Pixel-level motion estimation (Optical flow)

- Optical flow is a velocity field in the image that transforms one image to the next image in the sequence
- We have to make some assumptions to constrain the space solutions
- ***Assumption 1: Brightness constancy***
    - *intensity of a point does not change during motion*
- ***Assumption 2: Small displacement***
    - *the displacement vector is sufficiently small*
- These two assumptions lead us to an optical flow constraint equation
    - ***Ix*u + Iy*v + It = 0***
- This equation does not constrain the solution space enough that's why we run into the aperture problem
    - *Component of the displacement which is parallel to the motion is unknown*
- There are two approaches to solving this problem
    - **Lucas - Kanade optical flow**
        - Accepts a third assumption
        - ***Assumption 3: Local motion coherency***
            - *neighbouring pixels (3x3) have equal (very similar) displacements*
            - *frames are sampled at discrete timestamps*
        - This assumption gives us more equations per pixel than we have unknowns which is why we use Least-squares solution by pseudo inverse

$$\begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = -\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

        -
        - Calculated from the spatial derivatives Ix and Iy that can be estimated with convolution
        - And the temporal derivative It which is the difference between frame1 and frame2
        - We can smooth temporal derivatives by some Gauss
        - Sometimes we can average the spatial derivatives over the two frames
        - Flow can be computed reliably when the equation system is implicitly solved
            - Eigen values large enough and of similar magnitude
            - Cannot be too small so that we can invert the matrix
            - One cannot be bigger than other, that means there is an edge (aperture problem)
        - Works well for small motions, large motions break Assumption 2
        - Pyramid representation
            - used to handle larger motion

        - takes the initial image and constructs a pyramid where transitions are blurred with Gauss and reduced by half
        - We start at the bottom and estimate the flow at every level
        - At the end we combine all the estimates
    - **Horn - Schunk optical flow**
        - Construct such a function Ec that reflects per-pixel flow quality such that if the flow agrees with the 2 constraints Ec is low and if it doesn't Ec is high
        - Consider it as an energy minimization problem for an entire image
        - The goal is to find such an optical flow field, that minimizes the mentioned energy function
        - We can solve this using Euler-Lagrange, then discretize it
        - $$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} (I_x^{\,2} + \alpha) & I_x I_y \\ I_x I_y & (I_y^{\,2} + \alpha) \end{bmatrix}^{-1} \begin{bmatrix} \alpha\bar{u} - I_x I_t \\ \alpha\bar{v} - I_y I_t \end{bmatrix}$$
        - u and v are on both sides of the equation that is why we apply iterations for solving this. Setting initial u and v to 0 and iterating until convergence
        - Perhaps initializing Horn - Schunk with Lucas - Kanade can speed up the process of iterating

Patch Tracking (Lucas - Kanade tracker)

- A high level view of tracking
    - Assume some model of the target
    - Assume estimated position in previous timestep
    - The goal is to align a template to an input image (Target localization)
    - Similarity measure to measure the quality of the alignment
        - Sum of squared differences
    - Greedy approach is to calculate SSD for all displacements then select the point where it is the smallest
    - We can do better with gradient descent where we minimize the SSD as the cost function
    - $$\mathrm{E}(\Delta p) = \sum_x \left(I(\mathrm{W}(\mathrm{x}; \mathrm{p} + \Delta p)) - \mathrm{T}(\mathrm{x}) \right)^2$$
    - Displacement models
        - Introduce a warp function W(x) that warps image onto template
            - Think about it as a transformation model W(x;p) which takes the coordinates x and warps them according to parameters p
        - Rigid body motion (rotation, translation)
        - Affine motion (rotation, translation, scale, shear)
- **Lucas - Kanade tracker**

- We try to minimize the above equation so that the parameters delta p give the smallest SSD
- We use linearization at p
- ***Algorithm steps:***
    1. ***Warp image I(x) with W(x;p)***
    2. ***Warp gradient image with W(x;p)***
    3. ***Evaluate the Jacobian at (x;p) and compute the steepest descent image***
    4. ***Compute the Hessian***
    5. ***Compute the increment for p***
    6. ***Update parameters p = p + delta p***
    7. ***Repeat until convergence or delta p sufficiently small***
- Stability of the tracker depends if the Hessian is invertible
- Corners are good features to track
- Can be used for motion compensation
- We can use LK optical flow to estimate sparse flow then fit parametric model by least squares or RANSAC

Patch Tracking (Mean Shift tracker)

- We have a lot of potential centers spread across the entire image
- Some are false similarities but the most potential centers is around where the real target is
- We begin at the previous position then we calculate the mean of the surrounding centers. The vector from starting position to the mean is called the ***Mean Shift vector***
- We can repeat this until the vectors a very close to 0
- Mathematically speaking this is just finding the modes of a Probability Density Function or PDF
- Data points are already a PDF but we usually smooth it with a kernel like Epanechnikov, Uniform, Normal. This is called a Kernel Density Estimation or KDE
- We can then perform gradient ascent on that KDE
- **Mean Shift tracker**
    - Using SSD as similarity measure is not good because the target can just rotate and the error will be huge.
    - We use color histograms that are invariant to rotations and such (not always a good idea)
    - Represent the target by choosing a feature space and then representing the image by a PDF in that feature space
    - Assign higher weights to pixels closer to the center
    - We have a histogram of the template and the region
    - We use the Bhattacharyya measure to compare the two histograms
    - Localization by histogram similarity
        - start at the previous position
        - search in the models neighborhood in next frame

- find best candidate by maximizing the histogram similarity
    - gradient ascent on the similarity function
        - We can use Mean Shift iterations
- If we used epanechnikov kernel for smoothing we can use its derivative in the Mean Shift which is the uniform kernel (simplifies things)

$$w_i = \sqrt{\frac{q_{b(x_i)}}{p_{b(x_i)}(x_0)}}$$

- $$x^{(k+1)} = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}$$

- ***Algorithm steps***
    - ***1. Initialize the target histogram and use smooth kernel (e.g. Epanechnik) = q***
    - ***2. Start at same location on the new frame and extract the histogram p at that location***
    - ***3. Calculate the weight (wi) for each pixel in the bounding box***
    - ***4. Calculate the new position (x k + 1)***
    - ***5. Reapeat until convergence***
- Can search for the target by focusing on the features that discriminate from the background
- We can battle scale change by running localization on 3 different sizes then reporting the best one

Discriminative tracking (Tracking by classifiers)

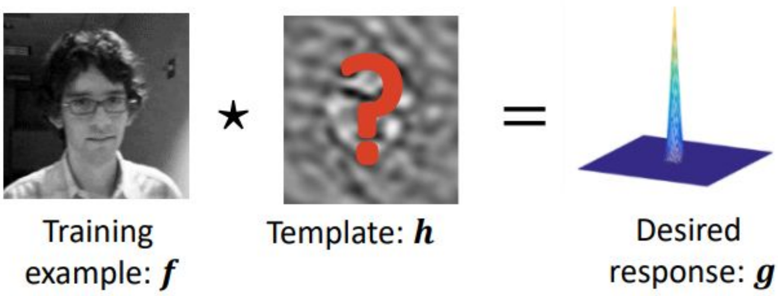
- Tracking as binary classification
    - A single supervised training example provided in the first frame
        - We take the target region as a positive example and background images as negative examples
    - This classifier might not be valid next frame so (self supervised) update is required
    - We can use combination of simple image features (Boosting as feature selection)
- Tracking by online adaboost
    - We have a starting position

- When the object moves we capture a certain region around the position on the previous frame and evaluate the subregions with the classifier
- This way we create a confidence map whose maximum is the position of the object in the new frame
- Update the classifier with positive and negative examples
    - This can fail because not all examples are good
    - We can put weight on examples proportional to the estimated overlap

- **Correlation-based tracking**
    - We can localize the target by using the maximum of the correlation response between the image and the template
    - Correlation is equivalent to point-wise multiplication in the Fourier domain

$$g = f \star h \quad \Leftrightarrow \quad \hat{g} = \hat{f} \odot \bar{\hat{h}}$$

    - After this we use the reverse Fourier transform to get the correlation
    - Correlation is circular in the discrete Fourier transform so we use Hanning window to reduce the boundary effect
    - This implements a linear classifier at all displacements, now all that we need is a good template, such that when correlated with a patch gives us a great response



    - We try to learn such a template h, that would minimize the difference between the correlation of the image with h and the desired response g
    - We arrive at a closed form solution

$$\bar{\hat{h}} = \frac{\hat{g} \odot \bar{\hat{f}}}{\hat{f} \odot \bar{\hat{f}} + \lambda}$$

Pixel-wise division!

    - ***Algorithm steps***
        - ***---------- LOCALIZATION STEPS---------------***
        - ***1. Extract search region (apply Hanning window)***
        - ***2. Compute FFT of the modified region***
        - ***3. Multiply the region and template and inverse FFT***
        - ***4. Take Maximum positions as centers***
        - ***---------UPDATE STEP (FILTER LEARNING)-----***

        - ***5. Extract region***
        - ***6. Compute FFT of the modified region***
        - ***7. Compute FFT of the desired response***
        - ***8. Compute new filter with the equation above***
        - ***9. Average the filter with the filter from previous steps***
    - Scale estimation can be done similarly as before, different scales and report the best matching one
    - Or we can do scale estimation with DCF
        - Take initial patch, resize it a couple of times then take pixel values along the scale-space
        - Learn the correlation filter over the 1d signal and repeat this over all the signals
        - Then when we are estimating we do the same initial steps only then we compute the correlation of the 1d signals with learned filters
        - average the responses and take the maximum scale
    - CNN correlation trackers where filters are learned with a CNN
    - Standard DCF tracker has problems because the filter is learned from cyclical shifts which produce unrealistic training examples

Recursive Bayes Filtering

- A principled way to address uncertainty in visual tracking
- A state estimation problem
    - given the location obtained by the detector and everything we know about the target what is the probability that the target is at state xt
    - *Key idea 1: Reason target states in terms of pdfs*
        - *detector uncertain (can consider each detection as a center of Gaussian)*
    - *Key idea 2: Recursively estimate the posterior*
        - *predict from uncertain motion model*
        - *measure from uncertain sensor*
        - *update distribution*
    - *Key ingredients:*
        - *Prior pdf: State definition*
        - *Predict: Dynamic model*
        - *Measure: Observation model*
        - *Inference: How do we combine the prior dynamics and measurements*
    - State definition
        - Target properties at a time step (encodes parameters)
    - Observation model
        - Transforms measurement into a probability
        - Choose a visual model (histogram, HOG, template)
        - Define similarity function with the visual model
        - Define a function that maps similarity to probability
    - Dynamic model
        - Predicts the target state from its previous estimate

    - We can assume that the velocity is nearly constant or that the acceleration is nearly constant
    - We mention models such as Random Walk, Nearly constant velocity, Nearly constant acceleration
  - Inference
    - Bring it all together, the goal is to rewrite the posterior at current time stamp as a function of the posterior of the previous time stamp

$$\underbrace{p(\mathbf{x}_k|\mathbf{y}_{1:k})}_{\text{posterior estimate}} \propto \underbrace{p(\mathbf{y}_k|\mathbf{x}_k)}_{\text{Observation model}} \int \underbrace{p(\mathbf{x}_k|\mathbf{x}_{k-1})}_{\text{motion model}} \underbrace{p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})}_{\text{posterior at k-1}} d\mathbf{x}_{k-1}$$

    -
- The Kalman Filter
  - We have a position from previous frame
  - We have a measurement from the observation model
  - We have a prediction from the motion model
  - We want to combine these into a new posterior
  - Given that all of these are Gaussian we can use the Kalman filter
  - Kalman filter takes the errors of the prediction and the measurement and calculates the Kalman Gain which tells us which of the two estimations we wil use more
  - Then it estimates the new position
- **Particle filter tracking**
  - Approximate the posterior by weighted samples
  - Instead of integration we sample and apply summation
  - ***Algorithm steps***
  - ***1. Draw N samples***
  - ***2. Move each sample by dynamic model***
  - ***3. Recursion steps***
    - ***4. start with posterior of the previous time step***
    - ***5. draw N new samples from the previous set***
    - ***6. apply the motion model to each particle (apply noise)***
    - ***7. obtain an observation for each particle***
    - ***8. evaluate likelihood of particle (hellinger distance)***
    - ***9. set the weights of the particle to the likelihood value***
  - ***10. update the reference histogram***

Deep learning trackers

- MDNet: Multi Domain Convolutional Neural Network Tracker
  - Target localization principle
    - Sample bounding boxes around t-1 location
    - Compute classification score
    - Take bounding box with maximum score
    - regres the BB parameters
  - Backbone pretraining

        - Pretrained on sequences with each sequence having its own fc6 component
        - in each selected frame sample 50 positive and 200 negative examples
    - Initialization on a new sequence
        - after pretraining the fc6 layer are removed and new fc6 is created
        - during tracking fine tune al fc layers
    - Online tracking
        - sample target positions, classify, output the max score
        - fine tune all fc layers
            - hard negative mining
- Recall tracking by correlation
    - pretrain the backbone of the network such that the correlation will yield a well-expressed maximum for arbitrary target
    - we push the patch through network and we push the template through the network and correlation between the results is what we are interested in
- SiamFc
    - Pretraining
        - take database, take random images different time intervals apart compute the correlation response then minimize it with respect to the ideal response
    - Tracking
        - template extracted in the first frame
        - target localization in the t-th frame
            - maximum of the correlation between the search region and the template (both encoded by CNN)
        - template not updated during tracking
    - Scale estimation
        - same as always, try different scale, report the best response
        - this gives a bad approx.
- Region proposal network
    - at each location test for k bounding box shapes
    - test the hypothesis that a certain bb is there
    - predicts a difference of that bounding box
- SiamRPN
    - added region proposal to Siam
- We can also use DCF as a part of the network
- ATOM: Accurate Tracking By Overlap Maximization
    - approximately localize by deep dcf
    - generate the proposal at dcf output
    - refine the proposal by net to predict bbox fit
    - update the deep dcf
- Discriminative tracking by segmentation (D3S)
    - single-shot segmentation network
    - two target appearance models
        - geometrically constrained euclidian model GEM
        - geometrically invariant model GIM

        - fusion for accurate segmentation
        - GEM
            - deep discriminative correlation filter formulation
            - localization: correlation response target center likelihood, per pixel target region likelihood
        - GIM
            - two sets of features extracted on the first frame
                - background features Xb
                - object features Xf
            - localization per pixel cosine similarity with Xb and Xf
        - Combine both outputs

Long term tracking

- Tracking by tracking, learning, detection (Predator)
    - the main component is the detector
    - run detector and short term tracker in parallel and use them to construct training samples for detector
    - The short term tracker
        - a cell grid of ~100 Lucas - Kanade trackers
        - each has a reliability estimate
    - The detector visual model
        - appearance model: greyscale patch
        - bounding box with fixed aspect
        - object model is collection of multiple positive and negative patches
    - The detector application
        - a scanning window
        - compare patches using a normalized cross correlation
        - a nearest neighbour using the NCC score
        - we would have to compare all pairs like this so we use a cascade approach
    - The detector cascade
        - stage 1: variance of patch: ignore regions with at least 50% smaller intensity variance
        - stage 2: ensamble of weak classifiers
    - The interaction algorithm
        - PN learning trains the detector
        - PN learning assumptions
            - two classes of labeling processes are available P and N
            - P proposes positive examples only
            - N proposes negative examples only
            - both noisy, can make mistakes but carefully combine and useful
        - P-event “Loop”
            - do not trust learning examples untill you are sure about their labels

        - N-event “Uniques”
            - object is unique in a single model
            - if tracked patch is in the frame all other detections are assumed wrong
- Tracking by oversampling local features (Alien)
    - require a multiview local appearance of the object
    - multiple local appearances should be combined with a global shape model
    - Represent them by key points
    - The idea is to detect the key points
    - align the target with the key points
    - store the key points along with their relative position to the target
    - …
- FuCoLot: Fully correlational Long-term Tracker (FCLT)
    - Discriminative correlation filter in two seperate components
    - short term tracking
    - detection
    - detector activated when tracker not confident
    - motion model used with detector

Performance evaluation

- Measure types
    - center error
        - root mean squared error
        - normalize the distance by size of target
    - overlap error
        - intersection over union
    - succes plot
        - a tracker is initialized and run until the end of the sequence
        - performance visualized as portion of frames with overlap over the overlap threshold
        - the measure is then AUC which was shown to be equivalent to average overlap
    - failure rate
        - counts the number of times the tracker had to be reset
- The VOT initiative
    - Evaluation system
        - a toolkit that automatically performs a battery of standard experiments
    - Dataset
        - keep it sufficiently small, diverse and well annotated
        - put together some existing and add other ones
        - then eliminate the bad ones
        - then asign 11 global attributes and split into groups according to those
        - sort by tracking difficutly with some basic trackers

- annotated with 6 different properties, occlusion, illumination change, object motion, camera motion, object size change, unassigned
- Evaluation methodology
    - selected robustness and accuracy amongst other measures by having low correlation with other measures
    - expected average overlap
        - combines acc and rob into single score
        - interpretation: the expected overlap a tracker obtains on a short term sequence of an everage length
    - issues with the reset protocol
        - anchor based protocol
            - avoid tracker specific reset points
            - introduce anchors
            - track in the direction of the largest number of tracking frames
        - failure redefinition
            - potential failure if overlap< 0.1
            - failure if the tracker does not recover in 10 frames
    - Long term tracking evaluation
        - precision
        - recall
        - f score = (2*Pr*Re)/(Pr + Re)
        - agreement = sufficient overlap
        - two thresholds -> sufficient overlap and detection threshold
        - primary measures are pre, re and f with such thresholds, that maximize f score of the detector
- Challenges and workshops
    - growing
    - all top trackers are deep trackers