Visual Object Tracking

Jianan Wu
Megvii (Face++) Researcher
wjin@megvii.com
Dec 2017
Applications

• From image to video:
  • Augmented Reality
  • Motion Capture
  • Surveillance
  • Sports Analysis
  • ……
Wait. What is visual tracking?

• When we talk about visual tracking, we may refer to something completely different.

• Main topics covered in this lesson:
  1. Motion estimation / optical flow
  2. Single object tracking
  3. Multiple object tracking

• We will also glance at other variants:
  • fast moving, multi-camera, …
Outline

1. **Motion Estimation / Optical Flow**
2. Single Object Tracking
3. Multiple Object Tracking
4. Other
Motion Field

• The projection of the 3D motion onto a 2D image.
• However, the true motion field can only be approximated based on measurements on image data.
Optical Flow

- Optical flow: the pattern of apparent motion in images.
  - Approximation of the motion field
  - Usually adjacent frames
  - Pixel level
  - Either dense or sparse

\[ I(x, y, t) = I(x + dx, y + dy, t + dt) \]
Motion Field \approx Optical Flow

- Not always the same.

- Such cases are unusual. In most cases we will assume that optical flow corresponds to the motion field.

Image from: Gary Bradski slides
Kanade-Lucas-Tomasi Feature Tracker

• Steps:
  1. Find good feature points
     • E.g. Shi-Tomasi corner points
  2. Calculate optical flow
     • Lucas-Kanade method (Assume all the neighboring pixels have similar motion)
       the local image flow (velocity) vector \((V_x, V_y)\) must satisfy
       
       \[
       \begin{align*}
       I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\
       I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2) \\
       &\vdots \\
       I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n)
       \end{align*}
       \]
       where \(q_1, q_2, \ldots, q_n\) are the pixels inside the window, and \(I_x(q_i), I_y(q_i), I_t(q_i)\) are the partial derivatives of the image \(I\) with respect to position \(x, y\) and time \(t\), evaluated at the current time.

• Free Implementations: [http://cecas.clemson.edu/~stb/klt/](http://cecas.clemson.edu/~stb/klt/)
• Also available in OpenCV


Kanade-Lucas-Tomasi Feature Tracker
Optical Flow with CNN

• FlowNet / FlowNet 2.0
  • Learn optical flow directly from image pairs.
  • Lack of training data? Let’s synthesize!
    • Flying Chairs / ChairsSDHom
    • Flying Things 3D
  • Train with simple datasets first.
  • Combine multiple FlowNets for large displacement

• https://github.com/lmb-freiburg/flownet2

FlowNet: Structure

FlowNetS

FlowNetC
Optical Flow: Summary

• Establishing point to point correspondences in consecutive frames of an image sequence.

• Issues:
  • Missing concept of object
  • Large displacement handling
  • Occlusion handling
  • Failure (assumption validity) not easy to detect
Outline

1. Motion Estimation / Optical Flow
2. **Single Object Tracking**
3. Multiple Object Tracking
4. Other
Single Object Tracking

• Single object, single camera
• Model free:
  • Nothing but a single training example is provided by the bounding box in the first frame
• Short term:
  • Tracker does not perform re-detection
  • Fail if tracking drifts off the target
• Subject to Causality:
  • Tracker does not use any future frames
Single Object Tracking

- Protocol:
  Setup tracker
  Read initial object region and first image
  Initialize tracker with provided region and image
  loop
    Read next image
    if image is empty then
      Break the tracking loop
    end if
    Update tracker with provided image
    Write region to file
  end loop
  Cleanup tracker

Luka Čehovin, TraX. “The visual Tracking eXchange Protocol and Library”. Neurocomputing. 2017
Correlation Filter

denotes VOT baseline experiment
denotes VOT realtime experiment

https://github.com/foolwood/benchmark_results
Correlation Filter

- **Cross-correlation:**
  - Cross-correlation is a measure of similarity of two series as a function of the displacement of one relative to the other

\[
(f \star g)(\tau) \overset{\text{def}}{=} \int_{-\infty}^{\infty} f^*(t) g(t + \tau) \, dt
\]

- Similar to convolution

```
\begin{array}{ccc}
\text{Convolution} & \text{Cross-correlation} \\
\hline
f & g & f^* \star g \\
\hline
\end{array}
```
Convolution Theorem

- Cross-correlation is equivalent to an element-wise product in Fourier domain.

\[ g = f \star h \iff \mathcal{F}(g) = \mathcal{F}^*(f) \odot \mathcal{F}(h) \]

where \( \mathcal{F}(g) \) is the Discrete Fourier Transform (DFT) of \( g \) (likewise for \( f \) and \( h \))

\( \star \) is cross-correlation and \( \odot \) means element-wise product

\( * \) is complex-conjugate

- Computation is much faster in Fourier Domain: \( O(P^2) \rightarrow O(P \log P) \), \( P \) is number of pixels in tracking window.
Minimum Output Sum of Squared Error Filter

• Initialization: learn filter by applying random small affine perturbations to the tracking window for the first frame of the video

\[ \min_{H^*} \sum_i |F_i \odot H^* - G_i|^2 \]

where \( F_i \) is training image (gray scale, weighted by cosine window), \( G_i \) is generated from ground truth: a compact 2D Gaussian shaped peak centered on the target in training image \( F_i \).

• Closed form solution:

\[ H^* = \frac{\sum_i G_i \odot F_i^*}{\sum_i F_i \odot F_i^*} \]

Minimum Output Sum of Squared Error Filter

- Filter Update: \( H_i^* = \frac{A_i}{B_i} \)
  \[
  A_i = \eta G_i \odot F_i^* + (1 - \eta) A_{i-1}
  
  B_i = \eta F_i \odot F_i^* + (1 - \eta) B_{i-1}
  \]

where \( \eta \) is learning rate

- Simple
- Extremely Fast! 600+ fps
Discriminative Tracking

• Tracking by Detection
Kernelized Correlation Filter

- Circulant matrices

\[ X = C(x) = \begin{bmatrix}
  x_1 & x_2 & x_3 & \cdots & x_n \\
  x_n & x_1 & x_2 & \cdots & x_{n-1} \\
  x_{n-1} & x_n & x_1 & \cdots & x_{n-2} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  x_2 & x_3 & x_4 & \cdots & x_1
\end{bmatrix}. \]

- Diagonalization:

\[ X = F \text{diag} \left( \hat{x} \right) F^H \]

- Where \( F \) is a constant matrix (DFT matrix) that does not depend on \( x \), and \( \hat{x} = \mathcal{F}(x) \)

Kernelized Correlation Filter

- Ridge regression as classifier:
  \[
  \min_w \sum_i (w^T x_i - y_i)^2 + \lambda \|w\|^2 \quad \Rightarrow \quad w = (X^T X + \lambda I)^{-1} X^T y
  \]

  We can replace \( X = C(x) \) (circulant data), and \( y = g \) (Gaussian targets):

- The regression target of the additional samples follow a Gaussian function, which takes a value of 1 for a centered target and smoothly decays to 0 for any other shifts, according to a bandwidth \( s \).
  \[
  \hat{w} = \frac{\hat{x}^* \odot \hat{y}}{\hat{x}^* \odot \hat{x} + \lambda}
  \]

- Exactly the MOSSE solution!
- Good learning algorithm with lots of data.
Kernelized Correlation Filter

- Circulant matrices are very general tool which allows to replace standard operations with fast Fourier operations.
- Kernel Ridge Regression as classifier:
  - with $K$ kernel matrix $K_{ij} = \kappa(x_i, x_j)$ and dual space representation
    \[
    w = \sum_i \alpha_i \varphi(x_i) \quad \varphi^T(x)\varphi(x') = \kappa(x, x'), \quad K_{ij} = \kappa(x_i, x_j)
    \]
    \[
    \Rightarrow \quad \alpha = (K + \lambda I)^{-1} y
    \]
- For many kernels, circulant data $\Rightarrow$ circulant $K$ matrix
  \[
  K = C(k^{xx})
  \]
- $k^{xx}$ is the first row of the kernel matrix
  \[
  \hat{\alpha} = \frac{\hat{y}}{\hat{k}^{xx} + \lambda}
  \]
Kernelized Correlation Filter

The $k^{xx'}$ is kernel correlation of two vectors $x$ and $x'$

$$k^{xx'}_i = \kappa(x', P^{i-1}x)$$

For Gaussian kernel it yields:

$$k^{xx'} = \exp\left(-\frac{1}{\sigma^2} (\|x\|^2 + \|x'\|^2 - 2\mathcal{F}^{-1}(\hat{x}^* \odot \hat{x}'))\right)$$

Evaluation on subwindows of image $z$ with classifier $\alpha$ and model $x$:

$$K^z = C(k^{xz})$$

$$f(z) = \mathcal{F}^{-1}(k^{xz} \odot \hat{\alpha})$$

Update classifier $\alpha$ and model $x$ by linear interpolation from the location of maximum response $f(z)$

Kernel allows integration of more complex and multi-channel features
Kernelized Correlation Filter

very few
hyperparameters
code fits on one slide!

Use HoG features
(32 channels)

~300 FPS

Open-Source
(Matlab/Python/Java/C)

Algorithm 1: Matlab code, with a Gaussian kernel. Multiple channels (third dimension of image patches) are supported. It is possible to further reduce the number of FFT calls. Implementation with GUI available at:

http://www.isr.uc.pt/~henriques/

Inputs
- \( x \): training image patch, \( m \times n \times c \)
- \( y \): regression target, Gaussian-shaped, \( m \times n \)
- \( z \): test image patch, \( m \times n \times c \)

Output
- responses: detection score for each location, \( m \times n \)

function alphaf = train(x, y, sigma, lambda)
    k = kernel_correlation(x, x, sigma);
    alphaf = fft2(y) ./ (fft2(k) + lambda);
end

function responses = detect(alphaf, x, z, sigma)
    k = kernel_correlation(z, x, sigma);
    responses = real(ifft2(alphaf.*fft2(k)));
end

function k = kernel_correlation(x1, x2, sigma)
    c = ifft2(sum(conj(fft2(x1)) .* fft2(x2), 3));
    d = x1(:)*x1(:) + x2(:)*x2(:) - 2 * c;
    k = exp(-1 / sigma^2 * abs(d) / numel(d));
end

slides material by João F. Henriques
From KCF to Discriminative CF Trackers

• Martin Danelljan et al. – DSST
  • PCA-HoG + grayscale pixels features
  • Filters for translation and for scale (in the scale-space pyramid)

• Li et al. – SAMF
  • HoG, color-naming(CN) and grayscale pixels features
  • Quantize scale space and normalize each scale to one size by bilinear inter.

• Martin Danelljan et al. – SRDCF
  • Spatial regularization in the learning process
    • limits boundary effect
    • penalize filter coefficients depending on their spatial location
  • Allow to use much larger search region
  • More discriminative to background (more training data)

• Martin Danelljan et al. – Deep SRDCF
  • CNN features

\[ \varepsilon = \| g - \sum_{l=1}^{d} h^l \ast f^l \|^2 + \lambda \sum_{l=1}^{d} \| h^l \|^2 \]

Sample weights

\[ S_f(x) = \sum_{l=1}^{d} x^l \ast f^l. \]
Continuous-Convolution Operator Tracker

- Multi-resolution CNN features

Continuous-Convolution Operator Tracker

\[ E(f) = \sum_{j=1}^{m} \alpha_j \| S_f \{ x_j \} - y_j \|^2 + \sum_{d=1}^{D} \| w f^d \|^2 \]

\[ S_f \{ x \} = \sum_{d=1}^{D} f^d \ast J_d \{ x^d \}, \quad x \in \mathcal{X} \]

- Interpolation operator

- Optimized
- Implementation: https://github.com/martin-danelljan/Continuous-ConvOp
- Very Slow, ~ 1fps
- A lot of parameters, easy to overfitting
Efficient Convolution Operators

- Based on C-COT
- Main Improvements:
  1. Introduce a factorized convolution operator that dramatically reduces the number of parameters in the DCF model.
  2. A Gaussian mixture model to reduce the number of samples in the learning, while maintaining their diversity.
  3. Only optimize every N frames for faster tracking.

- Implementation: [https://github.com/martin-danelljan/ECO](https://github.com/martin-danelljan/ECO)
- ~ 15 FPS on GPU

Deep Learning

https://github.com/foolwood/benchmark_results
Multi-Domain Convolutional Neural Network Tracker

• A multi-domain learning framework based on CNNs

Hyeonseob Nam, Bohyung Han. “Learning Multi-Domain Convolutional Neural Networks for Visual Tracking”. CVPR. 2016
Multi-Domain Convolutional Neural Network Tracker

• Online tracking:
  • Replace fc1-fc6 to a single branch with random initialization
  • Sample positive (iou>0.7) and negative (iou<0.5) samples for online training
  • Multi scale target candidate samples from Gaussian

• Hard minibatch mining
• Bounding box regression
• ~ 1 fps

https://github.com/HyeonseobNam/MDNet

Algorithm 1 Online tracking algorithm

<table>
<thead>
<tr>
<th>Input</th>
<th>Pretrained CNN filters {w₁, ..., w₅}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>Estimated target states x^t_i</td>
</tr>
</tbody>
</table>

1: Randomly initialize the last layer w₆.
2: Train a bounding box regression model.
3: Draw positive samples S¹⁺ and negative samples S¹⁻.
4: Update \{w₄, w₅, w₆\} using S¹⁺ and S¹⁻.
5: \(T_s \leftarrow \{1\}\) and \(T_l \leftarrow \{1\}\).
6: repeat
7: Draw target candidate samples x^t_i.
8: Find the optimal target state x^+_i by Eq. (1).
9: if \(f^+(x^+_i) > 0.5\) then
10: Draw training samples S¹⁺ and S¹⁻.
11: \(T_s \leftarrow T_s \cup \{t\}, T_l \leftarrow T_l \cup \{t\}\).
12: if |\(T_s\)| > \(\tau_s\) then \(T_s \leftarrow T_s \setminus \{\min_{v \in T_s} v\}\).
13: if |\(T_l\)| > \(\tau_l\) then \(T_l \leftarrow T_l \setminus \{\min_{v \in T_l} v\}\).
14: Adjust x^+_i using bounding box regression.
15: if \(f^+(x^+_i) < 0.5\) then
16: Update \{w₄, w₅, w₆\} using \(S^+_{v \in T_s}\) and \(S^-_{v \in T_s}\).
17: else if \(t \mod 10 = 0\) then
18: Update \{w₄, w₅, w₆\} using \(S^+_{v \in T_l}\) and \(S^-_{v \in T_l}\).
19: until end of sequence
GOTURN

- Simple and no online model update
- ~ 100 fps

SiameseFC

• A deep FCN is trained to address a more general similarity learning problem in an initial offline phase
• Training from ImageNet Video dataset
  • >> online learning methods
• No online model update
• [https://github.com/bertinetto/siamese-fc](https://github.com/bertinetto/siamese-fc)
• ~ 60 fps

SiameseFC
Benchmark

denotes VOT baseline experiment

denotes VOT realtime experiment

https://github.com/foolwood/benchmark_results
Benchmark: VOT

- http://www.votchallenge.net/index.html

- VOT 2017:
  - 60 sequences (50 from VOT 2016 and 10 new)
  - An additional sequestered dataset for top trackers.
Evaluation Metrics: VOT

• Accuracy:
  • Average overlap during successful tracking

• Robustness:
  • Number of times a tracker drifts off the target

• Expected Average Overlap (EAO):

\[ \Phi_{i} : \text{average of per-frame overlaps} \]

\[ \Phi_{Ns} = \frac{1}{N_s} \sum_{i=1:N_s} \Phi_{i} \]

\[ \hat{\Phi}_{Ns} = \langle \Phi_{Ns} \rangle \]

\[ \hat{\Phi} = \frac{1}{N_{hi} - N_{lo}} \sum_{N_s=N_{lo}:N_{hi}} \Phi_{Ns} \]


Benchmark: OTB

- OTB:
  - OTB2013
  - TB-100, OTB100, OTB2015
    - TB-50, OTB50: 50 difficult sequences among TB-100
Evaluation Metrics: OTB

• One Pass Evaluation (OPE):
  • Run tracker throughout a test sequence initialized by ground truth bounding box in the first frame and return the average precision.

• Spatial Robustness Evaluation (SRE):
  • Run tracker throughout a test sequence with initialization from 12 different bounding boxes by shifting or scaling ground truth in the first frame and return the average precision.

Results of TB-100

[Graph showing precision plots of OPE]

- ECO [0.910]
- MDNet [0.909]
- CCOT [0.898]
- MCPF [0.873]
- ECO-HC [0.856]
- DeepSRDCF [0.851]
- HDT [0.848]
- CF2 [0.837]
- SRDCFdecon [0.825]
- CNN-SVM [0.814]
- SRDCF [0.789]
- Staple [0.784]
- MEEM [0.781]
- CFNet_conv3 [0.777]
- SiamFC_3s [0.771]
- LCT [0.762]
- SAMF [0.751]
- KCF [0.696]
- DSST [0.680]

https://github.com/foolwood/benchmark_results
Results of VOT2017

Outline

1. Motion Estimation / Optical Flow
2. Single Object Tracking
3. **Multiple Object Tracking**
4. Other
Multiple Object Tracking

• For each frame in a video, localize and identify all objects of interest, so that the identities are consistent throughout the video.

• Compared to single object tracking:
  • Target is not given in the first frame.
  • Classes of targets are known and models are always trained offline.
  • Long term: detection can be done whenever necessary.
  • Online and offline tracking are both available.
  • The number of objects is unknown.
  • The number of objects may change.

• Example:
  tracking all the persons in the video
Tracking by Detection

- For each frame, first localize all objects using an object detector.
- Associate detected objects between frames.
- Make multiple object tracking to be a association problem more than a tracking problem.
- Association based on location, motion, appearance and so on.
Location

• Intersection over union (IOU):
  \[ \text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \]

  • Problem: lack of discriminability if \( \text{iou} == 0 \)
  • Sometimes we use intersection over minimum (IOM)

• L1/L2 distance
  • Problem: related to object’s shape and camera’s parameters.
  • Better to convert into world coordinate if possible.
Motion

• Modeling the movement of objects.
• Kalman filter:
  • Using Kalman filter is a way of optimally estimating the state of a linear dynamical system.
  • A possible state space: center position \((x, y)\), aspect ratio \(a\), height \(h\) and their respective velocities of the bounding box.
  • Use detection result as observation.
Appearance

• Techniques in single object tracking like cross correlation and SiameseFC can be used here
• Hand-crafted features like histograms and color names
• CNN features
  • For pedestrian tracking, we can use reid feature
Association

• Location, motion and appearance features need to be combined.
  • Different weights in different applications

• Three kinds of assignments
  • Detection – Detection
  • Trajectory – Detection
  • Trajectory – Trajectory

• Do not trust the detector!
  • FP and FN of the detector make the association even more difficult.

• Tune your tracker according to your detector.
  • It is good for association to understand the mistakes your detector often makes.
Association as Optimization

- **Local method:**
  - Hungarian algorithm (Kuhn-Munkres algorithm)

- **Global methods:**
  - Clustering
  - Network flow
  - Minimum cost multi-cut problem
  - ……

- Global optimization for a whole video is impractical if there are too many objects.
  - Merging nearby bounding boxes together to get reliable tracklets.

- To trade off speed against accuracy, we can do optimization in a window.
Network Flow

- Each detection $d_i$ is represented with two nodes connected by an edge (red)
- Add a source node and a sink node represent appearing and disappearing
- Every edge has a cost $c$
- Every edge has a binary flow variable $x$ means if they belong to the same trajectory
- Relax the binary constraint on $x$ to solve the problem

$$x^* = \arg \min_x c^T x$$

s.t. $Ax \leq b$, $Cx = 0,$

$$A = [I, -I]^T \in \mathbb{R}^{2M \times M}$$ and $b = [1, 0]^T \in \mathbb{R}^{2M}$

$$x_i^{in} + \sum_j x_{ji}^{link} = x_i^{det}$$

$$x_i^{out} + \sum_j x_{ij}^{link} = x_i^{det} \forall i$$

State Transition

- a6: long term tracking, do interpolation if necessary.
- a7: objects lost for more than specified frames will no longer be considered.

Benchmark

• **MOT**
  • [https://motchallenge.net](https://motchallenge.net)
  • Pedestrian tracking
  • 7 training videos and 7 test videos

• **KITTI**
  • [http://www.cvlibs.net/datasets/kitti/eval_tracking.php](http://www.cvlibs.net/datasets/kitti/eval_tracking.php)
  • Car and pedestrian tracking

• **ImageNet VID**
  • 30 classes
Evaluation Metrics:

- Multiple object tracking accuracy (MOTA):
  - It is a combination of three errors:

\[
MOTA = 1 - \frac{\sum_t (fn_t + fp_t + id_sw_t)}{\sum_t g_t}
\]

where \(id_sw_t\) means number of identity switches in frame \(t\)

Summary

• “Visual object tracking” is not a single problem, but a series of problems.
• The area is just starting to be affected by CNNs.
• Key components of trackers:
  • Representation of object’s appearance, location and motion
  • Integration with detection
• Speed is very important for real applications.
Outline

1. Motion Estimation / Optical Flow
2. Single Object Tracking
3. Multiple Object Tracking
4. Other
World of Fast Moving

• Fast Moving Object (FMO)
  • An object that moves over a distance exceeding its size within the exposure time

Multiple Camera Tracking

- Tracking between cameras
  - Cameras may have overlap
  - Time of cameras need to be synchronized
  - Calibration of cameras
Tracking with Multiple Cues

• With multiple detectors:
  • Head + pedestrian detector for pedestrian tracking

• With key points:
  • Skeleton for pedestrian tracking
  • Landmark for face tracking

• With semantic segmentation
  • Semantic optical flow

• With RGBD camera
Crowds Tracking

Multiple Object Tracking with NN

Multiple Object Tracking with NN


Figure 2: The architecture of the proposed Quad-CNN for multi target tracking using temporal coherency.
Thank You!

Q&A

"... Although tracking itself is by and large a solved problem..."

-- Jianbo Shi & Carlo Tomasi, CVPR 1994