In the name of God, the Compassionate, the Merciful

ADVANCED COMPUTER VISION

2D Tracking

Introduction

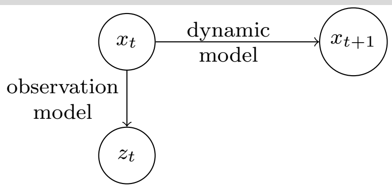
Detection VS Tracking:

Detection: Estimate X at an instant.
✓ Exhaustive. ✗ Inefficient. ✗ Dataassociation.
Tracking: Maintains an estimate of X overtime & predict future location.
✓ Efficient. ✓ Smoothes. ✗ Assumptions about object behavior.

Approaches to Tracking:

Sequential: Recursive, Online.
✓ Inexpensive → real-time. ✗ No future information. ✗ Cannot revisit past errors.
Batch processing: Offline.
✗ Inexpensive → not real-time. ✓ Considersall information. ✓ Can correct past errors.
Parallel trackers: Several single-object trackers.
✓ Computationally less expensive. ✗ Ad-hoc interaction.
Joint state: Single multi-object representation.
✗ Computationally expensive. ✓ Explicit principled interaction.
Non-probabilistic: Mean shift, gradient descent, least squares, AI (agents).
✓ Quick convergence. ✓ Efficient. ✗ Stuck in local max/min. ✗ Modeling multiple objects.
Probabilistic: Kalman & particle filter, Bayes net inference, kernel density estimation, relevance vector machine, principled.
✓ Multi-modal. ✗ Slower. ✗ Interpretation.
Tracking Challenges: Appearance change, Occlusion, Distraction, Illumination, Difficult motion, Multiple objects, Scale change.

Bayesian Filtering



Dynamic Model:	$x_t = F_t x_{t-1} + w_t$ $w_t \sim N(0, Q_t)$ $p(x_t\|x_{t-1}) = N(F_t x_{t-1}, Q_t)$
Observation Model:	$z_t = H_t x_t + v_t$ $v_t \sim N(0, R_t)$ $p(z_t\|x_t) = N(H_t x_t, R_t)$
Posterior:	$p(x_t\|z_t) = N(x_{t\|t}, P_{t\|t})$

Recursive Bayesian Filtering:

- $p(x_t|z_t) \propto p(z_t|x_t) \int_{x_{t-1}} p(x_t|x_{t-1}) p(x_{t-1}|z_{t-1})$

Kalman Filter

Prediction:	$\hat{x}_{t\|t-1} = F\hat{x}_{t-1\|t-1}$ $P_{t\|t-1} = F_t P_{t-1\|t-1} F_t^T + Q_{t-1}$
Measurement residual:	$\tilde{y}_t = z_t - H_t \hat{x}_{t\|t-1}$ $S_t = H_t P_{t\|t-1} H_t^T + R_t$
Kalman gain:	$K_t = P_{t\|t-1} H_t^T S_t^{-1}$
Correction:	$\hat{x}_{t\|t} = \hat{x}_{t\|t-1} + K_t \tilde{y}_k$ $P_{t\|t} = I - K_t H_t P_{t\|t-1}$

Particle Filters

Posterior: $p(x_{t-1}|z_{t-1}) = \sum_{n=1}^{N} w_{t-1}^{(n)} \delta(x_{t-1} - x_{t-1}^{(n)})$

other topics

Section 8: Numerical Optimization

- **General Optimization Methods:**
  - Gradient/Steepest Descent: Move through the negative direction of the gradient vector to find minima locations.
  - Conjugate Gradient: Gradient descent with optimal λ.
- **Weaknesses:** How to ▸initialize the algorithm? ▸choose the step sizeλ? ▸manage lots of iterations in long & narrow valleys?
- **NOTE:** ▸Linear estimation algorithm is fast, but sensitive to noise. ▸Iterative nonlinear estimation is slower, but more precise. ▸Linear algorithm can be used to initialize nonlinear ones.
- **Nonlinear Least-Squares:**
  - Newton-Raphson: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
  - Gauss-Newton (GNA): $p_{i+1} = p_i + \Delta_i$
    $$\Delta_i = \min_{\Delta} \parallel f(p_i + \Delta) - b \parallel^2$$
  - Levenberg-Marquardt: ▸interpolates between (GNA) & gradient descent method. ▸more robust but bit slower than GNA.

Some robust objective functions:

- Maximum-likelihood estimator (M-estimator): reduce the effect of outliers by replacing distance by another function to down weight outliers. ✗ Handles few percents of outliers. ✗ Requires a threshold.
- RANSAC: ✓ Handles more than 50% outliers. ✗ Requires a threshold.
- Least-Median-of-Squares (LMedS) estimator: ✓Handles 50% outliers. ✓ Does not require a threshold. ✓ can be speeded up considerably by means of parallel computing

Homography RANSAC loop:

1. Select 4 feature pairs (at random).
2. Compute the homography H (exact).
3. Compute inlierswhere $\parallel p_i', Hp_i \parallel < \epsilon$.
4. Keep the largest set of inliers.
5. Re-compute the least-squares H estimate, using all of inliers.

Epipolar RANSAC loop:

1. Select at random a set of 8 successful matches.
2. Compute the fundamental matrix F_k.
3. Determine the subset of inliers (Sampson distance).
4. Count the number of points in the consensus set.

Robustness:

- the number of model parameters (minimal correspondences needed): Bad exponential
- the percentage of inliers: Base of the exponential
- the number of iterations: Good exponential

$(1 - G^P)^N$ G:Proportion of inliers, Model needs P pairs, N iterations

PROSAC: we have a measure of confidence for each measurement

Section 10: Segmentation of Dynamical Scenes

Kmeans Algorithm

1. Set, iteration=1;
2. Choose randomly K-means $m_1, \cdots, m_k$;
3. For each data point x_i, compute the distance to each of the means & assign it to the cluster with the nearest mean.
4. Iteration=iteration+1;
5. Recompute the means based on new assignments of points to clusters (adjust means);
6. Repeat Steps 3-5 until the cluster centers converge (do not change much).

- **Pros:** ✓ K-means is computationally efficient. ✓ Gives good results if the clusters are compact.
- **Cons:** ✗ Memory-intensive. ✗ Choice of K & initial partitions. ✗ Sensitive to initialization stage. ✗ Sensitive to outliers. ✗ Only finds "spherical" clusters. ✗ Suffers from problems of local minima.

Minkowski metric (or L_p norm):

$$L_p(x, y) = (\sum_{i=1}^{d} |x_i - y_i|^p)^{1/p}$$

▸L_2: Euclidean, L_1=Manhattan (or block city)

KNN

non-parametric method, dinstance-based learning, or lazy learning.
KNN requires: ▸An integer K. ▸training dataset ▸A metricto measure closeness.
large values of K: ✓ Yields smoother decision regions. ✓ Provides probabilistic information. ✓ The ratio of examples for each class gives information about the ambiguity of the decision.
K too large: ✗ Destroys the locality of the estimation. ✗ Increases the computational burden.

- **Pros:** ✓ Analytically tractable. ✓ Simple implementation. ✓ Nearly optimal in the large sample. ✓ Uses local information ✓ easy to parallel implementations.
- **Cons:** ✗ Large storage requirements. ✗ Computationally intensive recall. ✗ Sensitive to the local structure of the data. ✗ Highly susceptible to the curse-of-dimensionality.

EM

EM is an iterative method for finding ML, or MAP, estimates of parameters in statistical models.

- **Pros:** ✓ Probabilistic interpretation. ✓ Soft assignments between data points & clusters. ✓ Generative model.✓ Relatively compact storage.
- **Cons:** ✗ Local minima. ✗ Needs an initial guess of the parameters.(Often good idea to start with a K-means clustering). ✗ Needs to know the number of components. ✗ There can be numerical problems.

Mean Shift

kernel function properties: ▸Integrate to 1. ▸symmetric. ▸maximum at 0. ▸Decay quickly to zero. ▸Extent of the kernel be the same along all dimensions.

Algorithm

1. Define the kernel K at each data point.
2. Sum up the result into a single function.
$$f(X) = 1/N \sum_i K(X - X_i)$$
3. Move in the direction of mean shift vector to converge to the closest mode.
$$X \leftarrow X + M(X) = \frac{\sum_i X_i g(\|X-X_i\|^2/h^2)}{\sum_i g(\|X-X_i\|^2/h^2)}$$

Kernel Bandwidth h:

- Too small → overfits the data points.
- Too large → smoothes out the details of data.

Section 11: Keypoint Description

Canny Edge Detector:

i Apply a Gaussian filter.
ii Find the magnitude & direction of the gradient (Sobel).
iii Apply nonmaxima suppression.
iv Apply two thresholds (hysteresis):

Harris: ▸localize the point whitch shifting a window in any direction should give a large change in intensity. ▸corner points has two large positive eigenvalues. $Tr(H) = I_{xx} + I_{yy} = \lambda_1 + \lambda_2, Det(H) = I_{xx}I_{yy} - I_{xy}^2 = \lambda_1\lambda_2$

Eigen values of A^{-1} define the error ellipses (Förstner)

Harris-Laplacian: Find local maximum of Laplacian in scale.

Hessian $\frac{Tr(H^2)}{Det(H)} = \frac{(r+1)^2}{r}$, r=eigenvalue ratio

Harris-Affine Detector Algorithm: **i.**Identify initial region points using scale-invariant Harris-Laplace detector. **ii.**For each initial point, normalizethe region to be affine invariant using affine shape adaptation. **iii.**Iteratively estimate the affine region: ▸Selection of proper integration scale, differentiation scale, & spatially localized interest points. **iv.**Updatethe affine region using these scales & spatial localizations. **v.**RepeatStep 3, if the stopping criterion is not met.

Maximally Stable Extremal Region (MSER)

Connected component of thresholded image

Local Affine Frame (LAF) Assumptions: ▸Local planarity. ▸Perspective camera.

Comparison to other Region Detectors:

- Region density: Offers the most variety detection.
- Region size: Detects many small regions
- Viewpoint change: Outperforms the other region detectors.
- Scale change: 2^{nd} under a scale change & in-plane rotation after Hessian-affine
- Blur: The most sensitive to this type of change in image.
- Light change: Shows the highest repeatability score.

Hessian-Affine=textured scenes, corner-like parts, structured scenes. MSER=well-structured (segmentable)

Förstner

▸Detects line crossing (corner, junction). ▸Center of circular structures.

Algorithm:

i Find region where the autocorrelation is of rank 2.
ii Compute the form of error ellipses (w, q).
size of ellipse $w = \frac{det(A)}{trace(A)}$ shape of ellipse $q = \frac{4det(A)}{trace(A)^2}$
iii Apply non-maximal suppression.
iv Compute Förstner points.

Scale Invariant Feature Transform (SIFT)

LoG: Detects blobs if the convolution scale matches the size of the blob.

keypoint detection:

1. Super sample original image.
2. Compute smoothed images using different scales for entire octave.
3. Compute DoG images from adjacent scales for entire octave.
4. Subsample image 2σ of current octave & repeat Steps 2 & 3 for the next octave.
5. Isolate keypoints in each octave by detecting extremain DoG compared to neighboring pixels.

Speeded Up Robust Feature (SURF)

SURF: It uses the sum of Haar wavelet response around the point of interest. these can be computed with the aid of integral image.

HOG

Describes the local object appearance & shape by using the distribution of intensity gradients

Algorithm: Input Images → Normalize gamma → Compute gradients → Vote weights in spatial & orientation cells → Normalize contrast in overlapping spatial cells (Blocks: R-HOG, C-HOG) → HoG Features

HoG Descriptor Parameters: Gradient scale, Orientation bins, Block overlap percentage [other: Normalization method, Transformation Functions, scale-space pyramid space]

GIST

GIST: Apply oriented Gabor filters over different scales. Average filter energy in each bin. (16 Bins) x (8 Orientations) x (4 Scales)=512

The procedure is based on a very low dimensional representation of the scene called ”spatial envelope”.

Gist properties:

- Invariant to: Luminance transformations, blur, resize, etc.
- Not invariant to: Translation, rotation, occlusion, crop, etc.
- Distance measure: L2 distance to compare, NN-search.

Applications: Scene recognition, Copy detection, Depth estimation, Image classification, Scene completion (inpainting), Robot navigation

Section 12 Keypoint Matching

- **Blobs** are characterized by connected components of contours.
- Factors leading to l**arge costs** in **correlation matching**: ✗ Image is much larger than template. ✗ We might have many templates. ✗ orientation. ✗ scale.
- **Reducing the Cost:** Reducing the number of ✓ image windows ✓ database objects (index templates by features such as moments, Measure moments of candidate windows, Only match ”similar” templates.) ✓ operations (1-Reduce the number of pixels: Multiresolution, Principal component. 2-Match a subset of M against a subset of N: random, boundary)
- **Chamfer matching** is the correlation between a binary edge template & the distance transform.
- **Distance Transform:** Each pixel has a (Manhattan) distance to the nearest edge pixel.
- Global templates are **sensitive** to: ✗ Partial occlusions. ✗ Non-rigid deformations. ✓ **solution:** Constellation of local edge fragments.
- **Hausdorff** distance is the greatest of all distances from a point in one set to the closest point in the other set.

KD-Tree

- **Construction of KD-Tree**
  1. Find the dimension of maximum variation.
  2. Split the data on its median/mean value (equal partition).
  3. Repeat the procedure.
- **KD-Tree NN Search**
  1. Start with the root node.
  2. Once you reach a leaf node, save that node point as the ”current best”.
  3. At each node:
     i If the current node is closer than the current best, consider it as the current best.
     ii Check if there is any points on the other side of the splitting plane that are closer to the search point than the current best.
        a If the hyper sphere (with a radius equal to the current nearest distance) crosses the plane, there could be nearer points on the other side of the plane:
        - Move down the other branch of the tree from the current node looking for closer points.

b If the hyper sphere does not intersect the splitting plane, continue walking up the tree.
-Eliminate the entire branch on the other side of that node.

4. Finish this process (for the root node) when the search is complete.

- **Best-Bin-First Search** Key ideas:
  - Search KD-tree bins in the order of distancefrom query. (Requires use of a priority queue)
  - Search afixed number of neighboring KD-tree bins. (Only an approximate NN is found.)
  - Backtrack according to a priority based on closeness.
  - Reduce the boundary effects by randomization.

References:
[1] Selected Topics in Computer Vision at EPFL `COM-711`
[2] Prof. Shohreh Kasaei, *Advance vision course notes*, spring 2014.
Made by ma.mehralian using LaTeX