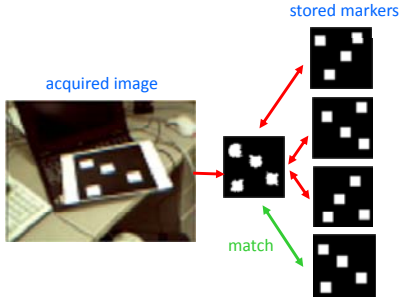
Markerless Augmented Reality

Concepts

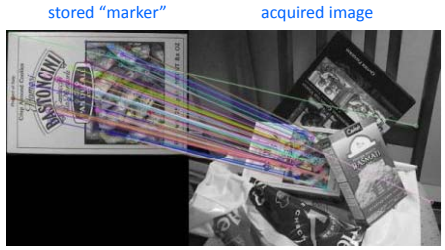
- A marker is a feature or artefact in the real world that is used to anchor an augmentation.
- This includes
  - markers such as barcodes, 2D matrix codes and other machine readable patterns,
  - artefacts such as posters,book covers or pictures, and
  - natural features such as buildings and landmarks.
- A distinction (marker vs. markerless) is frequently made between objects that have been created specifically for the purpose of an augmentation (markers, barcodes), and objects that are part of the everyday environment (posters, book covers, etc.).
- In computer vision, the term feature is used to describe a pattern that image recognition algorithms can use to identify an object.
  - feature extraction algorithms are fundamental for image recognition approaches to registration and tracking.

Marker *vs* Markerless AR

- Fiducial object recognition



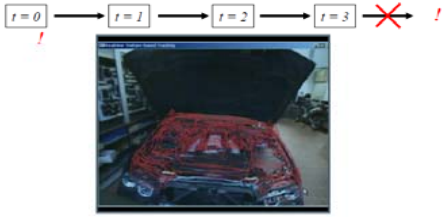
Marker-based AR



Markerless-based AR

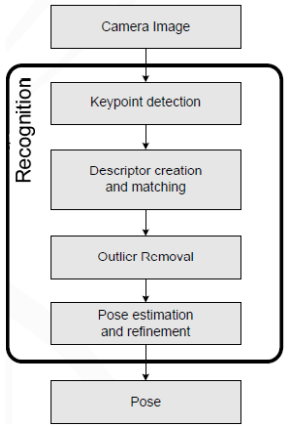
Detection and Tracking

- Detection:
  aims to retrieve the object pose without prior on the pose
- Tracking:
  depends on the pose estimated for time *t-1* to estimate the pose at time *t*.
  - Requires an initialization.
  - Fails when the object or the camera move too fast, or when the target object is completely occluded.

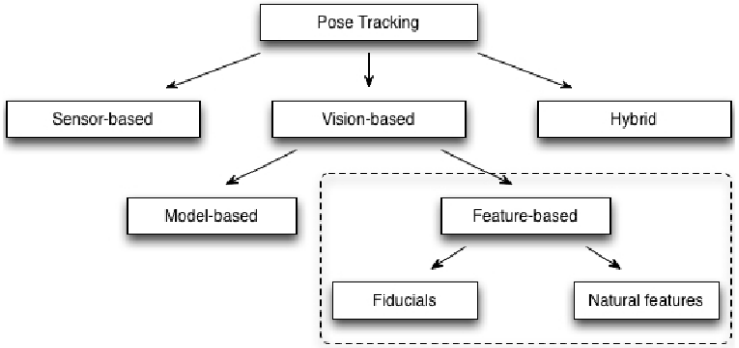


Tracking by detection

- This is what most trackers do… (?)
- Targets are detected every frame
- Popular because tracking and detection are solved simultaneously



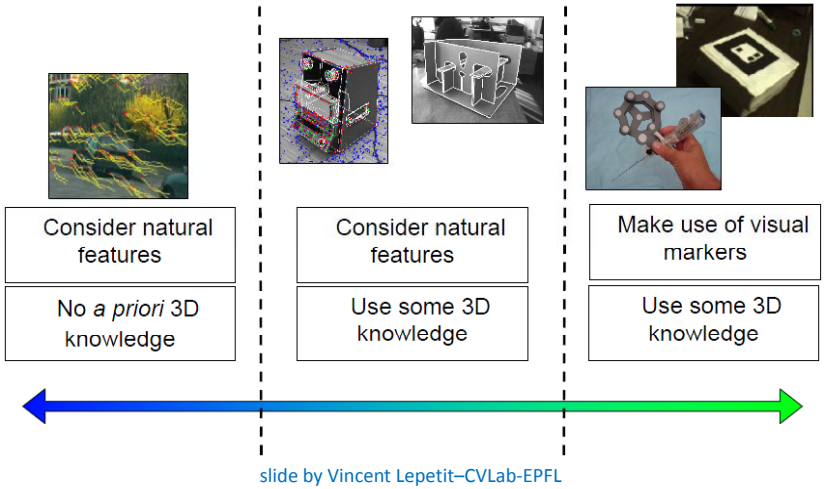
Pose tracking techniques



Different approaches to sensor-based tracking

- Location
  - GPS
  - wi-fi positioning
  - cell tower triangulation
- Orientation
  - compass
- Relative movement/rotation
  - accelerometer
  - gyroscope
- Context
  - light sensor
  - audio
  - proximity

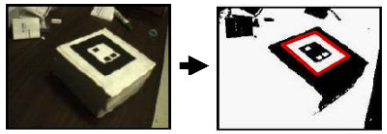
Different approaches to vision-based 3D tracking



slide by Vincent Lepetit–CVLab-EPFL

Using planar markers

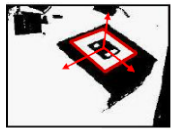
- Analyzed in previous lectures
- Marker(s) detected in every image;
  camera pose inferred from the appearance of the marker



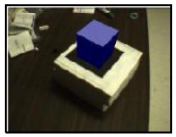
Input image

Binarized image
Straight line detection
Corner detection

Estimation of the homography from the corners (actually another method is used in ARToolkit)



Pose estimation



Augmentation

adapted from slide by Vincent Lepetit–CVLab-EPFL

Natural Feature Tracking - Overview

Feature-based recognition

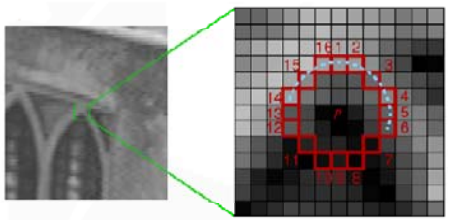
- Feature detection
- Feature description
- Feature matching

Natural feature tracking (NFT)

- Tracking from
  features of the surrounding environment
  - Corners, edges, blobs, ...
- Generally more difficult than marker tracking
  - Markers are designed for their purpose
  - The natural environment is not…
- Less well-established methods
  - Every year new ideas are proposed
- Usually slower than marker tracking

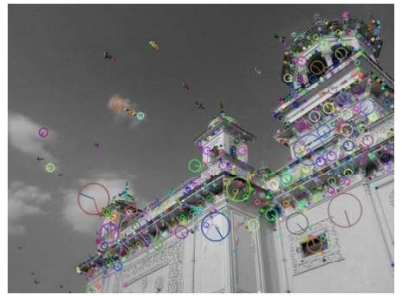
NFT- What is a keypoint / feature point?

- It depends on the detector you use!
- For high performance, use the FAST corner detector *(see later)*
- Apply FAST to all pixels of your image
- Obtain a set of keypoints for your image
  - Reduce the amount of corners using non-maximum suppression
- Describe the keypoints



NFT - What is a descriptor?

- Again, depends on your choice of a descriptor!
- Can use SIFT *(see later)*
  - Estimate the dominant keypoint orientation using gradients
  - Compensate for detected orientation
  - Describe the keypoints in terms of the gradients surrounding it
    - *Why gradients instead of intensities …?*

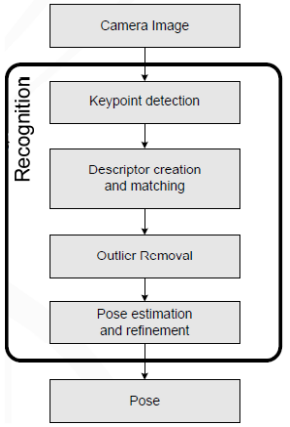


NFT – Database creation

- Offline step
- Searching for corners in a static image
- For robustness look at corners on multiple scales
  - Some corners are more descriptive at larger or smaller scales
  - We don't know how far users will be, during image acquisition
- Build a database file with all descriptors and their position on the original image.

NFT – Real-time tracking

- Search for keypoints in the video image.
- Create the descriptors.
- Match the descriptors from the live video against those in the database.
- Remove the keypoints that are outliers.
- Use the remaining keypoints to calculate the camera pose.



NFT – Outlier removal

- Some matches may be wrongly established ...
- Cascade of removal techniques
- Start with cheapest, finish with most expensive…
  - First , simple geometric tests
    - Ex: line tests
      - Select 2 points to form a line
      - Check all other points being on correct side of line
  - Then, homography-based tests

NFT – Pose refinement

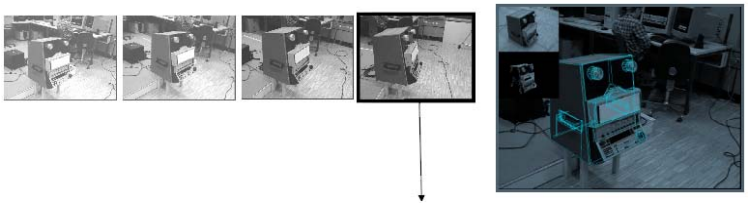
- Pose from homography makes good starting point
- Based on Gauss-Newton iteration (or other methods), try to minimize the reprojection error of the keypoints (*see previous lectures*)
- Typically 2-4 iterations are enough…

Marker vs. Natural feature tracking

- Marker tracking
  - (+) Usually requires no database of keypoints to be stored
    - marker database is anyhow needed
  - (+) Markers can be an eye-catcher
  - (+) Tracking is less demanding
  - (-) The environment must be instrumented with markers
  - (-) Markers usually work only when fully in view
- Natural feature tracking
  - (-) A database of keypoints must be stored/downloaded
  - (-) Natural feature targets might catch the attention less
  - (+) Natural feature targets are potentially everywhere
  - (+) Natural feature targets work also if partially in view

Model-based Tracking in Augmented Reality

What is model-based 3D tracking

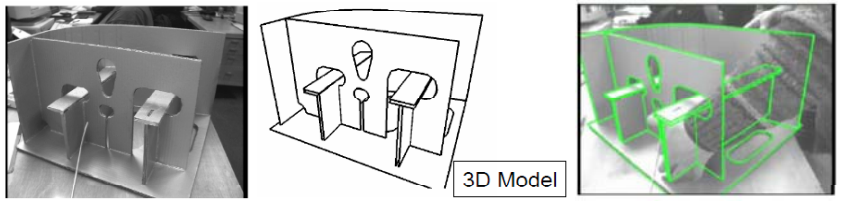


- Goal:
  estimate the extrinsic camera parameters
  for the image at time *t*
- Intrinsic parameters often assumed to be known
  (estimated beforehand using a calibration grid).
- Needs a manual initialization.
- Can use previous images, a motion model.

Model-based 3D tracking approaches

- Using intensity images:
  - Edges
  - ~~Template matching~~
  - Interest points
- Using depth images:
  - generate pose hypotheses and
    evaluate them on the observed depth/RGB-D data.
    (RGB-D = Red Green Blue – Depth; ex: the data provided by Kinect sensor)

Edge-based tracking



- RAPiD approach-based methods:
  - computationally efficient
  - relatively easy to implement
  - naturally stable to lighting changes

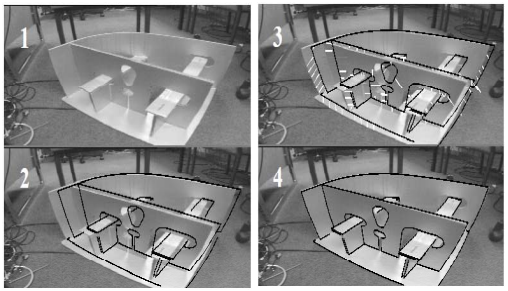
RAPiD: Harris MIT Press, 1992 + Drummond, PAMI, 2000

Edge-based tracking

- To determine the pose at time *t*:
  - The model is projected with respect to the pose
    determined at time *t-1*
  - The projected contours are sampled
  - For each point , search for the corresponding image
    contour as a image gradient maximum
  - Estimate the pose that minimizes:
    sum of 2D distances point ↔contour

Edge-based tracking

- Tracking system loop:
  - Image acquisition
  - Model rendering
  - Image measurement
  - Pose update

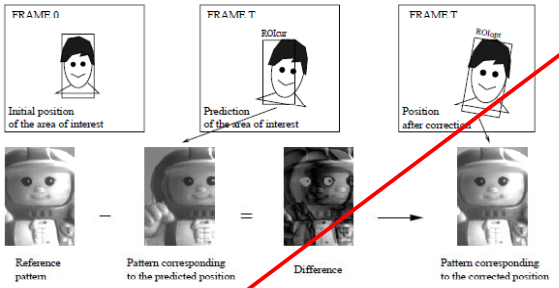


Visual Tracking for Augmented Reality, Georg Klein, King's College

Edge-based tracking

- Advantages:
  - Fast.
  - Robust to: partial occlusions, specular materials.
  - Accurate when enough contours are present in the image.
- Disadvantages:
  - Needs a 3D model.
  - Needs a manual initialization.
  - Assumption that edges of the 3D model correspond to gradient maxima in the image.
  - Can be disturbed by strong contours in the image.
  - Difficult to use on a textured object.

Template matching-based tracking



Frederic Jurie, Michel Dhome. "A simple and efficient template matching algorithm", ICCV, 2001

- The position of the target template in the first image is supposed to be known.
- The problem is then to estimate the position of this template in the subsequent images.
- By comparing the grey level values of the target template with the grey level values of the predicted region, it is possible to obtain the actual position of the template in the current image.
- This computation is possible because – during an off-line training stage – a relation between the variations of intensities and the variations of position has been learnt.

Feature point detection and tracking

- Detection:
  - Harris or FAST corners, SIFT or SURF feature points, ...
- Tracking: two general approaches:
  - Kanade-Lucas-Tomasi (KLT) tracker
    - Find a good point to track (Harris corner)
    - Use intensity second moment matrix and difference across frames to find displacement
    - Iterate and use coarse-to-fine search to deal with larger movements
    - (-) May loose features after a while
  - Detection in every frame and matching
    - (+) Better

Feature point detection & matching

What are local features ?

- Local feature
  - an image pattern which differs from its immediate neighborhood
- Image properties commonly considered:
  - intensity, color, texture
- Local features can be:
  - points, edgels, small image patches
- Typically, some measurements are taken from a region centered on a local feature and converted into descriptors

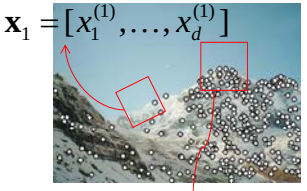
Why use local features ?

- Local (invariant) features are a powerful tool
- Possible usage examples:
  - They may have a semantic interpretation
    - edges in aerial images may correspond to roads
    - blobs in an tissue image may correspond to impurities
  - They provide a set of well localized and individually detectable anchor points (what the features represent is not actually relevant)
    - matching and tracking applications (KLT)
    - camera calibration & 3D reconstruction
  - They can be used as robust image representation
    - **object or scene recognition without the need for segmentation**

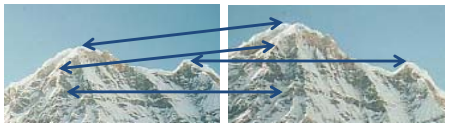
Local features: main components

- Detection: Identify the interest points

- Description: Extract vector feature descriptor surrounding each interest point



- Matching: Determine correspondence between descriptors in two views



slide by: Kristen Grauman, UT-Austin

Interest point detectors

- Detector / Extractor
  - the tool that extracts the features from the image
    - ex: corners, blobs, edges
- Interest point, Region or Local feature?
  - To localize features in images, a local neighborhood of pixels needs to be analyzed, giving all local features some implicit spatial extent.
  - Interest point:
    - For some applications (e.g., camera calibration or 3D reconstruction) this spatial extent is completely ignored in further processing, and only the location derived from the feature extraction process is used.
    - Location sometimes determined up to sub-pixel accuracy.
  - Region:
    - In most applications the features also need to be described, such that they can be identified and matched.
    - Often, this neighborhood is taken equal to the neighborhood used to localize the feature, but this need not be the case.
  - Local feature:
    - can be either points, regions or even edge points.

NOTE: in these slides "feature point" , "keypoint" and "interest point" are used interchangeably.

Ideal local features

- Ideally,
  one would like such local features to correspond to semantically meaningful object parts
- In practice, however, this is unfeasible,
  as this would require
  high-level interpretation of the scene content,
  which is not available at this early stage
- Instead,
  detectors select local features directly
  based on the underlying intensity patterns

Local features: desired properties

- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature has a distinctive description
- Compactness and efficiency
  - Many fewer features than image pixels
  - Should allow for time-critical applications
- Locality
  - A feature occupies a relatively small area of the image

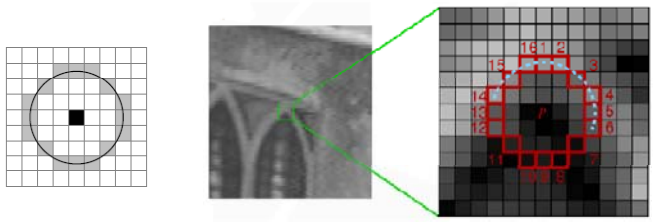
Interest point detectors

- Corner detectors
  - "corner score" is derived from the 2nd-order moment image gradient matrix
    - Moravec (1980), Harris & Stephens(1988), and Shi-Tomasi detectors (1994)
  - Features from Accelerated Segment Test (FAST) corner detector
- Blob detectors
  - use local extrema of the responses of certain filters as interest points
    - Laplacian of Gaussian applied at multiple image scales (Lindberg, 1994)
    - Local extrema of an image filtered with differences of Gaussians (Lowe, 1999, 2004)
    - Fast Hessian detector (Bay et al., 2008)
- Affine-invariant detectors
  - detectors that are invariant to affine changes (different scaling in different directions)
  - provide higher repeatability for large affine distortions
    - Lowe (2004); Mikolajczyk and Schmid (2002); Matas et al. (2002)

NOTE: usually non-maximum supression is applied to detector results
  - This has the effect of supressing all interest points that are not local maxima

FAST corner detector

- Features from Accelerated Segment Test (FAST)
  - Rosten and Drummond (2005, 2006)
- A highspeed corner detector.
- The algorithm operates on a discretized circle (16 points) around a candidate point ***p*** as shown below
- ***p*** is classified as a **corner** if there exists a contiguous arc of at least nine pixels that are all brighter or all darker than ***p*** by a threshold ***t***



FAST corner detector

- In contrast to other detectors, detection with the FAST algorithm does not inherently provide a measure of the "strength" of a feature
- In order to apply nonmaximum suppression, the following score is computed for each candidate point:

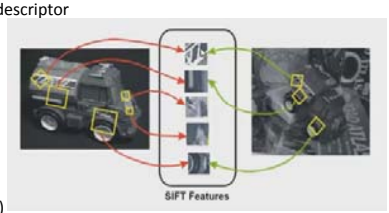
$$c(p) = \max\left\{ \sum_{q \in S_+} |I_q - I_p| - t, \sum_{q \in S_-} |I_q - I_p| - t \right\}$$

where

- S_+ / S_- are the subsets of pixels on the circle that are brighter / darker than p (by t)

Feature descriptors

- Early Approaches
  - The simplest and fastest descriptor could be a histogram of all pixel intensities.
  - A variety of features derived from the local image intensities have been proposed to derive robust feature descriptors .
- SIFT
  - The Scale-Invariant Feature Transform (SIFT) by Lowe (1999, 2004) is probably the most well-known and widely used local descriptor
  - It achieves invariance to changes in scale and rotation by operating in a local reference frame relative to a dominant scale and rotation that is computed from the image
  - The descriptor is based on local gradient histograms, sampled in a square grid around the keypoint
- SURF
  - A faster alternative to SIFT, proposed by Bay et al. (2008) adopting similar approaches for scale and rotation invariance combined with efficient approximations to speed up the computation
  - The descriptor is derived from responses to box filters instead of the computationally more expensive Gaussian filters used in SIFT



SIFT

- D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", IJCV, 2004
- Motivation: providing repeatable robust features for
  - Tracking,
  - Recognition,
  - Panorama Stitching, etc.
- Features:
  - Invariant to scaling and rotation.
  - Partly invariant to illumination and viewpoint changes.

SIFT - overview

- Detector
  - Create a scale space of images
    - Construct a set of progressively Gaussian blurred images
    - Take differences to get a "difference of Gaussian" pyramid (similar to a Laplacian of Gaussian)
  - Find local extrema in this scale-space. Choose keypoints from the extrema
- Accurate keypoint location
  - sub-pixel locate
  - filter response
    - remove low contrast features and
    - features primarily along an edge
- Keypoint orientation assignment
- Keypoint descriptor
  - For each keypoint, in a 16x16 window, find histograms of gradient directions
  - Create a feature vector out of these

- http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html
- http://www.vlfeat.org/api/sift.html
- http://en.wikipedia.org/wiki/Scale-invariant_feature_transform#Comparison_of_SIFT_features_with_other_local_features

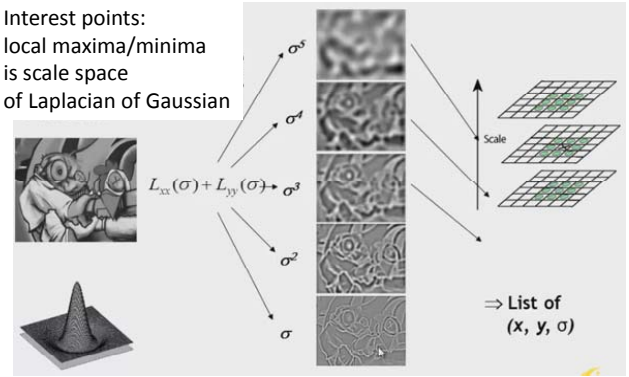
SIFT - overview

- A SIFT keypoint is
  - a circular image region
  - with an orientation.
- It is described by a special structure which has many attributes, like:
  - the keypoint center coordinates (x,y)
  - its scale (the radius of the region)
  - its orientation (an angle)
  - response that specifies strength of keypoints
  - (...)
- The SIFT detector uses as keypoints image structures which resemble "blobs"
- By searching for blobs at multiple scales and positions, the SIFT detector is invariant (or, more accurately, covariant) to
  - translation,
  - rotations, and
  - re-scaling of the image.

For more information, see for example: http://www.vlfeat.org/api/sift.html

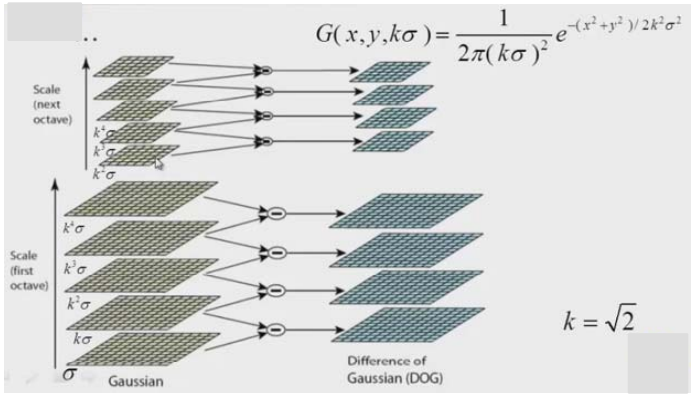
SIFT – Scale space extrema detection

Interest points:
local maxima/minima
is scale space
of Laplacian of Gaussian



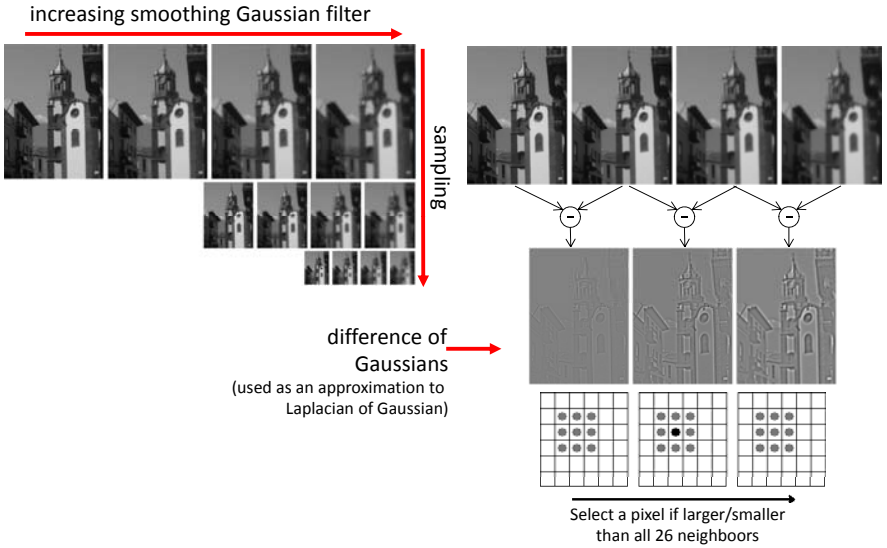
Slide by Mubarak Shah

SIFT – Scale space extrema detection



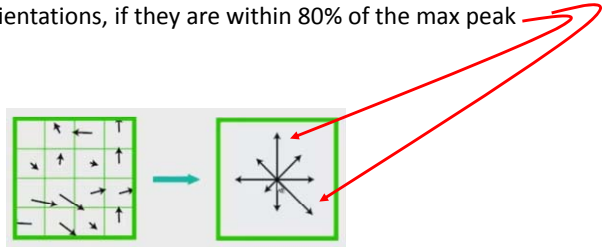
Slide by Mubarak Shah

SIFT – Scale space extrema detection



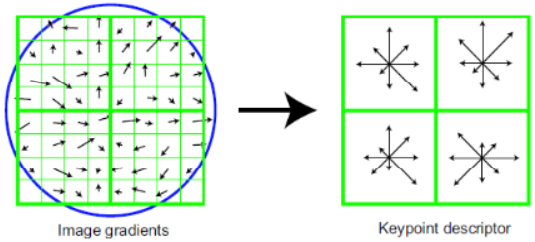
SIFT – orientation assignment

- To achieve rotation invariance:
- Create a weighted direction histogram in a neighborhhod of a keypoint (36 bins – each bin = 10 degrees)
  - Weights are gradient magnitudes
- Select the peak of the histogram as direction of the keypoint
  - note: several peaks may be considered to exist at same location with different orientations, if they are within 80% of the max peak

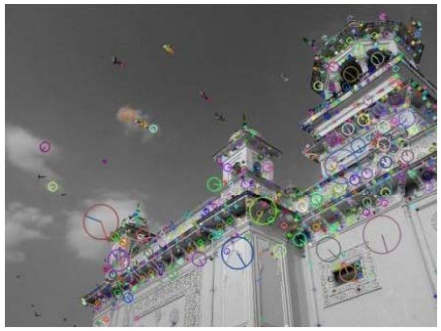


SIFT descriptor

- A sample region is shown on the left where magnitude and orientation of gradient need to be computed.
- Part of a sample descriptor is extracted and shown on the right.
- Each set of arrows is composed of 8 directions and the length of each arrow is related to the sum of magnitudes corresponding to that direction.
- SIFT descriptor:
  - is a 128 dimension feature vector composed of image gradient magnitude and orientation of the points around the keypoints.



SIFT - results



source: https://docs.opencv.org/3.3.0/da/df5/tutorial_py_sift_intro.html

- Scale represented by the size of the circle
- Notice the associated orientation

Object recognition using SIFT

Other interest point detectors/descriptors

- Detectors
  - Harris
  - Shi-Tomasi
  - FAST (Features from Accelerated Segment Test)
    - http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_fast/py_fast.html
  - MSER
  - CenSurE
- Descriptors
  - BRIEF (Binary Robust Independent Elementary Features)
    - http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_brief/py_brief.html
  - Randomized trees
  - Ferns
  - PCA-SIFT (http://www.cs.cmu.edu/~yke/pcasift/)
  - GLOH (Gradient Location and Orientation Histogram)
  - HOG (Histogram of Oriented Gradients)
- Detectors & Descriptors
  - SIFT (Scale-Invariant Feature Transform)
    - http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html
  - SURF (Speeded-Up Robust Features)
    - http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html
  - ORB (Oriented FAST and Rotated BRIEF)
  - BRISK (Binary Robust Invariant Scalable Keypoints)
    - http://docs.opencv.org/modules/features2d/doc/feature_detection_and_description.html

http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_table_of_contents_feature2d/py_table_of_contents_feature2d.html

Feature matching

- The next task is to find correspondences between the keypoints found in both images.
- The extracted features are placed in a structure that can be searched efficiently (such as a kd-tree).
  - a k-d tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space.
  - k-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key (e.g. range searches and nearest neighbor searches).
  - k-d trees are a special case of binary space partitioning trees.
- Usually it is sufficient to look up all local feature-descriptors and match each one of them to his corresponding counterpart from the other image.
  - Note: two images from a scene don't necessarily have the same number of feature-descriptors; some feature points may have no matching point

Feature matching

- Brute-force matcher
  - Brute-Force (BF) matcher is simple
  - It takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation
  - and the closest one is returned
- FLANN-based matcher
  - FLANN stands for **F**ast **L**ibrary for **A**pproximate **N**earest **N**eighbors
  - It implements a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features
  - It works more faster than BF matcher for large datasets

http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html

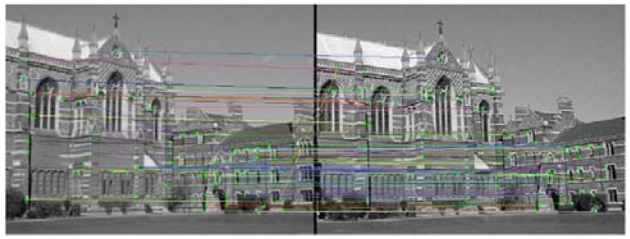
Distance measures

- Euclidean distance
  - Is the "ordinary" distance between two points in Euclidean space, i.e., the length of the line segment connecting them.
  - In Euclidean n-space: $\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2}$
- Hamming distance
  - In information theory, the Hamming distance between two "strings" of equal length is the number of positions at which the corresponding symbols are different.
  - The Hamming distance between:
    - "**Jo**h**n**" and "**Jo**a**o**" is 2.
    - **10**1**11**0 and **10**0**10**0 is 2.
    - **1**2**3**4 and **2**2**43** is 3.
  - J. Ventura, T. Hollerer, "Fast and scalable keypoint recognition and image retrieval using binary codes", *IEEE Workshop onApplications of Computer Vision (WACV), 2011*

Consistency checks

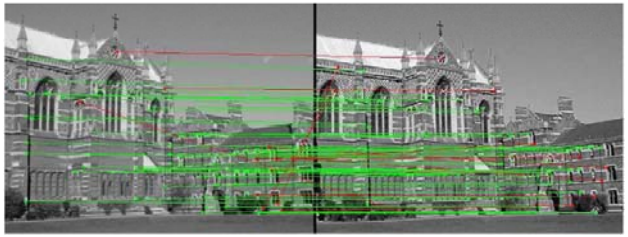
- Global constraints
  - Homography (RANSAC)
  - ...others ? (Epipolar constraint + RANSAC)
- Semi-local constraints
  - Topologic constraints (Ferrari et al., 2004)
    - The first constraint states that for a triplet of feature matches, the center of a feature **m1** should lie on the same side of the directed line going from the center **m2** of the second feature to the center **m3** of the third feature, in both views
      - This constraint holds always if the three features are coplanar, and in the vast majority of cases if they are not.
  - others ...

RANSAC for refining matching



Initial matching results

After RANSAC: green points and lines represent inliers and red ones are outliers



... although the points are not exactly coplanar ...!

Topological constraints

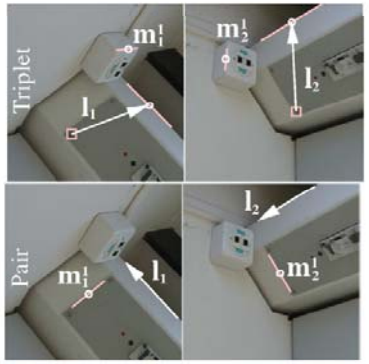
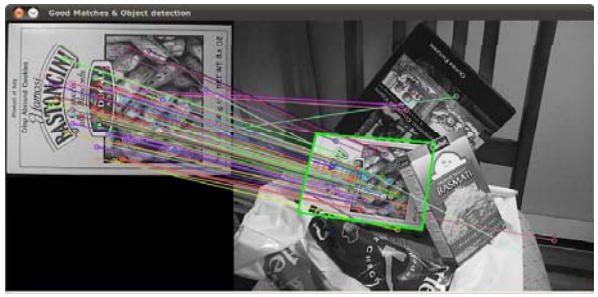


Figure 2. *Top: a triplet of features (two segments, with midpoints 'o', and a region center, marked with a square). $\mathbf{m}^1$ lies on the same side of the directed line* l *in both views. Bottom: a pair of segments, where one takes the role of the directed line* l.

source: H. Bay, V. Ferrari, L.Van Gool, "Wide-Baseline Stereo Matching with Line Segments"

Features 2D + Homography to find a known object

- Example from OpenCV site. Uses:
  - SURF extractor and descriptor
  - Flann-based matcher
  - Perspective transform to map object corners to scene (and then draw the green rectangle)



The detected object is highlighted in green

http://docs.opencv.org/doc/tutorials/features2d/feature_homography/feature_homography.html

OpenCV feature detection, description & matching

Feature Detectors:
- SIFT
- SURF
- MSER
- FAST
- BRISK
- ORB
- AKAZE
- KAZE
- STAR
- MSD

Feature Descriptors:
- SIFT
- SURF
- BRIEF
- BRISK
- ORB
- KAZE
- AKAZE
- FREAK
- DAISY
- LATCH
- LUCID

Feature Matchers:
- BF
- FLANN

- Correspondence rejection: RANSAC (Random Sample Consensus).

Pose estimation:
- solvePnP & solvePnPRansac
- decomposeHomographyMat

An example of how to use solvePnP for planar augmented reality can be found at opencv_source_code/samples/python2/plane_ar.py

3D tracking using depth data

- No need for a predefined CAD model or similar.
- Tracking and reconstruction of unknown 3D rigid objects is done simultaneously.
- Examples:
  - Structure from Motion (SFM) which simultaneously estimating both 3D geometry (structure) and camera pose (motion).
  - A more applicable mechanism in robot system is incremental SFM, known as Simultaneous Localization and Mapping (SLAM).
  - Model-based approach, which generates pose hypotheses and evaluates them on the observed depth/RGB-D data
    - Kinect Fusion (KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera, 2011)
    - STAR3D (Simultaneous Tracking and Reconstruction of 3D Objects Using RGB-D Data, 2013)

Kinect Fusion

- The results shown by KinectFusion [1].
- B) is Phong shaded reconstructed 3D model.
- C) shows the texture mapped 3D with real-time particles simulation.
- D) is a multi-touch interaction with the reconstructed surface.
- E) demonstrates how it can perform a real- time segmentation and tracking.
  - The interaction and segmentation both rely on a static environment built before motions happen.

[1] - S.Izadi et al., "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera", Proc. 24th annual ACM symposium on User Interface Software and Technology, pp.559-568, 2011

http://research.microsoft.com/apps/video/default.aspx?id=152815&r=1
https://www.youtube.com/watch?v=bRgEdqDiOuQ

http://pointclouds.org/news/2011/12/29/augmented-reality-with-kinect-fusion/
http://augmentedrealityoverview.blogspot.pt/2011/10/kinect-fusion.html
http://healthandscience.journalism.cuny.edu/portfolio/ieee-spectrum-kinect-brain-scan-augmented-reality-for-neurosurgeons/

Kinect Fusion – processing steps

Assumptions:
- *Static environment*
- *Small motion between frames*

- Surface Measurement
  - Raw depth map returned by Kinect is filtered to reduce noise
  - After some processing steps, the result is a point cloud with vertices and normals
- Camera Tracking
  - The camera pose is tracked for each depth frame, by estimating a single 6DOF transform that closely aligns points of two consecutive frames
  - The alignment of frames is completed by the Iterative Closest Points (ICP) algorithm.
- Volumetric Integration
  - After alignment is completed and camera pose is tracked, the raw data is integrated at this step to reconstruct the 3D model
  - The KinectFusion model integrates the data by using one volumetric representation ,TSDF; the TSDF (Truncated Signed Distance Function) volume is a 3D grid of voxels stored in GPU
- Ray Casting
  - A prediction of the current global surface is obtained during the rendering of the global 3D model, by using ray casting
  - The aim of ray casting is to predict a more accurate global vertex and normal map

ICP – Iterative Closest Point algorithm

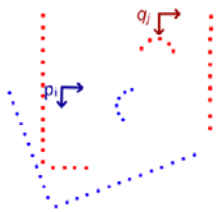
- Several problems:
  - find object in a scene
  - estimate motion of sensor
  - compare two scenes
  - merge observations into a map
  - ...
- Solution:
  - ICP: Iterative Closest Point algorithm

ICP – Iterative Closest Point algorithm

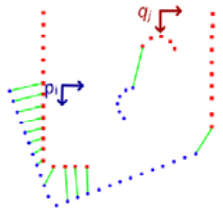
- Definition
  - find transformation between 2 set of points
- Input
  - reference point cloud
  - data point cloud
- Output
  - transformation between reference and data
    - 3 degrees of freedom in 2D
    - 6 degrees of freedom in 3D
- Outline
  - find corresponding points in both clouds
  - compute transformation minimizing error
  - move data points according to the transformation
  - loop...

Francis Colas, "Iterative Closest Point Algorithm", ETH, 2011

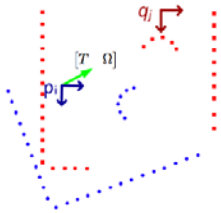
ICP – in 2D



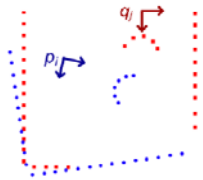
1- point clouds



2- transformation minimizing error



3- move data points according to transformation



4- 1st iteration result

Kinect Fusion links

- **How Kinect Fusion and Kinect Work**
  - http://research.microsoft.com/en-us/projects/surfacerecon/
  - http://research.microsoft.com/apps/pubs/default.aspx?id=155378
  - https://www.youtube.com/watch?v=zzb_RQWrt6I
  - http://ags.cs.uni-kl.de/fileadmin/inf_ags/3dcv-ws12-13/20130115/KinectFusion_Slides_ISMAR2011.pdf
  - http://spectrum.ieee.org/automaton/robotics/medical-robots/microsoft-kinect-fusion-augmented-reality-neurosurgeons
  - http://pointclouds.org/news/2011/12/29/augmented-reality-with-kinect-fusion/
  - http://augmentedrealityoverview.blogspot.pt/2011/10/kinect-fusion.html
  - http://healthandscience.journalism.cuny.edu/portfolio/ieee-spectrum-kinect-brain-scan-augmented-reality-for-neurosurgeons/

References

- T. Tuytelaars and K. Mikolajczyk, "Local Invariant Feature Detectors: A Survey", Foundations and Trends in Computer Graphics and Vision, Vol. 3, No. 3, pp. 177–280, 2007
- S. Gauglitz, T.Höllerer, M. Turk, "Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking", International Journal of Computer Vision, 94, pp. 335–360, 2011
- D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, D. Schmalstieg, "Real-Time Detection and Tracking for Augmented Reality on Mobile Phones" (May 2010)
- E. Rosten and T. Drummond, "Machine learning for high-speed corner detection" (May 2006).
- Other papers/sites that were referred in previous slides
- Some slides adapted from courses taught by
  - Mark Billinghurst, HIT Lab, New Zeland
  - Vincent Lepetit, CVLab-EPFL, Switzerland