\partial x} \frac{\partial I(x_i)}{\partial x}^T
\]
Scalar Measures of “Cornerness”

• A popular measure for measuring a corner $\lambda_1 + \lambda_2$, 

$$\text{tr}[H_{(x,y)}] = \|\nabla_x \ast I(x,y)\|_2^2 + \|\nabla_y \ast I(x,y)\|_2^2$$

$$\approx \|L \ast I(x,y)\|_2^2$$

“Laplacian (L)”
Scalar Measures of “Cornerness"

- A popular measure for measuring a corner \( \lambda_1 + \lambda_2 \),

\[
\text{tr}[\mathbf{H}(x,y)] = \| \nabla_x * I(x,y) \|^2_L + \| \nabla_y * I(x,y) \|^2_L \\
\approx \| L * I(x,y) \|^2_L
\]
Example - DoGs in SIFT
Review

• In classical Structure from Motion (SfM) computer vision pipeline there are four steps,
  1. Locate interest points.
  2. Generate descriptors.
Review - SIFT Descriptor

1. Compute image gradients
2. Pool into local histograms
3. Concatenate histograms
4. Normalize histograms
In classical Structure from Motion (SfM) computer vision pipeline there are four steps,

1. Locate interest points.
2. Generate descriptors.
Matching Descriptors

View 1

View 2

“Descriptor at position $x$"

ψ\{x\}

See BFMatcher class in OpenCV!!!
Matching Descriptors

\[ \zeta(i) = \arg \min_j \| \psi\{ \mathbf{x}_i^{(1)} \} - \psi\{ \mathbf{x}_j^{(2)} \} \|_2^2 \]

Variants other than nearest neighbor are possible!!!
Matching Descriptors

\[ \zeta(i) = \arg \min_j \| \psi \{ x_i^{(1)} \} - \psi \{ x_j^{(2)} \} \|_2^2 \]

Variants other than nearest neighbor are possible!!!
In classical Structure from Motion (SfM) computer vision pipeline there are four steps,

1. Locate interest points.
2. Generate descriptors.
4. Robust fit.

\[
\arg\min_{\Phi} \eta \{ x_i^{(1)} - \text{hom}[x^{(2)}_{\zeta(i)}; \Phi] \}
\]
Review - RANSAC

Original images  Initial matches  Inliers from RANSAC
Today

• Review.
• Efficient Interest Point Detectors.
• Efficient Descriptors.
Most classical interest point detectors require the employment of oriented edges.
Problem - Gaussian Filtering is Slow

• Naively, filtering with Gaussians is relatively slow on most modern architectures.

\[
\begin{bmatrix}
1, 0, -1 \\
2, 0, -2 \\
1, 0, -1
\end{bmatrix}
\]
(Sobel)

• Does not lend itself well to parallelization as the variance of the Gaussian filter increases (even with FFT).
• Computational cost increases dramatically as a function of the size of the filter.
Gaussian Filter is Separable

In MATLAB,

$$\text{h1D} = \text{fspecial('gaussian',[25,1],3)};$$

$$\text{h2D} = \text{kron(h1D,h1D')};$$
Gaussian Filter is Separable

In MATLAB,

```matlab
>> mesh(i)
```

```matlab
>> h2D = imfilter(i, h1D);
>> h2D = imfilter(h2D, h1D');
>> mesh(h2D)
```
More Problems - Scale

- However, even slower when you have to process things across multiple scales.

\[
\begin{bmatrix}
1, 0, -1 \\
2, 0, -2 \\
1, 0, -1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1, 0, -1 \\
2, 0, -2 \\
1, 0, -1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1, 0, -1 \\
2, 0, -2 \\
1, 0, -1
\end{bmatrix}
\]
One strategy has been to approximate oriented filters with box filters.

Most notably the SURF (Speed Up Robust Feature) descriptor of Bay et al. ECCV 2006.
Integral Image Trick

- We need to compute the box filter values many, many times and we must do it very fast!

\[
II(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')
\]
Computing Integral Images

- Computing sum of pixels in a rectangular area:

\[ f(A) = \]

\[ \sum \text{(A)} + \sum \text{(B)} + \sum \text{(C)} + \sum \text{(D)} \]

A 3-box filter takes 8 array lookups!
Computing Integral Images

• Computing sum of pixels in a rectangular area:

\[ f(A) = II(A) \]
Computing Integral Images

• Computing sum of pixels in a rectangular area:

\[ f(A) = II(A) - II(B) \]
Computing Integral Images

• Computing sum of pixels in a rectangular area:

\[ f(A) = II(A) - II(B) - II(C) \]
Computing Integral Images

- Computing sum of pixels in a rectangular area:

\[
f(A) = II(A) - II(B) - II(C) + II(D)
\]
Computing Integral Images

• Computing sum of pixels in a rectangular area:

\[ f(A) = II(A) - II(B) - II(C) + II(D) \]

• A 3 box filter array takes only 8 lookups.
Fast Gaussian Filtering

- Iterative box filters can also be applied to obtain extremely efficient Gaussian filtering,

In MATLAB,

\[
\text{>> mesh(b)}
\]

\[
\text{>> mesh(imfilter(imfilter(b,b),b))}
\]
SURF - Efficient Computation

- Positives:
  - Filters can be efficiently applied irrespective of size.
  - Integral images well suited in particular to SIMD.
  - Can take advantage of fixed integer arithmetic.

- Negatives:
  - Due to recursive nature cannot be easily parallelized.
  - All pixels in local region need to be touched.
  - Outputs floating/integer point metric of interest.
• Features from Accelerated Segment Test (FAST) - basis for most modern day computationally efficient interest point detectors.
• Proposed by Rosten et al. PAMI 2010.
• Operates by considering a circle of sixteen pixels around the corner candidate.
• Features from Accelerated Segment Test (FAST) - basis for most modern day computationally efficient interest point detectors.
• Proposed by Rosten et al. PAMI 2010.
• Operates by considering a circle of sixteen pixels around the corner candidate.
Why is it FAST?

- FAST relies heavily upon simple binary test,
  \[ I(x_p) - t > I(x_n) \]
- Does not have to touch all pixels before making a decision.
- Again, lends itself strongly to the employment of SIMD.
- Does not rely on integral images.
- Very good at finding possible corner candidates.
- Can still fire on edges, (can use Harris to remove false positives).
- Is NOT multi-scale.
• Review.
• Efficient Interest Point Detectors.
• Efficient Descriptors.
SURF Descriptor

- SURF also proposed a more efficient descriptor extraction strategy using box filters,

- Rest of the descriptor quite similar to SIFT.
Reminder - SIFT Descriptor

1. Compute image gradients
2. Pool into local histograms
3. Concatenate histograms
4. Normalize histograms
BRIEF Descriptor

• Proposed by Calonder et al. ECCV 2010.
• Borrows idea that binary comparison is very fast on modern chipset architectures,

\[ \psi_I(x, \Delta_1, \Delta_2) = \begin{cases} 
1 : I(x + \Delta_1) > I(x + \Delta_2) \\
0 : \text{otherwise}
\end{cases} \]

• Combine features together compactly,

\[ \psi_I(x) = \sum_{i} 2^{i-1} \psi_I(x, \Delta_1^{(i)}, \Delta_2^{(i)}) \]
Why do Binary Features Work?

• Success of binary features says something about illumination invariance.
• Absolute values of pixels do not matter.
• Makes sense as variable lighting source will effect the gain and bias of pixels, but the local ordering should remain relatively constant.
BRIEF Descriptor

- Do not need to “touch” all pixels, can choose \( \{\Delta_1, \Delta_2\} \) pairs randomly and sparsely,
BRIEF Descriptor

• Measuring distance between descriptors,

\[ d(\psi\{\mathbf{x}_i\}, \psi\{\mathbf{x}_j\}) = \text{Hamming distance} \]

• e.g.,

\[ d\left(\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\right) = 1 \]

Why not use Euclidean distance?
ORB

- Rublee et al. ICCV 2011 proposed Oriented FAST and Rotated BRIEF (ORB).
- Essentially combines FAST with BRIEF.
- Demonstrated that ORB is 2 orders of magnitude faster than SIFT.
- Very useful for mobile devices.
ORB

- ORB is patent free and available in OpenCV,

<table>
<thead>
<tr>
<th>Detector</th>
<th>ORB</th>
<th>SURF</th>
<th>SIFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time per frame (ms)</td>
<td>15.3</td>
<td>217.3</td>
<td>5228.7</td>
</tr>
</tbody>
</table>

These times were averaged over 24 640x480 images from the Pascal dataset [9]. ORB is an order of magnitude faster than SURF, and over two orders faster than SIFT.
More to read...