W8 - Image classification

Object Recognition (spatial pyramid Matching)

BOW doesn't save any spatial information

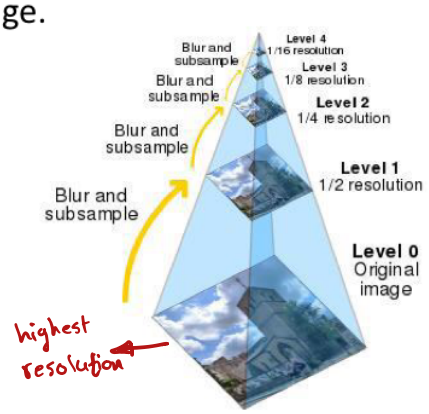
Solution: Spatial pyramid matching

Pyramids approach

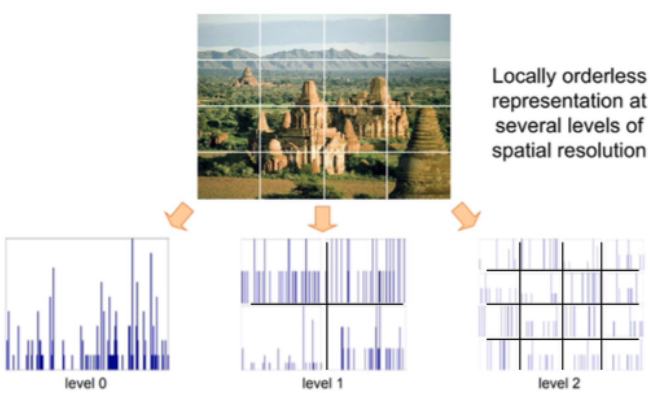
- Very useful for representing images.
- Pyramid is built by using multiple copies of image.
- Each level in the pyramid is 1/4 of the size of previous level
- The lowest level is of the highest resolution.
- The highest level is of the lowest resolution.

① split into 4 quarters

② decrese the resolution by $\frac{1}{2}$



- Spatial pyramid matching partitions the image into increasingly fine sub-regions and allows us to computes histograms (BoW) of local features inside each sub-region.



$$P(y|X) = \frac{P(X|y)\,P(y)}{P(X)}$$

Naive Bayes

- The Naive Bayes classifier assumes that the presence of a feature in a class is unrelated to any other feature. and contribute equally

Bayes' Theorem →

$$P(H|E) = \frac{P(E|H).P(H)}{P(E)}$$

Posterior Probability — hypothesis [class] y — evidences [features] X — Prior Probability

Likelihood

$$P(E|H) = \prod_{i=1}^{n} P(e_i|H)$$

$$P(E) = \prod_{i=1}^{n} P(e_i)$$

same for every class so we remove it

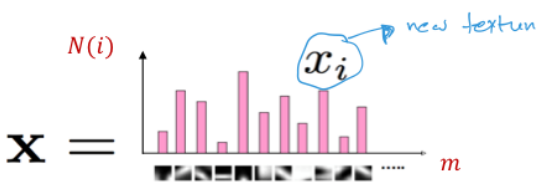
$$\Rightarrow P(H|E) \propto P(H) \prod_{i=1}^{n} P(e_i|H)$$

$$\Rightarrow H = \operatorname{argmax}_H P(H) \prod_{i=1}^{n} P(e_i|H)$$

do each class seperatily then take the max one

To classify image:

Classify image using histograms of occurrences on visual words:



- where:
  - x_i is the event of visual word v_i appearing in the image,
  - $N(i)$ the number of times word v_i occurs in the image,
  - m is the number of words in our vocabulary.

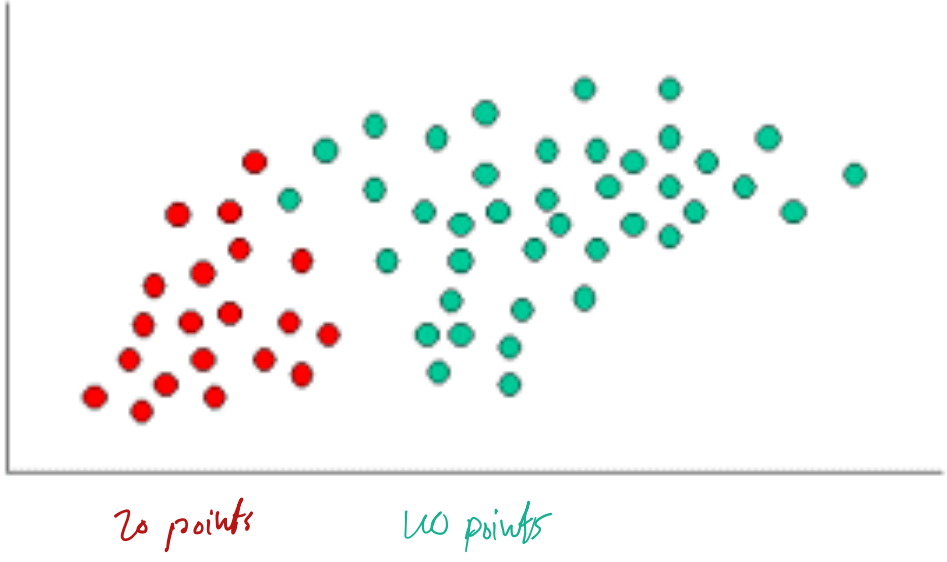
$$C = \operatorname{argmax}_c P(C) \prod_{i=1}^{m} P(x_i|C)^{N(i)}$$

Step 1: find the Prior Knowledge (P(c))

$$\sum P(c) = 1$$

Prior Red: $\frac{20}{60}$

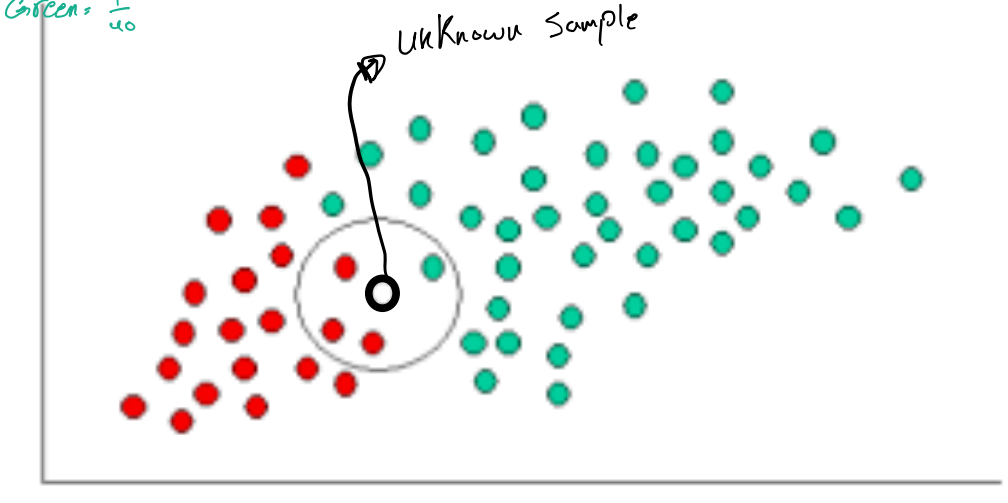
Prior Green: $\frac{40}{60}$



Step 2: find likelihood $P(x \mid c)$

Red = $\frac{3}{20} = \frac{1}{6}$

Green = $\frac{1}{40}$



Step 3: find Posterior probability (prior x Likelihood)

In the context of our example:

Posterior probability of X being GREEN ∝

Prior probability of GREEN × *Likelihood of X given GREEN*

$$= \frac{4}{6} \times \frac{1}{40} = \frac{1}{60}$$

Posterior probability of X being RED ∝

Prior probability of RED × *Likelihood of X given RED*

$$= \frac{2}{6} \times \frac{3}{20} = \frac{1}{20}$$

$$P(c \mid \boldsymbol{x}) = \frac{P(c)\, P(\boldsymbol{x} \mid c)}{\sum_{c'} P(c')\, P(\boldsymbol{x} \mid c')}$$

SVM → classifier, regression

for → Linear, non-Linear

- The goal of the line is to maximizing the margin between the points on either side of the so-called decision line.

Linear Classifier

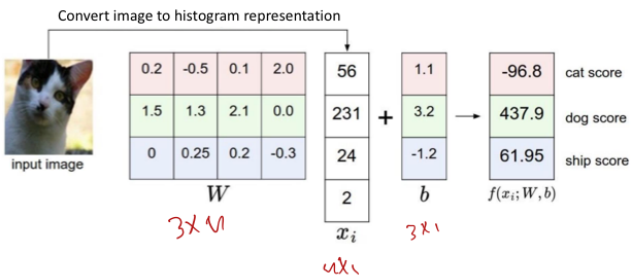
define a **score function**

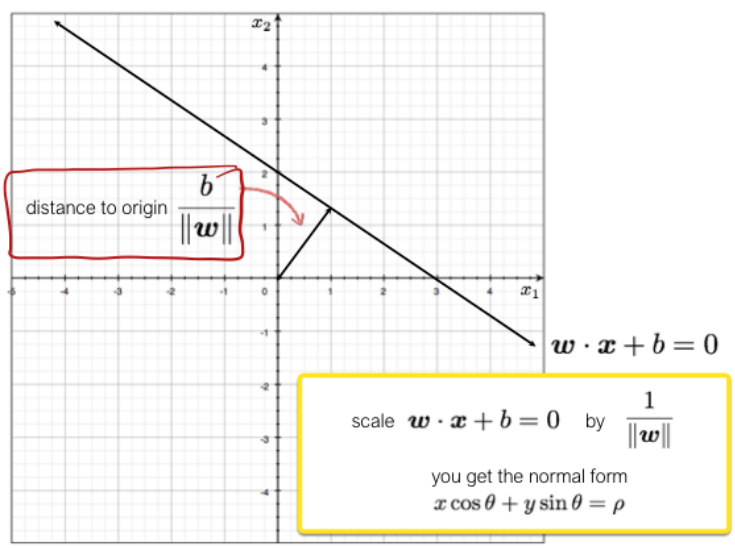
data (histogram)

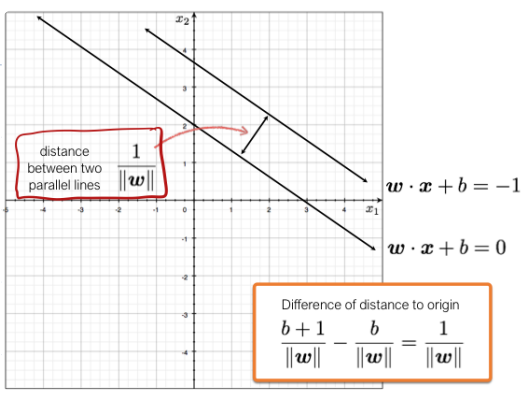
$$f(x_i, W, b) = Wx_i + b$$

class scores — "weights" — "bias vector" — "parameters"

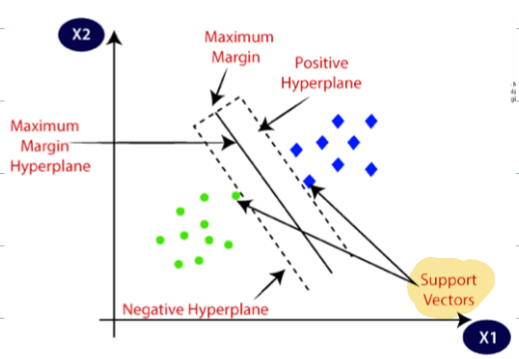
Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

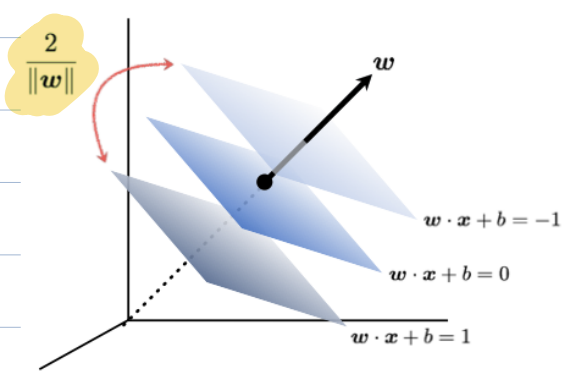




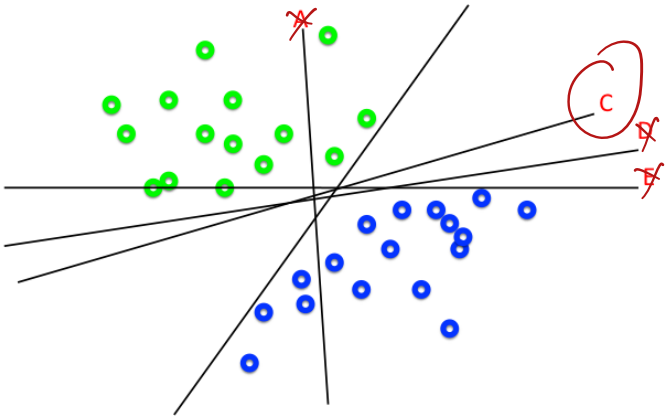


Hyperplanes (planes) in 3D





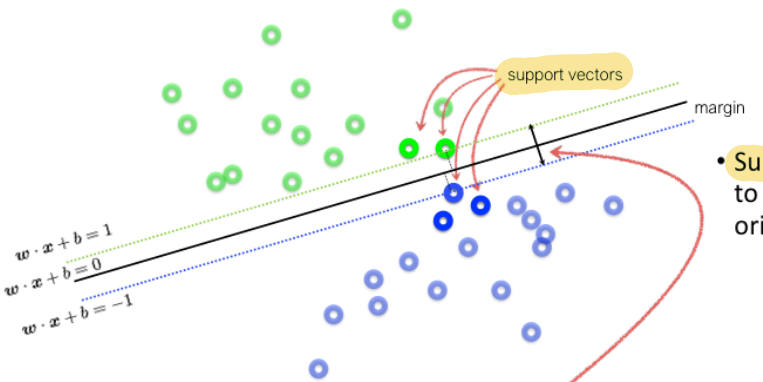
What's the best **w**?



Intuitively, the line that is the farthest from all interior points

Maximum Margin solution:
most stable to perturbations of data

What's the best **w**?

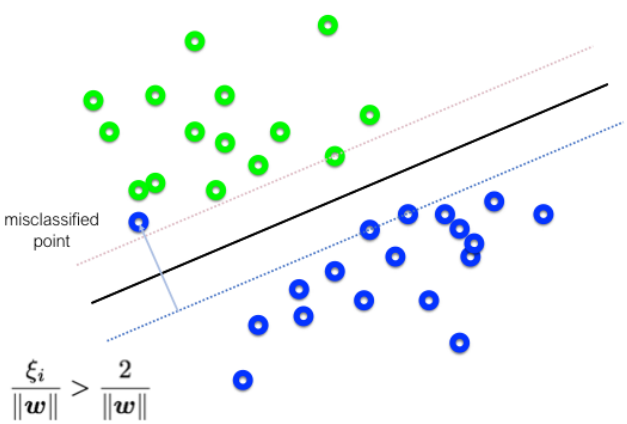


- Support vectors are the data points that are nearest to the hyper-plane and affect the position and orientation of the hyper-plane.

the gap between parallel hyperplanes $\frac{2}{\|w\|}$ is maximized

53

Adding slack variables $\xi_i \geq 0$



$\frac{\xi_i}{\|w\|} > \frac{2}{\|w\|}$

C = Infinity hard margin

C = 10 soft margin

Dimensionality Reduction

Principal Component Analysis (PCA)

what?

- Often used to reduce the dimensionality of large data sets
  - By transforming a large set of variables into a smaller one that still contains most of the information in the large set

Why?

- Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process.

General idea

- It comes at the expense of accuracy:
  - The trick in dimensionality reduction is to trade a little accuracy for simplicity.

detailed

Steps involve in PCA

1. Standardize the range of continuous initial variables.
2. Compute the covariance matrix to identify correlations.
3. Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components.
4. Create a feature vector to decide which principal components to keep.
5. Recast the data along the axes of the principal component.

normalize

$Z = \frac{x - \mu}{\sigma}$

is there any relationship?

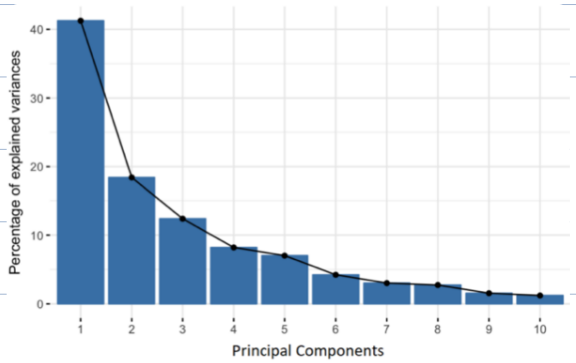
1 positive correlation

0 no --

inverse -1 negative

3 Principal components

- Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables.



- Principal components are less interpretable and don't have any real meaning since they are constructed as linear combinations of the initial variables.
- Geometrically speaking, principal components represent the directions of the data that explain a maximal amount of variance, that is to say, the lines that capture most information of the data.

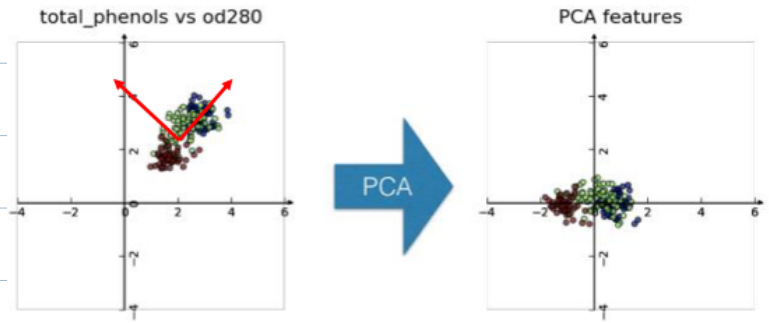
4. Create a feature vector to decide which principal components to keep.

- We choose whether to keep all these components or discard those of lesser significance (of low eigenvalues)
- And then form with the remaining ones a matrix of vectors that we call Feature vector.

The feature vector is simply a matrix that has as columns the eigenvectors of the components that we decide to keep.

5

$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

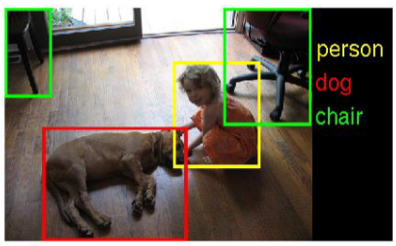


W 9 - Detecting Object by Parts

Object detection Task

- **Problem**: Detecting and localizing generic objects from various categories, such as cars, people, etc.

- Challenges:
  - Illumination,
  - viewpoint,
  - deformations,
  - Intra-class variability



Evaluation

Using → Benchmarks are used to make sure we are moving forward and performing better with new research.

- Examples of object detection benchmarks:
  - PASCAL VOC Challenge
  - ImageNet
  - COCO
  - MNIST
  - ...

PASCAL VOS Challenge

20 categories
had high variability within each category

ImageNet (ILSVR)

200 categories
more variability
many objects in a single image

Common Objects in Context [COCO]

80 Categories
to best detection and Segmentation
very time-consuming

How to evaluate object detection??

By compairing with ground truth

How do we evaluate object detection?

- True positives are objects that both the algorithm (prediction) and annotator (ground truth) locate.

predictions

ground truth

True positive:

- The overlap of the prediction with the ground truth is MORE than 0.5

How do we evaluate object detection?

2

False positives are objects that the algorithm (prediction) locates but the annotator (ground truth) does not locate. False positives are also referred to as false alarms.

predictions

ground truth

True positive:

False positive:

- The overlap of the prediction with the ground truth is LESS than 0.5

How do we evaluate object detection?

- False negatives are ground truth objects that our model does not find. These can also be referred to as misses.

predictions

ground truth

True positive:

False positive:

False negative:

- The objects that our model doesn't find

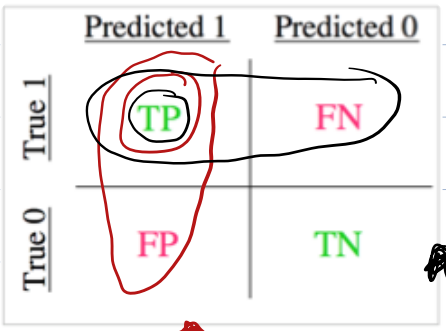
How do we evaluate object detection?

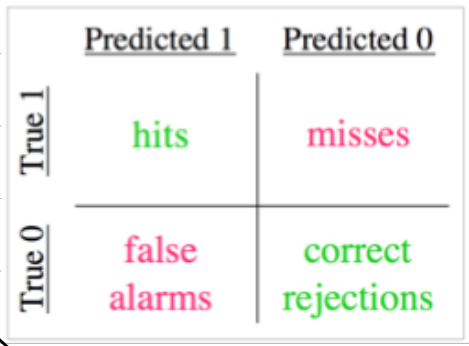
- False negatives are ground truth objects that our model does not find. These can also be referred to as misses.

predictions

ground truth

True positive:
False positive:
False negative:
True negative:
- are anywhere our algorithm didn't produce a box and the annotator did not provide a box.





$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

✓ Precision can be thought of as the fraction of correct object predictions among all objects detected by the model.

✓ Recall can be thought of as the fraction of ground truth objects that are correctly detected by the model.

In object detection

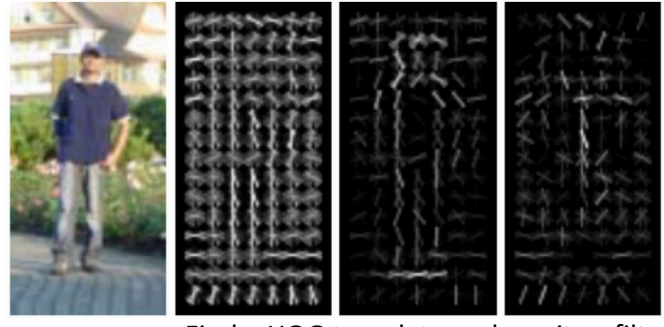
- Precision:
  - how many of the object detections are correct?
- Recall:
  - how many of the ground truth objects can the model detect?

A simple detector

Dalal-Triggs Detector

Histogram of Oriented Gradient

HOG features



Find a HOG template and use it as filter

- An image window is divided into blocks;
- The magnitude of the gradients of the pixels in each block are accumulated into bins according to the direction of the gradients.
- These local histograms of the blocks are then normalized and concatenated to produce a feature representation of the image window.

approatch 1: Sliding window + HOG features

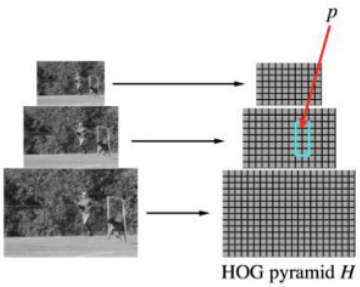
How

- Slide through the image and check if there is an object at every location

Problem: When the object is larger than the window

- We need to do multi scale sliding window

Using feature Pyramid



slide it on the image

Object Filter/Template:

- HOG features.
- Global for the entire object: no explicit information about the "parts" that make up the object
- Rigid: no explicit handling of object deformation/change of pose.

Problems

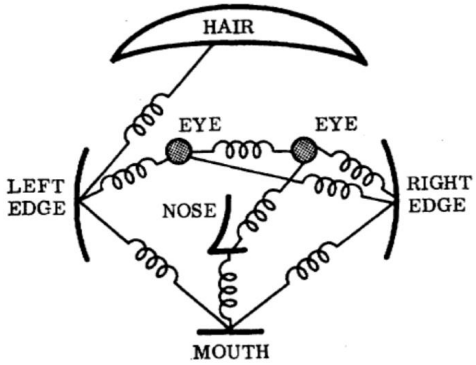
① ②

Deformation parts model

Deformable Parts Model (DPM)

from BOW

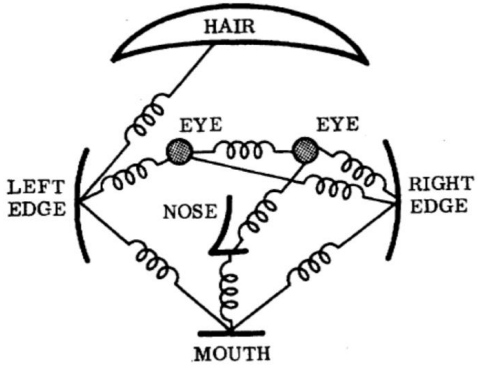
- Represents an object as a collection of parts arranged in a deformable configuration
- Each part represents local appearances
- Spring-like connections between certain pairs of parts



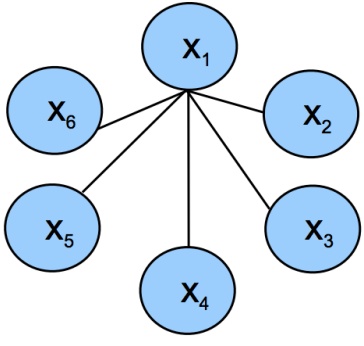
Fischler and Elschlager, Pictoral Structures,1973

Deformable Parts Model (DPM)

- The parts of an object form pairwise relationships.
- We can model this using a “star model”
  - where every part is defined relative to a root.

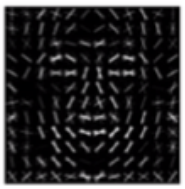


- Each model will have:
  - Global filter (root filter)
  - A set of part filters



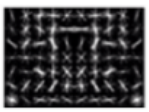
Person

Global/root filter

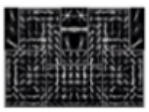


"side view" bike model component

Root filter



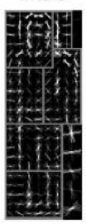
Part filters



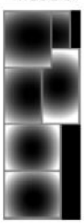
Six-component person model

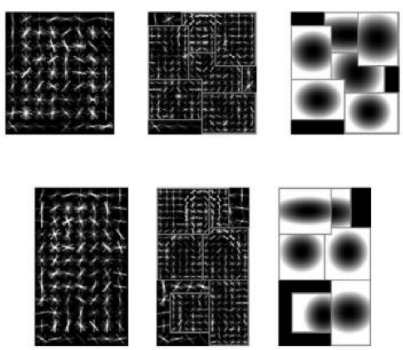
1 Root filter

2 Part filters



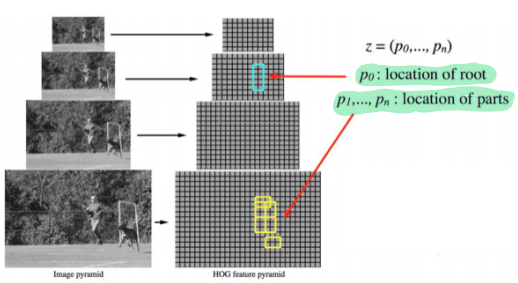
3 Deformations models





Deformable parts calculates a score for each part along with a global score

$p_i = (x_i, y_i, l_i)$ specifies the level and position of the i-th filter



Calculating the score for a detection

The score for a detection is defined as the sum of scores for the global and part detectors minus the sum of deformation costs for each part.

$$\text{detection score} = \sum_{i=0}^{n} F_i \, \phi(p_i, H) - \sum_{i=1}^{n} d_i\left(\Delta x_i, \Delta y_i, \Delta x_i^2, \Delta y_i^2\right)$$

deformation

global and parts

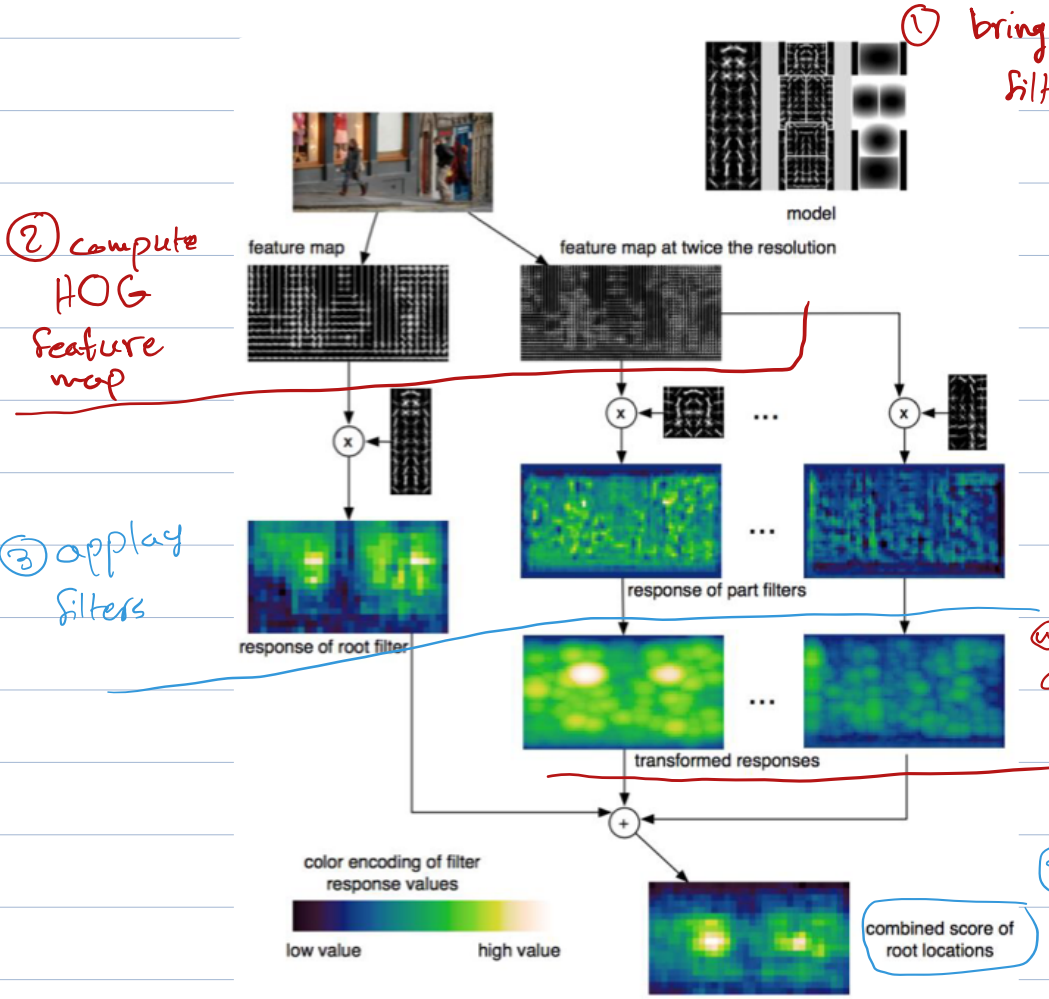
This means that if a detection's parts are really far away from where they should be, it's probably a false positive.

- Scores for each part filter + global filter (similar to Dalal and Triggs).
- The deformation costs for each part.
  - Δx_i measures the distance in the x-direction from where part i should be.
  - Δy_i measures the same in the y-axis direction.
  - d_i is the weight associated for part i that penalizes the part for being away.

Detection Pipeline

to make detection: sliding window with global and part filters

to score a detection: Σ global and part scores $-$ deformation of the parts

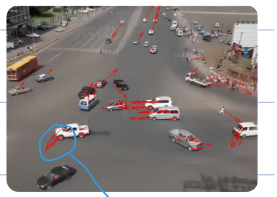


DPM - discussion

- Advantages
  - Parts have intuitive meaning.
  - Standard detection approaches can be used for each part.
  - Works well for specific categories.
- Disadvantages
  - Parts need to be selected manually
  - Semantically motivated parts sometimes don’t have a simple appearance distribution
  - No guarantee that some important part hasn’t been missed
- When switching to another category, the model has to be rebuilt from scratch.

W9 - P2 - Motion and Tracking

Optical Flow



Simply, optical flow is the movement of pixels over time.

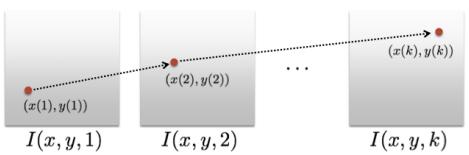
- Problem Definition
  - Given two consecutive image frames, estimate the motion of each pixel, to generate motion vector

- Assumptions
  1. Brightness constancy ⇒ • Allows for a pixel-to-pixel comparison (not image features)
  2. Small motion ⇒ • Linearization of the brightness constancy constraint

- By computing a motion vector field between each successive frame in a video, we can track the flow of objects, or, more accurately, "brightness patterns" over extended periods of time.

Assumption #1: Brightness constancy

Scene point moving through image sequence

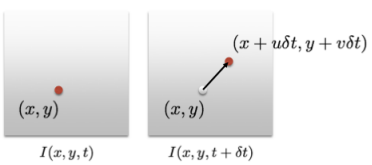


Assumption:Brightness of the point will remain the same

$$I(x(t), y(t), t) = C$$

constant

Assumption #2: Small motion



Optical flow (velocities): (u, v) Displacement: $(\delta x, \delta y) = (u\delta t, v\delta t)$

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t)$$

For small space-time step, brightness of a point is the same

These assumptions yield the …

Brightness Constancy Equation

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

change in x · change in y · in time

total derivative · partial derivative

Brightness Constancy Equation

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

are same

shorthand notation

$$I_x u + I_y v + I_t = 0$$

(x-flow) (y-flow)

change in x direction

to find the I value in the future

vector form

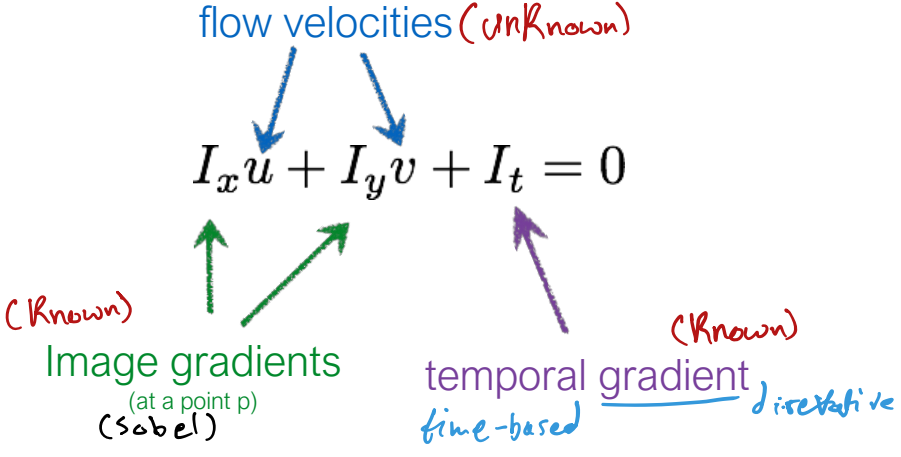
time

$$\nabla I^\top \boldsymbol{v} + I_t = 0$$

(1 x 2) (2 x 1)

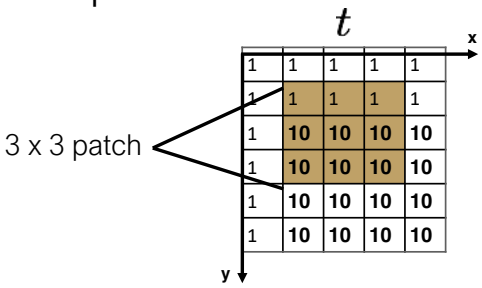
(putting the math aside for a second…)

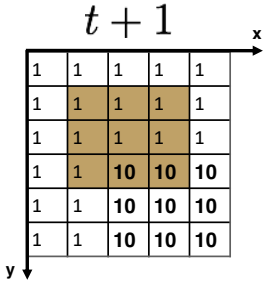
What do the terms of the brightness constancy equation represent?



How do you compute these terms?

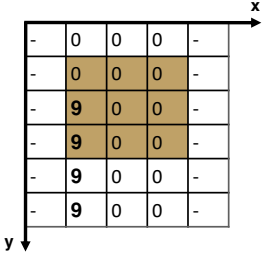
Example:





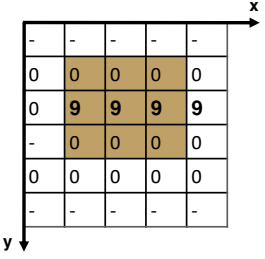
$t - (t+1)$

$$I_x = \frac{\partial I}{\partial x}$$



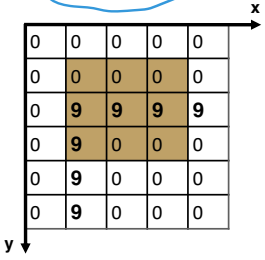
-1 0 1

$$I_y = \frac{\partial I}{\partial y}$$



-1
0
1

$$I_t = \frac{\partial I}{\partial t}$$



IN W10-P1

two approaches to find U, V

assume:
1. small motion
2. color constancy
3. spatial coherence

Horn-Schunck Optical Flow (1981)

use two assumptions:

1. brightness constancy
2. small motion

‘smooth’ flow
(flow can vary from pixel to pixel)

global method
(dense)

Lucas-Kanade Optical Flow (1981)

method of differences

‘constant’ flow
(flow is constant for all pixels)

local method
(sparse)

Assume that the surrounding patch (say 5x5) has 'constant flow'

Assumptions:

1. Flow is locally smooth
2. Neighboring pixels have same displacement

act as coherent object

pixels

Using a 5 x 5 image patch, gives us 25 equations

$$I_x(\boldsymbol{p}_1)u + I_y(\boldsymbol{p}_1)v = -I_t(\boldsymbol{p}_1)$$
$$I_x(\boldsymbol{p}_2)u + I_y(\boldsymbol{p}_2)v = -I_t(\boldsymbol{p}_2)$$
$$\vdots$$
$$I_x(\boldsymbol{p}_{25})u + I_y(\boldsymbol{p}_{25})v = -I_t(\boldsymbol{p}_{25})$$

Equivalent to solving:

$$A^\top A \qquad \hat{x} \qquad A^\top b$$

$$\begin{bmatrix} \sum_{p\in P} I_x I_x & \sum_{p\in P} I_x I_y \\ \sum_{p\in P} I_y I_x & \sum_{p\in P} I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum_{p\in P} I_x I_t \\ \sum_{p\in P} I_y I_t \end{bmatrix}$$

where the summation is over each pixel $\boldsymbol{p}$ in patch $\boldsymbol{P}$

Sometimes called 'Lucas-Kanade Optical Flow'
(can be interpreted to be a special case of the LK method with a translational warp model)

$$x = (A^\top A)^{-1} A^\top b$$

When is this solvable?

$$A^\top A\hat{x} = A^\top b$$

$A^\top A$ should be invertible

$A^\top A$ should not be too small

λ_1 and λ_2 should not be too small

$A^\top A$ should be well conditioned

λ_1/λ_2 should not be too large (λ_1=larger eigenvalue)

harris corner detector

$$A^\top A = \begin{bmatrix} \sum_{p\in P} I_x I_x & \sum_{p\in P} I_x I_y \\ \sum_{p\in P} I_y I_x & \sum_{p\in P} I_y I_y \end{bmatrix}$$

Implications

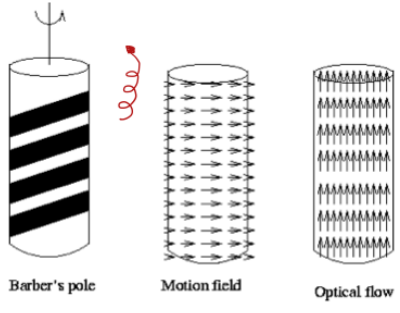
- Corners are when λ1, λ2 are big; this is also when Lucas-Kanade optical flow works best
- Corners are regions with two different directions of gradient (at least)
- Corners are good places to compute flow!

What happens when you have no 'corners'?

→ Barber's pole illusion

→ Aperture problem

Can't detect the direction of change [if there is no corner]



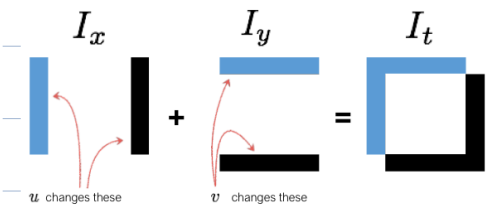
W10 - P1 - Motion and Tracking

Horn-Schunck optical flow

[1] Enforce brightness constancy

$$I_x u + I_y v + I_t = 0$$

Find the optical flow such that it satisfies:



For every pixel,

$$\min_{u,v} \left[I_x u_{ij} + I_y v_{ij} + I_t \right]^2$$

lazy notation for $I_x(i,j)$

[2] Enforce Smooth flow field

$$\min_{\boldsymbol{u}} (u_{i,j} - u_{i+1,j})^2$$

Combine them

$$\min_{\boldsymbol{u},\boldsymbol{v}} \sum_{i,j} \left\{ E_s(i,j) + \lambda E_d(i,j) \right\}$$

smoothness; brightness constancy; weight

Brightness constancy $E_d(i,j) = \left[I_x u_{ij} + I_y v_{ij} + I_t \right]^2$

Smoothness

$$E_s(i,j) = \frac{1}{4}\left[(u_{ij} - u_{i+1,j})^2 + (u_{ij} - u_{i,j+1})^2 + (v_{ij} - v_{i+1,j})^2 + (v_{ij} - v_{i,j+1})^2 \right]$$

Horn-Schunck Optical Flow Algorithm

1. Precompute image gradients $I_y \quad I_x$
2. Precompute temporal gradients I_t
3. Initialize flow field $\boldsymbol{u} = \mathbf{0}$, $\boldsymbol{v} = \mathbf{0}$
4. While not converged

   Compute flow field updates for each pixel:

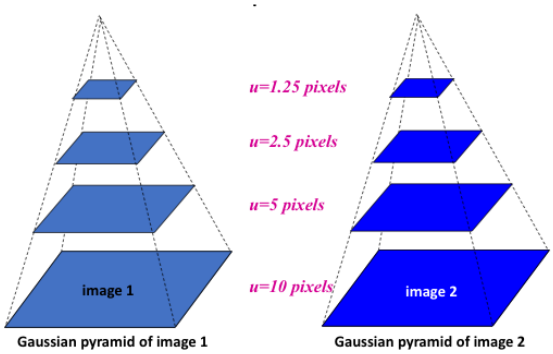
old average

goes to 0 when λ is small or $I_x^2 + I_y^2$ is big

$$\hat{u}_{kl} = \bar{u}_{kl} - \frac{I_x \bar{u}_{kl} + I_y \bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_x \quad \hat{v}_{kl} = \bar{v}_{kl} - \frac{I_x \bar{u}_{kl} + I_y \bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2} I_y$$

* In optical flow we assume that the motion is small
But, in real life the motion is much larger than one pixel

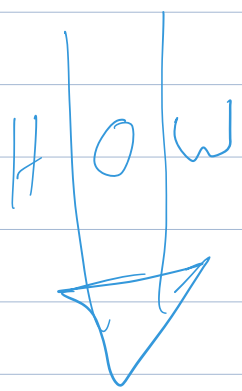
Solution: Reduce the resolution (using pyramids)



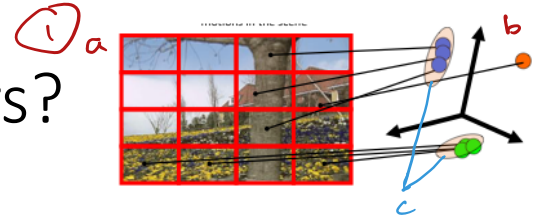
* Last assumption was spatial coherence and we can achive that by motion segmentation

using Gestalt - Common fate

Break image sequence into "layers" each of which has a coherent (affine) motion

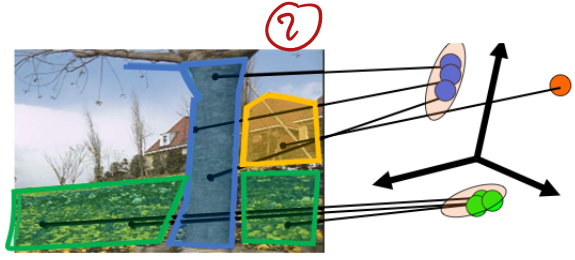


How do we estimate the layers?



1. Obtain a set of initial affine motion hypotheses
   - a • Divide the image into blocks and estimate affine motion parameters in each block by least squares
     - Eliminate hypotheses with high residual error
   - b • Map into motion parameter space
   - c • Perform k-means clustering on affine motion parameters
     - Merge clusters that are close and retain the largest clusters to obtain a smaller set of hypotheses to describe all the motions in the scene

2. Iterate until convergence:
   - Assign each pixel to best hypothesis
     - Pixels with high residual error remain unassigned
   - Perform region filtering to enforce spatial constraints
   - Re-estimate affine motions in each region



Source: Silvio Savarese

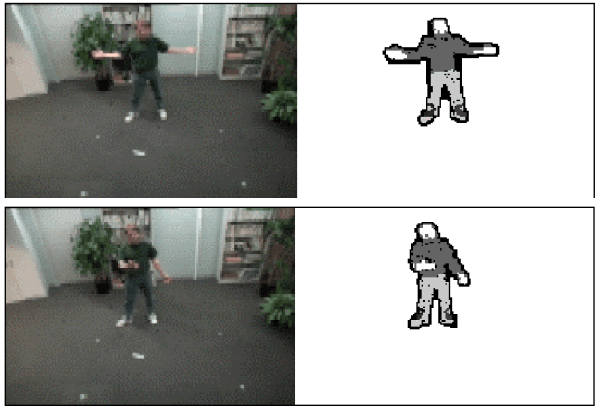
Where we can use motion

Uses of motion

- Tracking features
- Segmenting objects based on motion cues
- Learning dynamical models
- Improving video quality
  - Motion stabilization
  - Super resolution
- Tracking objects
- Recognizing events and activities

Segmenting objects based on motion cues

- Background subtraction
  - A static camera is observing a scene
  - Goal: separate the static background from the moving foreground



using clustering

morphological

Source: Silvio Savarese

W10 - P2 - Motion and Tracking

we finished motion let's see Tracking

- The objective of tracking is to associate target objects and estimate target state over time in consecutive video frames.

Challenges of tracking

- Tracking can be a time consuming process due to the amount of data that in video.
- Tracking relies on object recognition algorithms for tracking, which might become more challenging and prone (ميال) to failure for the following reasons:
  - Variations due to geometric changes like the scale of the tracked object
  - Changes due to illumination and other photometric aspects
  - Occlusions in the image frame
  - Motion that is non-linear
  - Blurry videos or videos with bad resolution might make the recognition fail
  - Similar objects in the scene

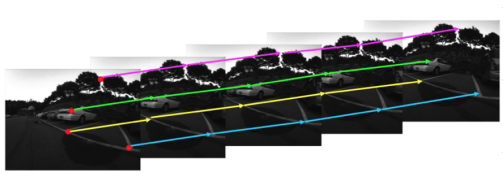
How?

Motion estimation techniques

- **Optical flow** ← what we have seen earlier
  - Recover image motion at each pixel from spatio-temporal image brightness variations (optical flow)
- **Feature-tracking**
  - Extract visual features (corners, textured areas) and "track" them over multiple frames

Feature point detection

Feature point tracking



Challenges in Feature tracking

1. Figure out which features can be tracked
   - Efficiently track across frames
2. Some points may change appearance over time
   - e.g., due to rotation, moving into shadows, etc.
3. Drift: small errors can accumulate as appearance model is updated
4. Points may appear or disappear.
   - need to be able to add/delete tracked points.

which features??

- avoid smooth regions and edges
- → use key points that produced from Harris corners

then we can use one of the Optical flow algorithms

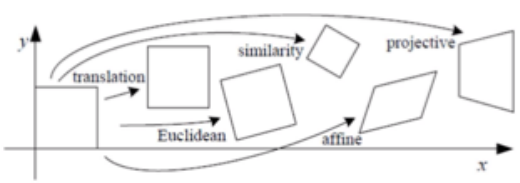
Simple Kanade–Lucas–Tomasi (KLT) feature tracker

1. Find a good point to track (Harris corner)
2. For each Harris corner compute motion (translation or affine) between consecutive frames.
3. Link motion vectors in successive frames to get a track for each Harris point
   - If the patch around the new point differs sufficiently from the old point, we discard these points.
4. Introduce new Harris points by applying Harris detector at every m (10 or 15) frames
5. Track new and old Harris points using steps 1-3

2D transformations:

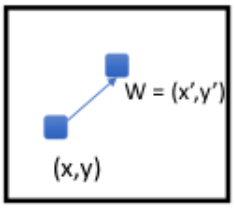
viewpoint movement

Depending on camera and objects, choose the right transformations



- Fixed overhead cameras will see only **translation** transformations.
- Fixed cameras of a basketball game will see **similarity** transformations.
- People in pedestrian (مشرجلين) detections can see **affine** transformations.
- And moving cameras can see **projective** (اسقاط) transformations.

① Translation (Shifting)



$$\begin{aligned} x' &= x + b_1 \\ y' &= y + b_2 \end{aligned} \iff \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$W(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \boldsymbol{p} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

② Similarity (① Scaling + ② translation)

$$\begin{aligned} x' &= ax + b_1 \\ y' &= ay + b_2 \end{aligned}$$

- $W(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} a & 0 & b_1 \\ 0 & a & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

$$\boldsymbol{p} = [a \quad b_1 \quad b_2]^T$$

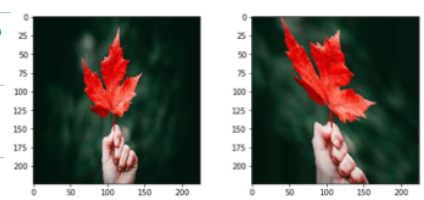
③ Affine (① Scaling + ② translation + ③ rotation + ④ Shear)

$$\begin{aligned} x' &= a_1x + a_2y + b_1 \\ y' &= a_3x + a_4y + b_2 \end{aligned}$$

- $W(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

$$\boldsymbol{p} = [a_1 \quad a_2 \quad b_1 \quad a_3 \quad a_4 \quad b_2]^T$$

shear



* Iterative KLT tracker

- This approach differs from the simple KLT tracker by the way it links frames:
  - instead of using optical-flow to link motion vectors and track motion, we directly solve for the relevant transforms using feature data and linear approximations.
- This allows us to deal with more complex (such as affine and projective) transforms and link objects more robustly.

why

Iterative KLT steps

1. First, use Harris corner detection to find the features to be tracked.
2. For each feature at location $x = [x, y]^T$: Choose a feature descriptor and use it to create an initial template for that feature (likely using nearby pixels): $T(x)$.
3. Solve for the transform p that minimizes the error of the feature description around $x_2 = W(x; p)$ (your hypothesis for where the feature's new location is) in the next frame. In other words, solve the equation

$$\sum_x [I(W(\boldsymbol{x}; \boldsymbol{p})) - T(x)]^2$$

4. Iteratively reapply this to link frames together, storing the coordinates of the features as the transforms are continuously applied. This should give you a measure of how objects move through frames.
5. Just as before, every 10-15 frames introduce new Harris corners to account for occlusion and "lost" features.

points to consider:

- Window size
  - Small window more sensitive to noise and may miss larger motions (without pyramid)
  - Large window more likely to cross an occlusion boundary (and it's slower)
  - 15x15 to 31x31 seems typical
- Weighting the window
  - Common to apply weights so that center matters more (e.g., with Gaussian)

W11 - P1 - Depth Estimation

- autonomous
- 3D Scene reconstruction
- Augmented reality

- Depth is essential for 3D vision.

means

Measuring distance relative to a camera

* 3D Map construction:

- Computing depth allows us to back project images captured from multiple views into 3D.

Challenges:

- Correspondence matching: which can be difficult due to reasons such as ① texture (اللمس), ② occlusion, ③ non-lambertian surfaces (غير ثابت مع), resolving ambiguous solution, where many 3D scenes can actually give the same picture on the image plane i.e., predicted depth is not unique.

occlusion: when 2 or more objects come too close and seemingly merged or combine with each other

* Alternative to measure depth:

Sensor such as Lidar

Challenges:

① occlusion

② dynamic object in the scene

③ imperfect stereo correspondence

For example: Cars' windshield (الزجاج الأمامي) often degrades matching and hence estimation.

Solutions: Sensor fusion (combine data from more than one source)

Approaches for Depth estimation using CV:

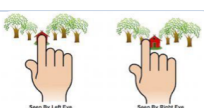
1. Depth from monocular (أحادي) images
2. Depth from Stereo images

How humans measure the depth?

- The mechanism at work here is our brain starts to reason about the incoming visual signals by recognizing patterns such as the size, texture and motion about the scene known as Depth Cues.

Judging Depth Using Cues

- There are basically 4 categories of depth cues:
  - Static monocular → depends on the spatial arrangements of things in the scene
  - Depth from motion → When you, as an observer, is in motion, things around you pass by faster than the one that is farther away.
  - Binocular cues → Retina Disparity (تفاوت الشبكية)
  - Physiological cues



- The greater the disparity, the closer things are to you.