SEMINAR CIIRC CTU

Czech Institute of Informatics, Robotics, and Cybernetics
Czech Technical University in Prague

Lecturer: Dmytro Mishkin
Title: Affine Correspondences and Where to Find Them
Project name: Intelligent Machine Perception
Project Registration No: CZ.02.1.01/0.0/0.0/15_003/0000468
Venue: CIIRC, B-670, Jugoslávských partyzánů 1580/3, Praha 6
Date: 25.8.2022, 10:00 - 12:00
Affine Correspondences and Where To Find Them

Presenter: Dmytro Mishkin


Many slides are re-used from Computer Vision Methods course at CTU in Prague
Local affine frame (LAF) =
Point coordinate + affine shape (in 1 image)
Affine Correspondences

Local affine frame (LAF) =
Point coordinate + affine shape (in 1 image)

Affine correspondence (AC) =
point correspondence + shape change as a linear transformation (in 2 images)
Example Applications of ACs

**Single-view Geometry Problems**
- Vanishing-point and line estimation
- Scene-plane segmentation
- Auto-calibration
- Repetitive texture detection

**Stereo problems**
- Homography estimation
- Relative pose estimation
- Multi-plane segmentation
- Surface normal estimation

**N-view Geometry Problems**
- Multi-view surface normal estimation
- Generalized pose estimation

**Local Features**
- Matching and filtering features
  - Feature correction

...
Reconstruction and Visual Localization

Oriented Point Cloud Reconstruction

Simultaneous Localization and Mapping (SLAM)


Planar Segmentation


Horizon and Gravity Direction
Rectification

Pritts et al., series of works.
Symmetry Detection and Segmentation

input

rotational symmetry

wallpaper

arbitrary

Pritts et al, Detection, Rectification and Segmentation of Coplanar Repeated Patterns, In CVPR 2014
Infrastructure-based Auto-calibration

Symmetries along the Manhattan frame...

can be used for auto-calibration

Lochman et al, Minimal Solvers for Single-View Lens-Distorted Calibration, In WACV 2021
Sparse Feature Extraction and Description Pipeline

1. Find good locations
2. Find correspondences
3. Matching and filtering
4. Geometric verification (RANSAC)

Image credit: Andrea Vedaldi, ICCVW 2017
Dense correspondences, aka scene flow, etc.

- Dense correspondences: for each location in image $\mathcal{I}_1$ find corresponding location in image $\mathcal{I}_2$. Or – almost the same task – optical flow estimation.

Source: RAFT: Recurrent All-Pairs Field Transforms for Optical Flow, Zachary Teed, Jia Deng. 2020
Examples of different types of features and their covariance: translation, scale, rotation

- Translation-covariant: Harris, Hessian, SuperPoint
- Translation, scale and rotation-covariant:
  - SIFT, ORB, Harris-Laplace, R2D2 (not rotation)
- Affine-covariant:
  - Handcrafted: Hessian-Affine, MSER, Edge Foci
  - Learned: AffNet, ASLFeat
  - Simulation-based: ASIFT, MODS
- Upgrades from (rough) region correspondences:
  - LOCATE, Patch2pix, Multiview-correction, etc
Here's the leaderboard of prompts to add to GPT-3. Can you guys come up with anything better?

<table>
<thead>
<tr>
<th>No.</th>
<th>Template</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Let's think step by step</td>
<td>78.7</td>
</tr>
<tr>
<td>2</td>
<td>First, (*1)</td>
<td>77.3</td>
</tr>
<tr>
<td>3</td>
<td>Let's think about this logically.</td>
<td>74.5</td>
</tr>
<tr>
<td>4</td>
<td>Let's solve this problem by splitting it into steps. (*2)</td>
<td>72.2</td>
</tr>
<tr>
<td>5</td>
<td>Let's be realistic and think step by step.</td>
<td>70.8</td>
</tr>
<tr>
<td>6</td>
<td>Let's think like a detective step by step.</td>
<td>70.3</td>
</tr>
<tr>
<td>7</td>
<td>Let's think</td>
<td>57.5</td>
</tr>
<tr>
<td>8</td>
<td>Before we dive into the answer,</td>
<td>55.7</td>
</tr>
<tr>
<td>9</td>
<td>The answer is after the proof.</td>
<td>45.7</td>
</tr>
<tr>
<td></td>
<td>(Zero-shot)</td>
<td>17.7</td>
</tr>
</tbody>
</table>
Keypoint location \((x,y)\)
Response-based local feature detector

The Algorithm:

1. Compute response map $R$
2. Find local maxima in $R$
3. Filter some of responses by:
   a) Thresholding at $T$
   b) Taking top-$K$ features
Response

Image

Hessian response

SuperPoint response

KeyNet response
Non-maxima suppression here means $x = x$ if $x = \text{MaxPool2d}(x, \text{neighborhood})$ else 0.
Keypoints

Image  Hessian keypoints  SuperPoint keypoints  KeyNet keypoints
Link to NMS in object detection

- In object detection nms is IoU-centric – we want better object coverage rather than finding “center of the horse”.

- In local feature detection we assume constant size and shape of the patch, so votes are only for the center location.
  - Note: it would be interesting to explore more advanced ways of integrating response map into local feature location

Keypoints

Image

Hessian keypoints

SuperPoint keypoints

KeyNet keypoints
...but each filter or CNN has receptive field
Therefore we have scale (even if not precise)
Keypoints

Image

Hessian keypoints

SuperPoint keypoints

KeyNet keypoints
Keypoints

Image  Hessian detections  SuperPoint detections (top-100)  KeyNet detections (top-200)
Moving from constant scale
to scale selection
Scale Selection for linear filters (Hessian, DoG)

- Construct Gaussian scalespace – “stack of gradually smoothed versions” of original image
- Run the same detector on Gaussian scalespace
- Response of Laplacian and the determinant of the Hessian on Gaussian blobs with standard deviations 8, 16, 24 and 32 in red x points of Gaussian scalespace
Handcrafted scale-convariant detectors

- Scale invariant detectors, find local extrema (both in space and scale) of Laplacian (f in example) or determinant of Hessian response in gaussian scalespace.

\[ f(I, \sigma) \to \sigma^3 \]

list of local maxima of \( f \) w.r.t. \((x, y, \sigma)\)
Sub-pixel/ Sub-level Keypoint Localization

- Detect maxima and minima of difference-of-Gaussian in scale space
- Fit a quadratic to surrounding values for sub-pixel and sub-scale interpolation (Brown & Lowe, 2002)

- Taylor expansion around point:

\[ D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \]

- Offset of extremum (use finite differences for derivatives):

\[ \hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x} \]

- That is why SIFT detector is so precisely localized
Keypoints

Image

Hessian detections (top-500)

SuperPoint detections (top-100)

KeyNet detections (top-200)
What can we do for learned detectors?
1. Denial: “A girl has no scale”

- Works well for descriptor, when receptive field is big and training data is diverse enough. Examples: **SuperPoint, DISK**
Scale selection strategies for learned detectors

1. Denial: “A girl has no scale”
   - Works well for descriptor, when receptive field is big and training data is diverse enough. Examples: SuperPoint, DISK

2. Anger
   - Solve the problem by the brute force: run detector on scale pyramid (most common factors are 1.4x and 2x) keep points from all levels. Examples: ORB, KeyNet, R2D2
2. Anger

• Solve the problem by the brute force: run detector on scale pyramid (most common factors are 1.4x and 2x) keep points from all levels. Examples: ORB, KeyNet, R2D2
Scale selection strategies for learned detectors

1. Denial: “A girl has no scale”
   - Works well for descriptor, when receptive field is big and training data is diverse enough. Examples: SuperPoint, DISK

2. Anger
   - Solve the problem by the brute force: run detector on scale pyramid (most common factors are 1.4x and 2x) keep points from all levels. Examples: ORB, KeyNet, R2D2

3. Acceptance
   - Train separate scale estimator per each keypoint location (self-sca-ori, S3Esti)
     Self-Supervised Learning of Image Scale and Orientation Estimation, BMVC 2021
     Learning Soft Estimator of Keypoint Scale and Orientation With Probabilistic Covariant Loss, CVPR 2022
Scale selection strategies for learned detectors

3. Acceptance

SuperPoint

Constant scale

self-sca-ori
Drawbacks of “external scale” for SuperPoint/DISK

- Original descriptors (and correspondences) are not connected to the estimated scale
- You can use other patch descriptor for new scale (HardNet, SOSNet, etc), but that might be suboptimal and time-consuming
- Pretrained models (self-sca-ori, S3Esti) are trained on DoG (SIFT) keypoints, so are also suboptimal
Good bye, SuperPoint!
It is not you, it is me – I need affinity

SuperAffinePoint – anyone?
Keypoints

Image

Hessian detections (top-500)

KeyNet detections (top-500)
Orientation
No orientation means up-is-up prior
Keypoints

Image

Hessian detections (top-500)

KeyNet detections (top-500)
If images are upright for sure: don’t detect orientation

<table>
<thead>
<tr>
<th></th>
<th>CV-$\sqrt{\text{SIFT}}$</th>
<th>HardNet</th>
<th>SOSNet</th>
<th>LogPolarDesc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NI↑</td>
<td>mAA($10^\circ$)↑</td>
<td>NI↑</td>
<td>mAA($10^\circ$)↑</td>
</tr>
<tr>
<td>Standard</td>
<td>281.7</td>
<td>0.4930</td>
<td>432.3</td>
<td>0.5543</td>
</tr>
<tr>
<td>Upright</td>
<td>270.0</td>
<td>0.4878</td>
<td>449.2</td>
<td>0.5542</td>
</tr>
<tr>
<td>$\Delta$ (%)</td>
<td>-4.15</td>
<td>-1.05</td>
<td>+3.91</td>
<td>-0.02</td>
</tr>
<tr>
<td>Upright++</td>
<td>358.9</td>
<td>0.5075</td>
<td>527.6</td>
<td>0.5728</td>
</tr>
<tr>
<td>$\Delta$ (%)</td>
<td>+27.41</td>
<td>+2.94</td>
<td>+22.04</td>
<td>+3.34</td>
</tr>
</tbody>
</table>

Upright: angle $\rightarrow$ zero, remove duplicates.
Upright++: angle $\rightarrow$ zero, remove duplicates, get more features until again 8k
If you cannot rely on photographer: detect “canonical” orientation
Dominant gradient orientation

- Extract scale-normalized patch
- Calculate image gradients per pixel
- Convert them into (magnitude and orientation)
- Gather them into a single histogram and take a maximum (or maxima)

Toy example

- Input patches

- Output patches

Code available [here](#)
Less toy example

- Input patches

- Output patches

Code available [here](#)
Keypoints

Image

Hessian detections (top-500)

KeyNet detections (top-500)
ORB: Center of mass orientation

- Local orientation for FAST features (corners):
  - Find the patch “center of the mass” = intensity centroid C from image moments
  - Corner orientation is angle between OC and x axis

\[
m_{pq} = \sum_{x,y} x^p y^q I(x, y)
\]

\[
C = \left( \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)
\]

\[
\phi = \begin{cases} 
\tan^{-1} \frac{m_{01}}{m_{10}} & \text{if colour = bright} \\
\tan^{-1} \frac{m_{01}}{m_{10}} + 180^\circ & \text{if colour = dark}
\end{cases}
\]

\[1\text{ P. L. Rosin “Measuring Corner Properties”. CVIU 1999}\]
Learned orientation estimation

• Train CNN to regress repeatable patch orientation
Toy example: OriNet

- Input patches

- Output patches

Code available [here](#)
Less toy example

- Input patches
- Output patches

Code available [here](#)
Shameless plug:
Many slides are created help of local features implemented in Kornia
What is Kornia?

- Kornia is a differentiable vision library built on top of PyTorch.
- Re-implementation of classical vision is terms of differentiable tensors.
- Curate the latest greatest vision algorithms and community.

https://github.com/kornia/kornia
Train a network in self-supervised way to estimate patch orientation and scale
S3Esti: similar idea

- Train a network in self-supervised way to estimate patch orientation and scale
S3Esti: example

Original patch $X_1$ in the 1st image → S3Esti → Predicting scale and orientation of $X_1$ → Rectified patch of $X_1$

Original patch $X_2$ in the 2nd image → S3Esti → Predicting scales and orientations of $X_2$ → Rectified patches of $X_2$

Learning Soft Estimator of Keypoint Scale and Orientation With Probabilistic Covariant Loss. Pei Yan, Yihua Tan, Shengzhou Xiong, Yuan Tai, Yansheng Li. CVPR2022. Poster 4.2, June 24
Affine shape
Harris/Hessian Affine Detector

1. Estimate the shape with the second moment matrix
2. Normalize the affine region to the circular one
3. Go to step 2 if the eigenvalues of the second moment matrix for the new point are not equal


Affine shape adaptation example

After adaptation
AffNet
Learning local descriptor

We want to learn such descriptor, so that patches $A_i, B_i$, corresponding to the same 3D location be near in descriptor space

$$d_{\text{pos}_i} = d(a_i, b_i) \rightarrow \min$$

and far from non-matching.

$$d_{\text{neg}_{i,j}} = d(a_i, a_j) \rightarrow \max$$
AffNet: learning measurement region

**Fig. 5.** AffNet training. Corresponding patches undergo random affine transformation $T_i, \hat{T}_i$, are cropped and fed into AffNet, which outputs affine transformation $A_i, \hat{A}_i$ to an unknown canonical shape. ST – the spatial transformer warps the patch into an estimated canonical shape. The patch is described by a differentiable CNN descriptor. $n \times n$ descriptor distance matrix is calculated and used to form triplets, according to the HardNegC loss.

$$L = \frac{1}{n} \sum_{i=1,n} \max (0, 1 + d(s_i, \hat{s}_i) - d(s_i, N)),$$

$$\frac{\partial L}{\partial N} := 0,$$
We want to learn such descriptor, so that patches $A_i, B_i$, corresponding to the same 3D location be near in descriptor space.

\[ L = \frac{1}{n} \sum_{i=1}^{n} \max(0, m + d_{pos_i} - d_{neg_i}) \]
Why HardNegC loss is needed?

HardNegLoss

<table>
<thead>
<tr>
<th>Avg. positive dist.</th>
<th>0.96</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. min negative dist.</td>
<td>0.27</td>
</tr>
<tr>
<td>Step = 0</td>
<td></td>
</tr>
</tbody>
</table>

HardNegCloss

<table>
<thead>
<tr>
<th>Avg. positive dist.</th>
<th>0.96</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. min negative dist.</td>
<td>0.27</td>
</tr>
<tr>
<td>Step = 0</td>
<td></td>
</tr>
</tbody>
</table>

PosDistLoss

<table>
<thead>
<tr>
<th>Avg. positive dist.</th>
<th>0.96</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. min negative dist.</td>
<td>0.27</td>
</tr>
<tr>
<td>Step = 0</td>
<td></td>
</tr>
</tbody>
</table>
Parameterization matters!

Table 2. Learning the affine transform: parameterization comparison. The average number of correct matches on the HPatchesSeq [37] hardest image pairs 1-6 for the Hessian detector and the HardNet descriptor. Cases compared, affine shape combined with the de-facto handcrafted standard dominant orientation, affine shape and orientation learnt separately or jointly. The match considered correct for reprojection error $\leq 3$ pixels. The HardNegC loss and HardNet descriptor used for learning. n/c – did not converge.

<table>
<thead>
<tr>
<th>Eq.</th>
<th>Matrix</th>
<th>Estimated parameters</th>
<th>biases</th>
<th>Orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>init</td>
<td>jointly</td>
<td>separately</td>
</tr>
<tr>
<td>(1)</td>
<td>$A$</td>
<td>$(a_{11}, a_{12}, a_{21}, a_{22})$</td>
<td>0</td>
<td>n/c</td>
</tr>
<tr>
<td>(1)</td>
<td>$A$</td>
<td>$(a_{11}, a_{12}, a_{21}, a_{22})$</td>
<td>1</td>
<td>n/c</td>
</tr>
<tr>
<td>(2)</td>
<td>$A'$, $R(\alpha)$</td>
<td>$(a'<em>{11}, 0, a'</em>{21}, a'_{22}), (\sin \alpha, \cos \alpha)$</td>
<td>1</td>
<td>250</td>
</tr>
<tr>
<td>(3)</td>
<td>$A''$</td>
<td>$(a''<em>{11}, a''</em>{21}, a''_{22})$</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>(3)</td>
<td>$A''$</td>
<td>$(1 + a''<em>{11}, a''</em>{21}, 1 + a''_{22})$</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>
Both AffNet and second moment matrix method estimate similar shapes.
Both AffNet and second moment matrix method estimate similar shapes but AffNet is more precise.
Example

Warped input patches

Second moment matrix rectification

AffNet rectification

Original patch

Transforms to original patch, should be equal

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.004</td>
<td>-0.247</td>
<td>0.9620</td>
<td>-0.016</td>
</tr>
<tr>
<td>-0.084</td>
<td>1.0165</td>
<td>0.0407</td>
<td>1.038</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.054</td>
<td>0.067</td>
<td>1.078</td>
<td>0.080</td>
</tr>
<tr>
<td>0.247</td>
<td>0.963</td>
<td>0.228</td>
<td>0.944</td>
</tr>
</tbody>
</table>

Robust, good, but not precise
IBR: Affine Invariant Detection

- Take a local intensity extremum as initial point
- Go along every ray starting from this point and stop when extremum of function $f$ is reached

$$f(t) = \frac{|I(t) - I_0|}{\frac{1}{t} \int_0^t |I(t) - I_0| \, dt}$$

We will obtain approximately corresponding regions

Remark: we search for scale in every direction
Affine Invariant Detection

- The regions found may not exactly correspond, so we approximate them with ellipses.
- Geometric Moments:

\[ m_{pq} = \int x^p y^q f(x, y) dx dy \]

Fact: moments \( m_{pq} \) uniquely determine the function \( f \).

- Taking \( f \) to be the characteristic function of a region (1 inside, 0 outside), moments of orders up to 2 allow to approximate the region by an ellipse.

This ellipse will have the same moments of orders up to 2 as the original region.
Affine Invariant Detection

Covariance matrix of region points defines an ellipse:

\[
[x_2, y_2] = A[x_1, y_1]
\]

\[
[x_1, y_1]^T \sum_1^{-1} [x_1, y_1] = 1
\]

\[
\sum_1 = \langle [x_1, y_1][x_1, y_1]^T \rangle_{\text{region}_1}
\]

\[
[x_2, y_2]^T \sum_2^{-1} [x_2, y_2] = 1
\]

\[
\sum_2 = \langle [x_2, y_2][x_2, y_2]^T \rangle_{\text{region}_2}
\]

\[
\sum_2 = A \sum_1 A^T
\]

Ellipses, computed for corresponding regions, also correspond!
The Maximally Stable Extremal Regions

- Consecutive image thresholding by all thresholds
- Maintain list of Connected Components
- Regions = Connected Components with stable area (or some other property) over multiple thresholds selected

The Maximally Stable Extremal Regions

MSER Stability

Properties:
- Covariant with continuous deformations of images
- Invariant to affine transformation of pixel intensities
- Enumerated in $O(n \log \log n)$, real-time computation

MSER+ (bright inside) (in green). The regions ‘follow’ the object (video1, video2).
How to convert MSER to an ellipse?

- One way: to fit an ellipse

- Another way – work with contours: MSER- SAF

Figure 1. Example of stable affine frame construction: (a) each 10th isophote on a part of an image, (b) entry and exit points (white) and the farthest point (green cross) from a bitangent (green lines) constructed on isophotes, (c) SAFs; white lines connecting points $(1,0)^T$, $(0,0)^T$ and $(0,1)^T$ in the frame coordinate system.
MSER-LAF

- Overview of affine-covariant primitives.
  - Rectangular blocks represent regions, detected or derived
  - Elliptical blocks represent the primitives.
### MSER-LAF: many ways of creating affine feature from contour

<table>
<thead>
<tr>
<th>Geometric primitive</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
<th>(g)</th>
<th>(h)</th>
<th>(i)</th>
<th>(j)</th>
<th>(k)</th>
<th>(l)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centre of gravity of region (i)</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Covariance matrix of region (ii)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Curvature minima* (iv)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Curvature maxima* (iv)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tangent points of concavity (v)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Farthest point on the contour (vi)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Farthest point on the concavity (vi)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centre of gravity of concavity (i)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Covariance matrix of concavity (ii)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direction of bitangent (v)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direction CoG to inflection point (vii)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direction of linear segment (viii)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direction from third-order moments (ix)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Object Recognition Using Local Affine Frames on Maximally Stable Extremal Regions, Štěpán Obdržálek & Jiří Matas, 2010

---

(a) [Image]  (b) [Image]  (c) [Image]  (d) [Image]

(e) [Image]  (f) [Image]  (g) [Image]  (h) [Image]

(i) [Image]  (j) [Image]  (k) [Image]  (l) [Image]
MSER-like features: need revamp, or deep reborn

- MSER is the best local feature on the Oxford-Affine benchmark – with planar images.
- It is also one of the worst (among reasonable) on the IMC-Phototourism, because handcrafted segmentation-based detectors are quite fragile.
Overview of WαSH detector. (a) Edges of the input image are (b) non-uniformly sampled. (c) We create the α-filtration of a triangulation over the samples and track the evolution of connected components. (d) We extract features by selecting stable and prominent components.
Algorithm 1: WoSH Feature Detection

\textbf{input} : grayscale image $f$

\textbf{output} : local feature set $F$

\begin{tabular}{ll}
1 $g \leftarrow \| \nabla f \| / \max \{ \| \nabla f \| \}$ & \text{// normalized gradient} \\
2 $E \leftarrow \text{CANNY}(g)$ & \text{// edge detection} \\
3 $P \leftarrow \text{SAMPLE}(E)$ & \text{// edge sampling} \\
4 $\mathcal{R} \leftarrow \text{REGULAR}(P)$ & \text{// regular triangulation} \\
5 $(\mathcal{K}, \rho) \leftarrow \text{COMPLEX}(\mathcal{R})$ & \text{// simplicial complex + sizes} \\
6 $N \leftarrow \text{NEIGHBOR}(\mathcal{K})$ & \text{// neighborhood system} \\
7 $F \leftarrow \emptyset$ \\
8 \textbf{foreach} $\sigma_T \in \mathcal{K}$ \textbf{do} & \text{// initialize each simplex} \\
9 \hspace{1em} $\text{MAKESET}(\sigma_T)$ & \text{// as an individual component} \\
10 \hspace{1em} $\sigma_T.\text{area} \leftarrow \text{AREA}(T)$ & \text{// with its own area} \\
11 \hspace{1em} $\sigma_T.\text{root} \leftarrow \sigma_T$ \\
12 \textbf{foreach} $\sigma_T \in \mathcal{K}$ \textbf{in descending order of} $\rho_T$ \textbf{do} & \text{// current simplex} \\
13 \hspace{1em} $\kappa_T \leftarrow \text{FIND}(\sigma_T)$ & \text{// current component} \\
14 \hspace{1em} $\kappa_T.\text{root} \leftarrow \sigma_T$ \\
15 \hspace{1em} \textbf{foreach} $\sigma_U \in N(\sigma_T)$ such that $\rho_U \geq \rho_T$ \textbf{do} & \text{// adjacent, processed simplex} \\
16 \hspace{2em} $\kappa_U \leftarrow \text{FIND}(\sigma_U)$ & \text{// adjacent component} \\
17 \hspace{2em} $\tau_U \leftarrow \kappa_U.\text{root}$ \\
18 \hspace{2em} \textbf{if} $\kappa_T \neq \kappa_U$ \textbf{then} & \text{// if different components} \\
19 \hspace{3em} \textbf{if} $|U| = 3 \land \tau_U.\text{area}/\rho_T > \tau$ \textbf{then} & \text{// if} $\kappa_U$ \text{ is triangle} \& \text{strong} \\
20 \hspace{4em} $F \leftarrow F \cup \tau_U$ & \text{// select it} \\
21 \hspace{4em} $\tau_T.\text{ADDCHILD}(\tau_U)$ & \text{// add it below} $\kappa_T$ \\
22 \hspace{4em} $\tau_T.\text{area} \leftarrow \tau_T.\text{area} + \tau_U.\text{area}$ & \text{// merge areas} \\
23 \hspace{4em} $\kappa_T \leftarrow \text{UNION}(\kappa_T, \kappa_U)$ & \text{// and disjoint sets} \\
24 \hspace{4em} $\kappa_T.\text{root} \leftarrow \tau_T$
\end{tabular}
Augmentatation-based affine feature: ASIFT

- One can turn any similarity-convariant feature into affine-convariant, using test-time augmentation.
- ASIFT proposes a principled way of doing so.

ASIFT: A New Framework for Fully Affine Invariant Image Comparison
Jean-Michel Morel and Guoshen Yu, SIAM, 2009

ASIFT view sphere sampling

Parameter Sampling

- Geometric sampling of $t$: $\Delta t = t_{k+1}/t_k = \sqrt{2}$.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>$\sqrt{2}$</th>
<th>2</th>
<th>$2\sqrt{2}$</th>
<th>4</th>
<th>$4\sqrt{2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>0°</td>
<td>45°</td>
<td>60°</td>
<td>69.3°</td>
<td>75.5°</td>
<td>79.8°</td>
</tr>
</tbody>
</table>

- Arithmetical sampling of $\phi$: $\Delta \phi = \phi_{k+1} - \phi_k = \Delta \phi = \frac{72^\circ}{t}$.

Perspective view  View from the zenith
Algorithm 1: ASIFT feature computation.

**Input**: Image $u$, tilt sampling step $\delta_t = \sqrt{2}$, tilt sampling range $n = 5$, rotation sampling step factor $b = 72$

**Output**: ASIFT keypoints (referenced by tilt and rotation values)

for $t = 1, \delta_t, \delta_t^2, \cdots, \delta_t^n$ // loop over tilts

do

if $t == 1$ // when $t = 1$ (no tilt), no need to simulate rotation
then

theta = 0 // calculate scale-, rotation- and translation-invariant features on the original image
key(t, theta) = SIFT(u) // C++ routine: compute_sift.keypoints

else

for theta = $0, b/t, 2b/t, \cdots, kb/t$ (such that $kb/t < 180$) // loop over rotations (angle in degrees)

do

$u_r = \text{rot}(u, \theta)$ // rotate image (with bilinear interpolation)

$u_f = G_{0.8t} u_r$ // anti-aliasing filtering ($G_{0.8t}$ is a Gaussian convolution with kernel standard deviation equal to 0.8t). This 1-dimensional convolution is made in the vertical direction, before sub-sampling in the same direction.

$u_t = \text{tilt}(u_f, t)$ // tilt image (subsample in vertical direction by a factor of $t$)

key(t, theta) = SIFT($u_t$) // calculate scale-, rotation- and translation-invariant features. C++ routine: compute_sift.keypoints

Remove the keypoints close to the boundary of the rotated and tilted image support (parallelogram) in $u_t$. The distance threshold is set equal to $6\sqrt{2}$ times the scale of each keypoint.

Normalize the coordinates of the keypoints from the rotated and tilted image $u_t$ to the original image $u$. 


Depth, or AffNet-informed view synthesis

- Instead of sampling whole (half)-view sphere, one can estimate views to generate

- One also could estimate regions, which need rectification. Detected there similarity-convariant features become affine-covariant in original image
Upgrading point correspondences to affine
Similarity feature correspondences are noisy affine correspondences

Selected matching SIFT keypoints and corresponding patches. Not only patch centers correspond to each other, but also other pixels, although less precise. Image pair from Sacre Coeur IMW dataset.
Let's take tentative correspondence and feed into CNN to get more precise affine correspondence.
MVO: split patch into 9 subpatches

- Instead of estimation of affine map directly, one can estimate few point correspondences.

**Fig. 3. Coarse-to-fine refinement and qualitative examples.** *Left:* We start by a coarse alignment at feature extraction resolution taking into account only the central point, followed by a fine refinement on sub-patches corresponding to each grid point. *Right:* The first and last columns show the source and the target patch, respectively. The 3 × 3 regular grid is plotted with circles. For the target patch, we plot the deformed grid predicted by the coarse-to-fine refinement with crosses. The middle column shows the warped target patch using bilinear interpolation in between grid locations.
Summary

- A point correspondence is never *just* a point correspondence
- There are many good methods to get affine correspondence, and even more of them are waiting for revival in a deep learning era
- If you are interested in the topic of the Affine Correspondences, there is a high probability, that our team at HOVER Inc. would be interested in you – for doing state-of-the-art 3D reconstruction from a few images. Let’s chat 😊