Keypoints and Descriptors[1]

Dalian, China, August 2014

See Related Material in
Reinhard Klette: Concise Computer Vision
Springer-Verlag, London, 2014
Related to VBDA

[1]See last slide for copyright information.

Agenda

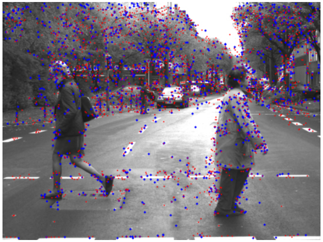
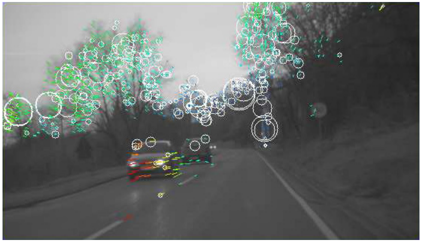
Keypoint, Descriptor, Feature

A keypoint and a descriptor define a feature

Examples: SIFT, SURF, and ORB

BRIEF and FAST are needed for defining ORB

Features are tracked over time: KLT, particle filter, or Kalman filter are possible tools

Left: DoG scale space keypoints
Right: Keypoints with disks of influence (radius = scale where keypoint has been detected

Invariance

Images are taken under varying illumination, different viewing angles, at different times, under different weather conditions, and so forth

When taking an aerial shot from an airplane, we do have a random rotation of shown objects, and *isotropy* (rotation invariance) is of interest

In outdoor scene analysis, we often request types of invariance with respect to some operations, such as illumination changes or recording images at different distances to the object of interest

Agenda

Keypoint

A *keypoint* (or *interest point*) is defined by some particular image intensities “around” it, such as a corner

A keypoint can be used for deriving a *descriptor*

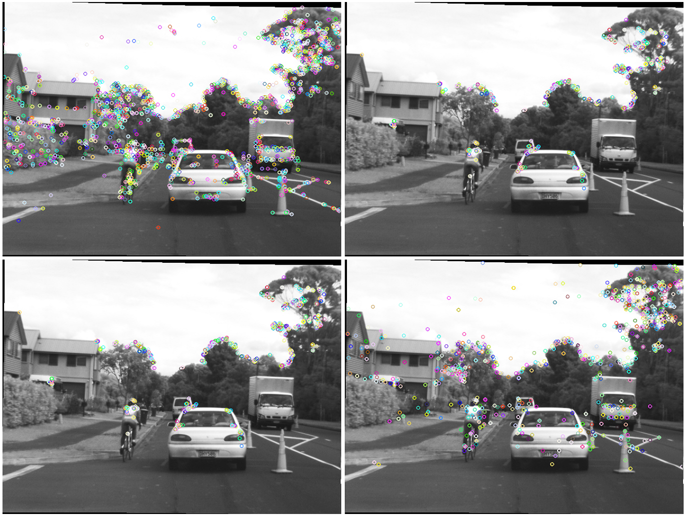
Not every keypoint detector has its particular way for defining a descriptor

A descriptor is a finite vector which summarizes properties for the keypoint

A descriptor can be used for classifying the keypoint

Keypoint and descriptor together define a *feature*

Four Examples



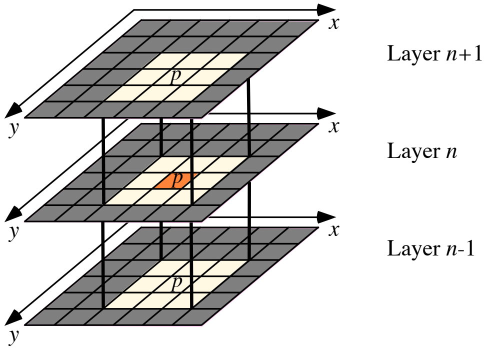
Four of the keypoint detectors in `OpenCV`: FAST, ORB, SIFT, SURF

Keypoints Defined in LoG or DoG Scale Space

Recall: DoG for scale σ and scaling factor $a > 1$ for combining two subsequent layers in Gaussian scale space:

$$D_{\sigma,a}(x,y) = L(x,y,\sigma) - L(x,y,a\sigma)$$

Use initial scale $\sigma > 0$; apply scaling factors a^n, $n = 0, 1, 2, \ldots$ for generating a finite number of Layers n in DoG scale space



3D Data Array and Disk of Influence

Layers D_{σ,a^n}, $n = 0, \ldots, m$, define a 3D data array

Each array position (x, y, n) in this 3D array has 17 or 26 adjacent array positions

Array position (x, y, n) and those 17 or 26 adjacent positions define the *3D neighborhood* of (x, y, n)

A *keypoint* is detected at $p = (x, y)$ if there is a Layer n, $0 \leq n \leq m$, such that $D_{\sigma,a^n}(x, y)$ defines a local minimum or local maximum within the 3D neighborhood of (x, y, n)

With a detected keypoint in the original image I at a pixel location $p = (x, y)$, we also have the scale $\sigma \cdot a^n$ where it has been detected

This scale defines the radius of the *disk of influence* for this keypoint p

Keypoints at Subpixel Accuracy

Keypoints detected in scale space in a layer defined by scale $\sigma \cdot a^n$

Are at pixel locations (i.e. with integer coordinates)

1. Interpolate a 2D second-order polynomial $g(x, y)$ to the detected keypoint and its four 4-adjacent neighbors
2. Take the derivatives of $g(x, y)$ in x- and y-direction
3. Solve the resulting equational system for a subpixel-accurate minimum or maximum

Agenda

Corresponding Keypoints

Given: Sets of keypoints in subsequent frames

Compare detected sets of keypoints in two subsequent frames of an image sequence using locations and descriptors

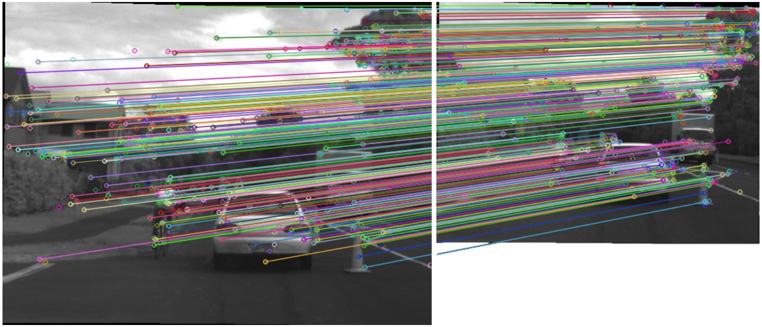
Find corresponding keypoints

There will be *outliers* which have no corresponding keypoint

Inliers have corresponding points

Correspondence e.g. also a (global) set problem (not only a point-by-point problem): There is a global pattern of keypoints, and we want to match this global pattern with another global pattern of keypoints

SIFT Example



Sets of SIFT keypoints in original and demagnified image (not a time sequence)

RANSAC-match of corresponding keypoints

Random Sample Consensus

RANSAC is an iterative estimation technique of parameters of an assumed mathematical model

Given is a set of data, called *inliers*, which follow the model, but there is also additional data, called *outliers*, which do not follow the model

For applying RANSAC, the probability of selecting inliers needs to be reasonably high

Example 1: Inliers and outliers as a noisy representation of a straight line $y = ax + b$

Task: estimate a and b (an alternative to Hough transform)

Example 2: Sets of keypoints in two different images; the model is a geometric transform (example of a *matching problem*), e.g. an affine transform

Test

Need a test for evaluating whether data *satisfy* or *fit* the parametrized model

Here: keypoint p in one image I is mapped by parametrized affine transform onto q in other image J

Test: If there is a keypoint r in J at distance $d_2(q, r) \leq \varepsilon$ then we say that p satisfies the given parametrized affine transform

Tolerance threshold $\varepsilon > 0$ determines whether data fit the model

RANSAC Algorithm

Initialization: select random subset S of given data as inliers; fit the model by estimating model parameters

Test: Test the parametrized model for *all* the other data

Define consensus set: All the data which satisfy the model go into a *consensus set*

Cardinality check: Compare the cardinality of the consensus set against the cardinality of all data

Stop criterion: Percentage is reasonably high

Refined model: Otherwise, estimate updated model parameters based on consensus set

Continue with refined model: if cardinality of newly established consensus set does not increase then go back to initialization step and select another random subset S

RANSAC for Feature Matching

Feature defined by keypoint and descriptor

Initial set S can be three randomly selected keypoints in image I

Search for three keypoints with reasonably matching descriptors in image J

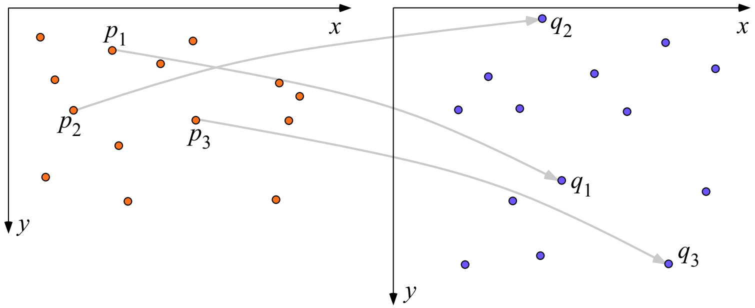
Estimate affine transform: Point $p = (x, y, 1)$ in homogeneous coordinates in image I is mapped into $q = (u, v, 1)$ in image J

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{aligned} u &= r_{11}x + r_{12}y + t_1 \\ v &= r_{21}x + r_{22}y + t_2 \end{aligned}$$

Six unknowns, solvable for non-collinear points p_1, p_2, and p_3 in I

Three Pairs



Definition of an affine transform by three pairs of points in images *I* (on the left) and *J*

Consensus Set

For calculated affine transform $A(p) = q$, we apply A now to *all* the keypoints p in I

We obtain points $A(p)$ in J

Point p goes into the consensus set if there is a keypoint q in J in a Euclidean distance less then $\varepsilon > 0$ to $A(p)$ with a "reasonable" match of descriptors, defining the expected image q_p of p in J

Obviously, initially used points p_1, p_2, and p_3 pass this test

Update

In general, consensus set now with more than just 3 points in I

Update the affine transform by calculating (using linear least-squares) the optimum transform for all the established pairs p in I and q_p in J

Defines a refined affine transform

Value ε cannot be "very small"; this would not allow to move away from the initial transform (for a better match between both sets of keypoints)

Agenda

Scale-Invariant Feature Transform

We detect keypoints in DoG or LoG scale space

For keypoint $p \in \Omega$, we also have scale $\sigma \cdot a^n$ which defines the radius $r_p = \sigma \cdot a^n$ of the disk of influence for this keypoint

Taking this disk, centered at p, in all layers of the scale space, we define a *cylinder of influence* for the keypoint

Intersection of this cylinder with the input image is a disk of radius r_p centered at p

Eliminating Keypoints with Low Contrast or on an Edge. Detected keypoints in low contrast regions are removed by calculating the contrast value at the point

For deciding whether keypoint p is on an edge, consider gradient $\triangle I(p) = [I_x(p), I_y(p)]^\top$: If both components differ significantly in magnitude then we conclude that p is on an edge

Keypoints at Corners

Another option: take only keypoints at a corner in the image

A corner can be identified by eigenvalues λ_1 and λ_2 of the Hessian matrix at pixel location p

If magnitude of both eigenvalues is "large" then we are at a corner; one large and one small eigenvalue identifies a step-edge, and two small eigenvalues identify a low-contrast region

After having already eliminated keypoints in low-contrast regions, we are only interested in the ratio

$$\frac{\lambda_1}{\lambda_2} = \frac{(I_{xx} + I_{yy})^2 + 4I_{xy}\sqrt{4I_{xy}^2 + (I_{xx} - I_{yy})^2}}{(I_{xx} + I_{yy})^2 - 4I_{xy}\sqrt{4I_{xy}^2 + (I_{xx} - I_{yy})^2}}$$

for deciding corner versus edge

Descriptors

Descriptor $\mathbf{d}(p)$ for remaining keypoint p:

Scale-invariant feature transform (SIFT) aims at rotation invariance, scale invariance (actually addressing “size invariance”, not really invariance w.r.t. scale σ), and invariance w.r.t. brightness variations

Rotation-Invariant Descriptor. Disk of influence with radius $r_p = \sigma \cdot a^n$ in layer $D_{\sigma,a^n}(x, y)$ is analyzed for a *main direction* along a main axis and

then rotated such that the main direction coincides with a (fixed) predefined direction

Examples: Use main axis defined by moments, or just gradient vector at p

Main Axis Method of SIFT

SIFT applies a heuristic approach

For locations (x, y) in the disk of influence in layer $L(x, y) = D_{\sigma,a^n}(x, y)$, centered at keypoint p, a local gradient is approximated by using

$$\begin{aligned} m(x,y) &= \sqrt{[L(x,y+1) - L(x,y-1)]^2 + [L(x+1,y) - L(x-1,y)]^2} \\ \theta(x,y) &= \mathrm{atan2}\left([L(x,y+1) - L(x,y-1)], [L(x+1,y) - L(x-1,y)]\right) \end{aligned}$$

Directions are mapped onto 36 counters, each representing an interval of 10 degrees

Counters have initial value 0; if a direction is within the 10 degrees represented by a counter, then the corresponding magnitude is added to the counter

Altogether, this defines a *gradient histogram*

Dominant Direction

Local maxima in counter values, being at least at 80% of the global maximum, define *dominant directions*

If more than one dominant direction, then the keypoint is used in connection with each of those dominant directions

Rotate disk of influence such that detected dominant direction coincides with a (fixed) predefined direction

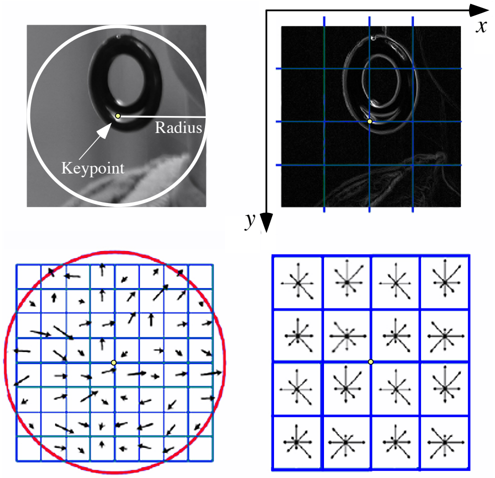
Brightness Invariance

Describe disk of influence in the input image (and not for the layer where the keypoint has been detected)

In general: we could apply any of the transforms discussed before for removal of lighting artifacts

SIFT calculates features for gradients in the disk of influence by subdividing this disk into square windows; for a square window in the input image we generate a gradient histogram as defined above for identifying dominant directions, but this time for intervals of 45 degrees, thus only eight counters, each being the sum of gradient magnitudes

4×4 Gradient Histograms



Square containing a disk of influence, gradient map, and sketches of detected gradients and of 16 gradient histograms

Scale Invariance

Partition the rotated disk of influence in the input image into 4×4 squares (geometrically "as uniform as possible")

For each of the 16 squares we have a vector of length 8 representing the counter values for the gradient histogram for this square

By concatenating all 16 vectors of length 8 each we obtain a vector of length 128

This is the SIFT descriptor $\mathbf{d}_{SIFT}(p)$ for the considered keypoint p

SURF Masks and the Use of Integral Images

Detector *speeded-up robust features* (SURF) follows similar ideas as SIFT

Designed for better run-time performance

Utilizes the integral images I_{int} and simplifies filter kernels

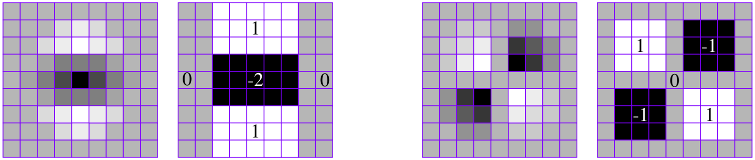


Illustration for $\sigma = 1.2$, the lowest scale, and 9×9 discretized and cropped Gaussian second-order partial derivatives and corresponding filter kernels in SURF

SURF Masks

Two of the four used masks (or filter kernels) are illustrated above

SURF's masks for x-direction and the other diagonal direction are analogously defined

Size of the mask corresponds to the chosen scale

After 9×9, SURF uses then masks of sizes 15×15, 21×21, 27×27, and so on

Values in Filter Kernels

Values in those filter kernels are either 0, -1, +1, or -2

Values -1, +1, and +2 are constant in rectangular subwindows W of the mask

This allows us to use integral images for calculating time-efficiently the sum S_W of all intensity values in W

It only remains to multiply the sum S_W with the corresponding coefficient (i.e., value -1, +1, or -2)

Sum of those three or four products is then the convolution result at the given reference pixel for one of the four masks

Scales and Keypoint Detection

Value $\sigma = 1.2$ is chosen for the lowest scale (i.e. highest spatial resolution) in SURF

Convolutions at a pixel location p in input image I with four masks approximate the four coefficients of the Hessian matrix

Four convolution masks produce values $D_{xx}(p, \sigma)$, $D_{xy}(p, \sigma)$, assumed to be equal to $D_{yx}(p, \sigma)$, and $D_{yy}(p, \sigma)$

$$S(p, \sigma) = D_{xx}(p, \sigma) \cdot D_{yy}(p, \sigma) - [c_\sigma \cdot D_{xy}(p, \sigma)]^2$$

as an approximate value for the determinant of the Hessian matrix at scale σ

With $0 < c_\sigma < 1$ is a weighting factor which could be optimized for each scale; SURF uses constant $c_\sigma = 0.9$

Keypoint p detected by a local maximum of a value $S(p, \sigma)$ within a $3 \times 3 \times 3$ array of S-values, analogously to keypoint detection in LoG or DoG scale space

SURF Descriptor

SURF descriptor is a 64-vector of floating point values

Combines local gradient information, similar to the SIFT descriptor

Uses weighted sums in rectangular subwindows (known as *Haar-like features*

Windows around the keypoint for simple and more time-efficient approximation of gradient values

FAST, BRIEF, ORB

Oriented robust binary features (ORB) based on *binary robust independent elementary features* (BRIEF) and keypoint detector FAST; both together characterize ORB

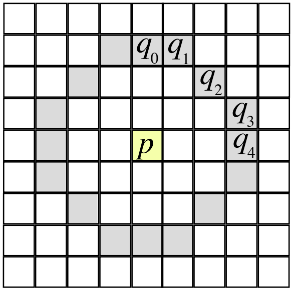
Binary Patterns

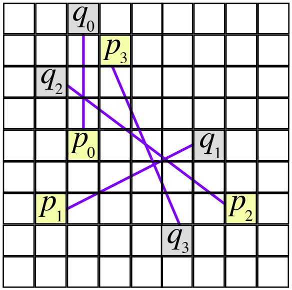
BRIEF reduces a keypoint descriptor from a 128-vector (such as defined for SIFT) to just 128 bits

Given floating-point information is binarized into a much simpler representation

Same idea has been followed when designing the census transform, when using *local binary patterns* (LBPs), or when proposing simple tests for training a set of classification trees

LBP and BRIEF





Pixel location p and 16 pixel locations q around p; $s(p,q) = 1$ if $I(p) - I(q) > 0$; 0 otherwise;
$s(p,q_0) \cdot 2^0 + s(p,q_1) \cdot 2^1 + \ldots + s(p,q_15) \cdot 2^{15}$ is LBP code at p

BRIEF uses an order of random pairs of pixels within a square neighborhood; here four pairs (p_i, q_i), defining
$s(p_0,q_0) \cdot 2^0 + s(p_1,q_1) \cdot 2^1 + s(p_2,q_2) \cdot 2^2 + s(p_3,q_3) \cdot 2^3$

BRIEF

LBP defined for a selection of n pixel pairs (p, q), selected around the current pixel in some defined order in a $(2k+1) \times (2k+1)$ neighborhood (e.g., $k = 4$ to $k = 7$)

After Gaussian smoothing defined by $\sigma > 0$ in the given image I

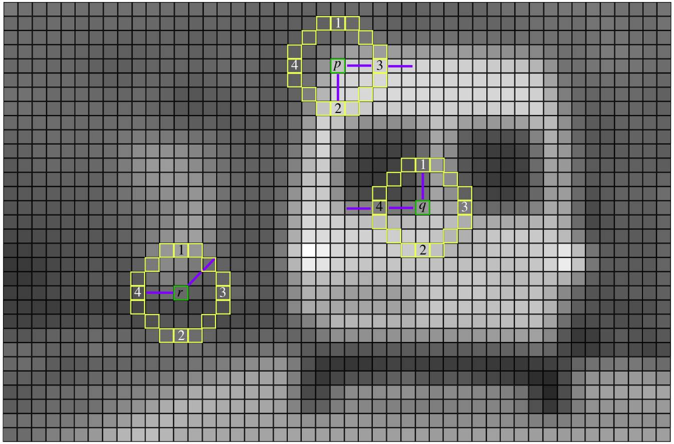
Order of those pairs, parameters k and σ define a BRIEF descriptor

Smoothing can be minor (i.e. a small σ) and the original paper suggested a random order for pairs of pixel locations

Scale or rotation invariance was not intended by the designers of the original BRIEF

FAST

Features from an accelerated segment test FAST: corner by considering image values on digital circle around given *p*



16 image values on a circle of radius $\rho = 3$

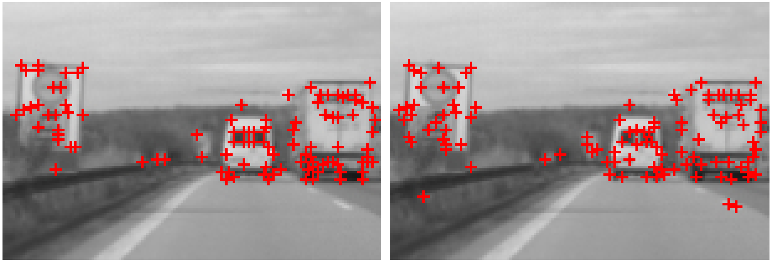
Cornerness test

Pixels p, q, and r are at intersections of edges; directions of those edges are indicated by the shown blue lines

Cornerness test

Value at center pixel needs to be darker (or brighter) compared to more than 8 (say, 11 for really identifying a corner and not just an irregular pixel on an otherwise straight edge) subsequent pixels on the circle, and "similar" to values of remaining pixels on circle

Harris versus FAST



Left: Detected corners using the Harris detector
Right: Corners detected by FAST

Corner Detector by Harris and Stephens

This *Harris detector* uses first-order derivatives of smoothed $L(.,.,\sigma)$, for some $\sigma > 0$

$$\mathbf{G}(p,\sigma) = \begin{bmatrix} L_x^2(p,\sigma) & L_x(p,\sigma)L_y(p,\sigma) \\ L_x(p,\sigma)L_y(p,\sigma) & L_y^2(p,\sigma) \end{bmatrix}$$

Eigenvalues λ_1 and λ_2 of $\mathbf{G}$ represent changes in intensities in orthogonal directions in image I

For small $a > 0$ (e.g., $a = 1/25$) consider *cornerness measure*

$$\mathcal{H}(p,\sigma,a) = \det(\mathbf{G}) - a \cdot \mathrm{Tr}(\mathbf{G})$$

$$\mathcal{H}(p,\sigma,\lambda) = \lambda_1\lambda_2 - a \cdot (\lambda_1 + \lambda_1)$$

One large and one small eigenvalue (such as on a step edge), then $\mathcal{H}(p,\sigma,a)$ remains reasonably small

Back to FAST: Time Efficiency

First: compare value at center pixel against values at locations 1, 2, 3, and 4 in this order

If still possible that the center pixel passes the cornerness test, we continue with testing more pixels on the circle

Original FAST paper proposes to learn a decision tree for time optimization

FAST detector in `OpenCV` (and also the one in `libCVD`) applies SIMD instructions for concurrent comparisons, which is faster then the use of the originally proposed decision tree

Non-maxima Suppression

For a detected corner, calculate maximum difference T between value at center pixel and values on discrete circle being classified as "darker" or "brighter" such that we still detect this corner

Non-maxima suppression deletes then in the order of differences T

ORB

ORB also for *oriented FAST and rotated BRIEF*

Combines keypoints defined by extending FAST and an extension of descriptor BRIEF

1. Multi-scale detection following FAST (for scale invariance), calculates a dominant direction
2. Applies calculated direction for mapping BRIEF descriptor into a steered BRIEF descriptor (for rotation invariance)

Authors of ORB suggest ways for analyzing variance and correlation of components of steered BRIEF descriptor

Test data base can be used for defining a set of BRIEF pairs (p_i, q_i) which de-correlate the components of the steered BRIEF descriptor for improving the discriminative performance of the calculated features

Multi-Scale, Harris Filter, and Direction

Define discrete circle of radius $\rho = 9$

(Above: FAST illustrated for discrete circle of radius $\rho = 3$)

Scale pyramid of input image is used for detecting FAST keypoints at different scales

Harris filter. Use cornerness measure (of Harris detector) to select T "most cornerness" keypoints at those different scales, where $T > 0$ is a pre-defined threshold for numbers of keypoints

Moments m_{10} and m_{01} of the disk S, defined by radius ρ, specify direction

$$\theta = \mathrm{atan2}(m_{10}, m_{01})$$

By definition of FAST it can be expected that $m_{10} \neq m_{01}$

Let $\mathbf{R}_\theta$ be the 2D rotation matrix about angle θ

Descriptor with a Direction

Pairs (p_i, q_i) for BRIEF, with $0 \leq i \leq 255$, are selected by a Gaussian distribution within the disk used (of radius ρ)

Form matrix **S** which is rotated into

$$\mathbf{S}_\theta = \mathbf{R}_\theta \mathbf{S} = \mathbf{R}_\theta \begin{bmatrix} p_0 & \cdots & p_{255} \\ q_0 & \cdots & q_{255} \end{bmatrix} = \begin{bmatrix} p_{0,\theta} & \cdots & p_{255,\theta} \\ q_{0,\theta} & \cdots & q_{255,\theta} \end{bmatrix}$$

Steered BRIEF descriptor calculated as the sum $s(p_{0,\theta}, q_{0,\theta}) \cdot 2^0 + \ldots + s(p_{255,\theta}, q_{255,\theta}) \cdot 2^{255}$, where s is defined as above

By going from original BRIEF to the steered BRIEF descriptor, values in the descriptor become more correlated

256 BRIEF Pairs

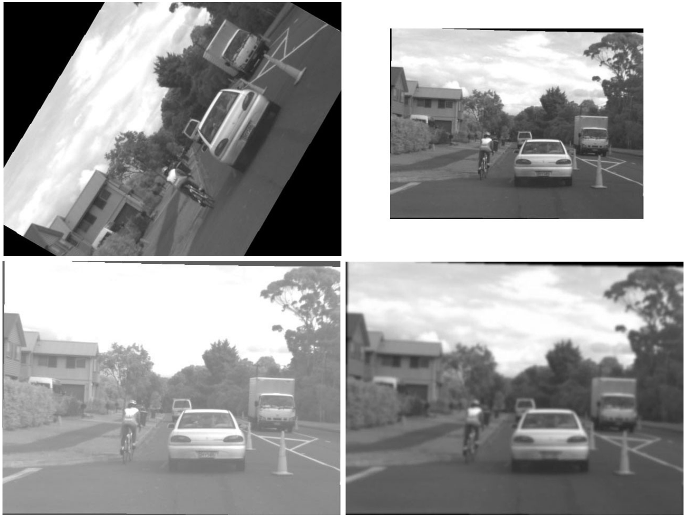
For time-efficiency reasons, a used pattern of 256 BRIEF pairs (generated by a Gaussian distribution) is rotated in increments of $2\pi/30$, and all those patterns are stored in a look-up table

This eliminates the need for an actual rotation; the calculated θ is mapped on the nearest multiple of $2\pi/30$

Agenda

Evaluation of Features

Evaluate feature detectors with respect to invariance properties



Caption

Rotated image; the original frame from sequence `bicyclist` from EISATS is 640×480 and recorded at 10 bit per pixel

Demagnified image

Uniform brightness change

Blurred image

Feature Evaluation Test Procedure

Four changes: rotation, scaling, brightness changes, and blurring; select sequence of frames, feature detector and do

1. Read next frame I, which is a gray-level image
2. Detect keypoints p in I and their descriptors $\mathbf{d}(p)$ in I
3. Let N_k be the number of keypoints p in I
4. For given frame, generate four image sequences
   1. Rotate I around its center in steps of 1 degree
   2. Resize I in steps of 0.01, from 0.25 to 2 times the original size
   3. Add scalar to pixel values in increments of 1 from -127 to 127
   4. Apply Gaussian blur with increments of 2 for σ from 3 to 41
5. Feature detector again: keypoints p_t and descriptors $\mathbf{d}(p_t)$
6. $N_t =$ number of keypoints p_t for transformed image
7. Descriptors $\mathbf{d}(p)$ and $\mathbf{d}(p_t)$ to identify matches between features in I and I_t
8. Use RANSAC to remove inconsistent matches
9. $N_m =$ number of detected matches

Repeatability Measure

Repeatability $\mathcal{R}(I, I_t)$

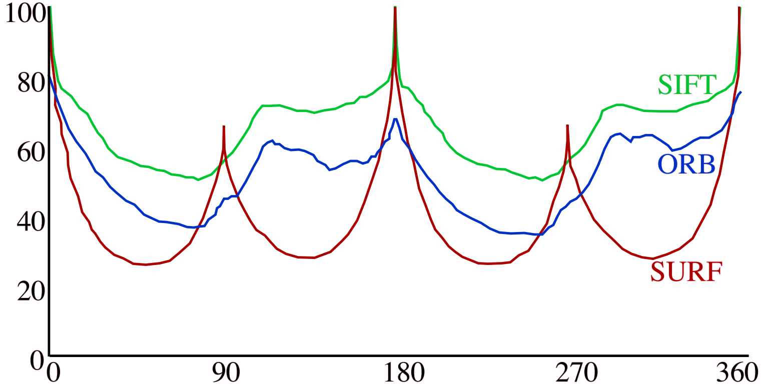
Ratio of number of detected matches to number of keypoints in the original image

$$\mathcal{R}(I, I_t) = \frac{N_m}{N_k}$$

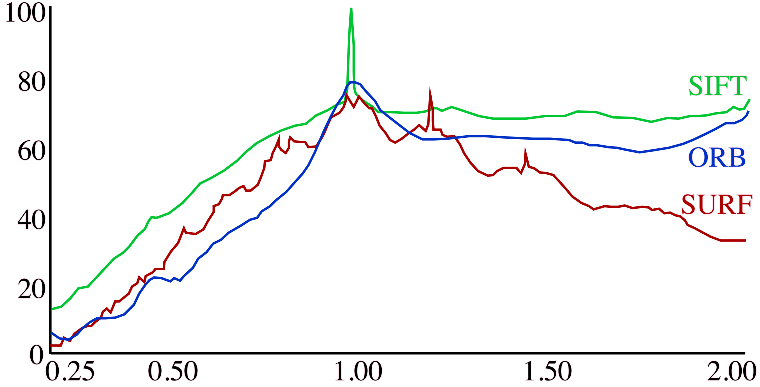
Report means for selected frames in test sequences

Use `OpenCV` default parameters for the studied feature detectors and a set of 90 randomly selected test frames

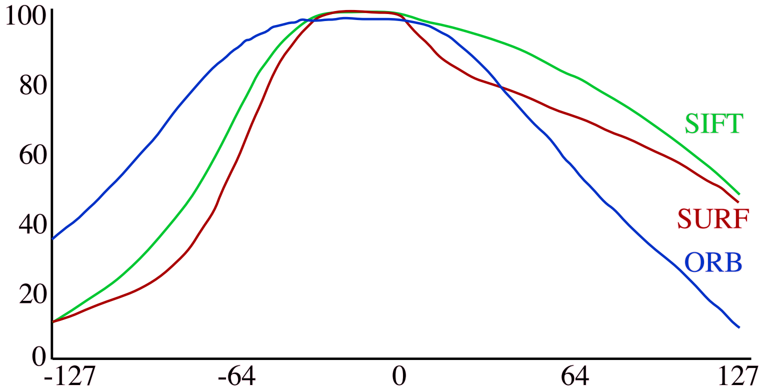
Repeatability Diagram For Rotation



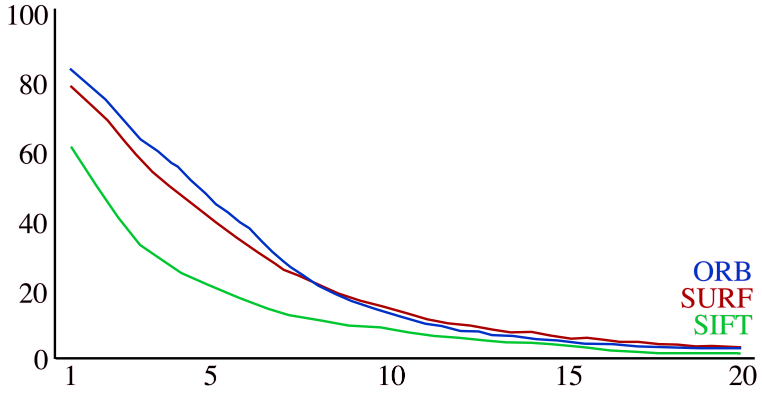
Repeatability Diagram For Scaling



Repeatability Diagram For Brightness Variation



Repeatability Diagram For Blurring



Discussion of the Experiments

Invariance has certainly its limits

If scaling, brightness variation, or blurring pass some limits then we cannot expect repeatability anymore

Rotation is a different case; here we could expect invariance close to the ideal case (of course, accepting that digital images do not rotate as continuous 2D functions in $\mathbb{R}^2$

Detector	Time per frame	Time per keypoint	Number N_k
SIFT	254.1	0.55	726
SURF	401.3	0.40	1,313
ORB	9.6	0.02	500

Mean values for 90 randomly selected input frames

Third column: numbers of keypoints for the frame used for generating the transformed images

Summary

SIFT is performing well (compared to SURF and ORB) for rotation, scaling, and brightness variation, but not for blurring

All results are far from the ideal case of invariance

If there is only a minor degree of brightness variation or blurring, then invariance can be assumed

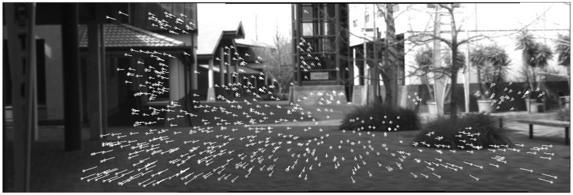
Rotation or scaling leads already to significant drops in repeatability for small angles of rotation, or minor scale changes

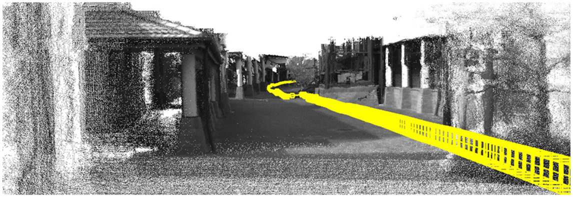
There was no significant run-time difference between SIFT and SURF

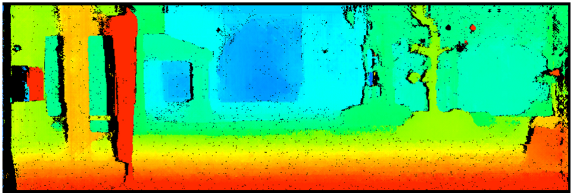
There was a very significant drop in computation time for ORB, which appears (judging from this comparison) as a fast and reasonably competitive feature detector

Agenda

Example







Caption

Top: Tracked feature points in a frame of a stereo video sequence recorded in a car

Middle: Tracked feature points are used for calculating the motion of the car; this allows to map 3D points provided by stereo vision into a uniform 3D world coordinate system

Bottom: Stereo matcher `iSGM` has been used for the shown example (example of a disparity map for the recorded sequence).

Example of an Application Scenario

A car, which is called the *ego-vehicle* because it is the reference vehicle where the considered system is working in, in distinction to "other" vehicles in a scene

This ego-vehicle is equipped with a stereo vision system and it drives through a street, providing reconstructed 3D clouds of points for each stereo frame at time t

After understanding the motion of the ego-vehicle, these 3D clouds of points can be mapped into a uniform 3D world coordinate system supporting 3D surface modeling of the road sides

For understanding the motion of the ego-vehicle, we track detected features from Frame t to Frame $t + 1$, being the input for a program calculating the *ego-motion* of the car

Tracking is a Sparse Correspondence Problem

Binocular stereo
Point or feature correspondence is calculated between images taken at the same time; the correspondence search is within an epipolar line; thus, stereo matching is a *1D correspondence problem*

Dense Motion (i.e. optic flow) Analysis
Point or feature correspondence is calculated between images taken at subsequent time slots; movements of pixels not constrained to be along one straight line; dense motion analysis is a *2D correspondence problem*

Feature Tracking
A sparse 2D correspondence problem

Tracking and Updating of Features

Theoretically, its solution could also be used for solving stereo or dense motion analysis

But there are different strategies for solving a dense or a sparse correspondence problem

In sparse correspondence search we cannot utilize a smoothness term, and need to focus more at first on achieving accuracy based on the data term only

We can use global consistency of tracked feature point patterns for stabilizing the result

Tracking with Understanding 3D Changes

Pair of 3D points $P_t = (X_t, Y_t, Z_t)$ and $P_{t+1} = (X_{t+1}, Y_{t+1}, Z_{t+1})$, projected at times t and $t+1$ into $p_t = (x_t, y_t, f)$ and $p_{t+1} = (x_{t+1}, y_{t+1}, f)$, respectively, when recording a video sequence

Z-ratio

$$\psi_Z = \frac{Z_{t+1}}{Z_t}$$

We can derive X- and Y-ratios

$$\begin{aligned}\psi_X &= \frac{X_{t+1}}{X_t} = \frac{Z_{t+1}}{Z_t} \cdot \frac{x_{t+1}}{x_t} = \psi_Z \frac{x_{t+1}}{x_t} \\ \psi_Y &= \frac{Y_{t+1}}{Y_t} = \frac{Z_{t+1}}{Z_t} \cdot \frac{y_{t+1}}{y_t} = \psi_Z \frac{y_{t+1}}{y_t}\end{aligned}$$

Update Equation

$$\begin{bmatrix} X_{t+1} \\ Y_{t+1} \\ Z_{t+1} \end{bmatrix} = \begin{bmatrix} \psi_X & 0 & 0 \\ 0 & \psi_Y & 0 \\ 0 & 0 & \psi_Z \end{bmatrix} \cdot \begin{bmatrix} X_t \\ Y_t \\ Z_t \end{bmatrix}$$

Knowing ψ_Z and ratios $\frac{x_{t+1}}{x_t}$ and $\frac{y_{t+1}}{y_t}$ allows us to update the position of point P_t into P_{t+1}

Assuming that P_t and P_{t+1} are positions of a 3D point P, from time t to time $t+1$, we only have to

1. decide on a technique to track points from t to $t+1$
2. estimate ψ_Z

Initial Position and Z-Ratios

If an initial position P_0 of a tracked point P is known then we may identify its 3D position at subsequent time slots

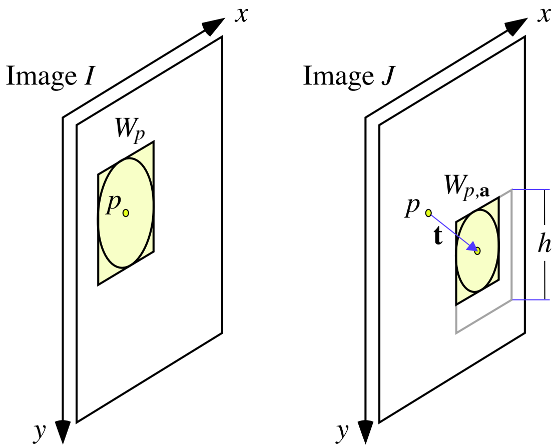
Without having an initial position, we only have a 3D direction P_t to P_{t+1}, but not its 3D position

Stereo vision is the general solution for estimating Z-values or (just) ratios ψ_Z

We can also estimate ψ_Z in a monocular sequence from scale-space results

Now: how to track points from t to $t+1$?

Lucas-Kanade Tracker



Template or base window W_p in base image I compared with a match window $W_{p,\mathbf{a}}$ in match image J

Sketch

Shown case: dissimilarity vector $\mathbf{a}$ is a translation $\mathbf{t}$ and a scaling of height h into a smaller height

Figure indicates that a disk of influence is contained in W_p

Pixel location p in J is the same as in I; it defines the start of the translation

Lucas-Kanade Tracker

Match *template* W_p, being a $(2k+1) \times (2k+1)$ window around keypoint $p = (x, y)$ in a base image I, with windows $W_{p,\mathbf{a}}$ in a match image J

Method should be general enough to allow for translation, scaling, rotation and so forth between base window W_p and match window $W_{p,\mathbf{a}}$ in J

Vector $\mathbf{a}$ parametrizes the transform from p into a new center pixel, and also the transformation of window W into a new shape

Newton-Raphson Iteration

Task: Calculate a zero of a smooth unary function $\phi(x)$, for $x \in [a, b]$, provided that we have $\phi(a)\phi(b) < 0$

Inputs are the two reals a and b

We also have a way to calculate $\phi(x)$ and the derivative $\phi'(x)$ (e.g. approximated by difference quotients), for any $x \in [a, b]$

Calculate a value $c \in [a, b]$ as an approximate zero of ϕ:

```
1: Let $c \in [a, b]$ be an initial guess for a zero.
2: **while** STOP CRITERION = false **do**
3:     Replace $c$ by $c - \frac{\phi(c)}{\phi'(c)}$
4: **end while**
```

Derivative $\phi'(c)$ is assumed to be non-zero; if ϕ has a derivative of constant sign in $[a, b]$ then there is just one zero in $[a, b]$

Comments

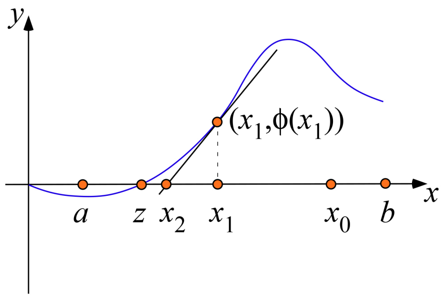
Initial value of c can be specified by (say) a small number of binary-search steps for reducing the run-time of the actual Newton-Raphson iteration

A small $\varepsilon > 0$ is used for specifying the STOP CRITERION "$|\phi(c)| \leq \varepsilon$"

Method converges in general only if c is "sufficiently close" to the zero z

If $\phi''(x)$ has a constant sign in $[a, b]$, then we have the following: if $\phi(b)$ has the same sign as $\phi''(x)$ then initial value $c = b$ gives convergence to z, otherwise chose initial value $c = a$

Figure



A smooth function $\phi(x)$ on an interval $[a, b]$ with $\phi(a)\phi(b) < 0$

Assume that we start with $c = x_1$

Tangent at $(x_1, \phi(x_1))$ intersects x-axis at x_2 and defined by

$$x_2 = x_1 - \frac{\phi(x_1)}{\phi'(x_1)}$$

Have $\phi'(x_1) \neq 0$. Now continue with $c = x_2$ and new tangent, etc.

Convergence and Valleys

For initial value x_1, sequence $x_2, x_3, \ldots$ converges to zero z

If start at $c = x_0$ then the algorithm would fail

Note that $\phi''(x)$ does not have a constant sign in $[a, b]$

We need to start in the "same valley" where z is located

We search for the zero in the direction of the (steepest) decent

If we do not start in the "same valley" then we cannot cross the "hill" in between

Following the Newton-Raphson Iteration.

Lucas-Kanade tracker uses approximate gradients which are robust against variations in intensities

For window matching, an error function E is defined based on an LSE optimization criterion

Translation

Simplest case: only a translation $\mathbf{t} = [t.x, t.y]^\top$ such that $J(x + t.x + i, y + t.y + j) \approx I(x + i, y + j)$, for all i, j, with $-k \leq i, j \leq k$, defining relative locations in template W_p

Simplifying notation: assume that $p = (x, y) = (0, 0)$, and we use W or $W_\mathbf{a}$ instead of W_p or $W_{p,\mathbf{a}}$, respectively

Case of translation-only: approximate a zero (i.e. a minimum) of the error function

$$E(\mathbf{t}) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} [J(t.x + i, t.y + j) - I(W(i, j))]^2$$

where $\mathbf{t} = [t.x, t, y]^\top$ and $W(i, j) = (i, j)$

Goal for General Warps

Tracker not just for translations but for general *warps* defined by an affine transform, with a vector **a** parametrizing the transform

Let $J(W_{\mathbf{a}}(q))$ be the value at that point $W_{\mathbf{a}}(q)$ in J which results from warping pixel location $q = (i, j)$, with $-k \leq i, j \leq k$, according to parameter vector **a**

Warping will not map a pixel location onto a pixel location, thus we also apply some kind of interpolation for defining $J(W_{\mathbf{a}}(q))$

Translation with $\mathbf{a} = [t.x, t.y]^\top$: for $q = (i, j)$ we have $W_{\mathbf{a}}(q) = (t.x, t.y) + q$ and $J(W_{\mathbf{a}}(q)) = J(t.x + i, t.y + j)$

General case: calculate *dissimilarity vector* **a** which minimizes error function

$$E(\mathbf{a}) = \sum_q [J(W_{\mathbf{a}}(q)) - I(W(q))]^2$$

Iterative Steepest-Ascent Algorithm

Assume: we are at a parameter vector $\mathbf{a} = [a_1, \ldots, a_n]^\top$

Similarly to mean-shift algorithm for image segmentation, calculate (as partial step) shift $m_\mathbf{a} = [m_1, \ldots, m_n]^\top$ which minimizes

$$E(\mathbf{a} + m_\mathbf{a}) = \sum_q [J(W_{\mathbf{a}+m_\mathbf{a}}(q)) - I(W(q))]^2$$

Solving this LSE optimization problem:

Consider Taylor expansion (analog to deriving the Horn-Schunck constraint) of $J(W_\mathbf{a}(q))$ with respect to dissimilarity vector **a** and a minor shift $m_\mathbf{a}$

$$J(W_{\mathbf{a}+m_\mathbf{a}}(q)) = J(W_\mathbf{a}(q)) + m_\mathbf{a}^\top \cdot \mathbf{grad}\, J \cdot \frac{\partial W_\mathbf{a}}{\partial \mathbf{a}} + e$$

Assume $e = 0$, thus linearity of values of image J in the neighborhood of pixel location $W_\mathbf{a}(q)$

LSE Optimization Problem

Second term on the right-hand side is a scalar: product of shift vector $m_{\mathbf{a}}$, derivative **grad** J of the outer function (i.e. the usual image gradient), and the derivative of the inner function

Window function W defines a point with x- and y-coordinates; derivative of W with respect to locations identified by **a**:

$$\frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}(q) = \begin{bmatrix} \frac{\partial W_{\mathbf{a}}(q).x}{\partial x} & \frac{\partial W_{\mathbf{a}}(q).x}{\partial y} \\ \frac{\partial W_{\mathbf{a}}(q).y}{\partial x} & \frac{\partial W_{\mathbf{a}}(q).y}{\partial y} \end{bmatrix}$$

This is the *Jacobian matrix* of the warp; minimization problem now:

$$\sum_q \left[J(W_{\mathbf{a}}(q)) + m_{\mathbf{a}}^\top \cdot \mathbf{grad}\, J \cdot \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}} \; - \; I(W(q)) \right]^2$$

Follow standard LSE optimization for calculating optimum shift $m_{\mathbf{a}}$

LSE Procedure

1. Calculate the derivative of this sum with respect to shift $m_{\mathbf{a}}$
2. Set this equal to zero
3. Obtain the equation (with 2×1 zero-vector $\mathbf{0}$)

$$2 \sum_q \left[\mathbf{grad}\ J\ \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}\right]^{\top} \left[J(W_{\mathbf{a}}(q)) + m_{\mathbf{a}}^{\top} \cdot \mathbf{grad}\ J \cdot \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}} \quad - \quad I(W(q))\right] = \mathbf{0}$$

2×2 Hessian matrix

$$H = \sum_q \left[\mathbf{grad}\ J\ \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}\right]^{\top} \left[\mathbf{grad}\ J\ \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}\right]$$

Solution defines optimum shift vector $m_{\mathbf{a}}$

$$m_{\mathbf{a}}^{\top} = H^{-1} \sum_q \left[\mathbf{grad}\ J\ \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}\right]^{\top} [I(W(q)) - J(W_{\mathbf{a}}(q))]$$

from given parameter vector $\mathbf{a}$ to updated vector $\mathbf{a} + m_{\mathbf{a}}$

Analogy to the Newton-Raphson Iteration

1. Start with an initial dissimilarity vector $\mathbf{a}$
2. New vectors $\mathbf{a} + m_{\mathbf{a}}$ are calculated in iterations
3. Follow the steepest ascent

Possible stop criteria

1. Error value or length of shift vector $m_{\mathbf{a}}$ is below a given $\varepsilon > 0$
2. A predefined maximum of iterations

Example: Translation Case

Only translation $\mathbf{a}$ with $W_{\mathbf{a}}(q) = [t.x + i, t_y + j]^\top$, for $q = (i, j)$

Jacobian matrix

$$\frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}(q, \mathbf{a}) = \begin{bmatrix} \frac{\partial W_{\mathbf{a}}(q).x}{\partial x} & \frac{\partial W_{\mathbf{a}}(q).x}{\partial y} \\ \frac{\partial W_{\mathbf{a}}(q).y}{\partial x} & \frac{\partial W_{\mathbf{a}}(q).y}{\partial y} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Hessian matrix (approximated by products of first-order derivatives)

$$H = \sum_q \left[\mathbf{grad}\ J\ \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}\right]^\top \left[\mathbf{grad}\ J\ \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}\right] = \sum_q \begin{bmatrix} \left(\frac{\partial J}{\partial x}\right)^2 & \frac{\partial J^2}{\partial x \partial y} \\ \frac{\partial J^2}{\partial x \partial y} & \left(\frac{\partial J}{\partial y}\right)^2 \end{bmatrix}$$

Steepest ascent

$$\mathbf{grad}\ J \cdot \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}} = \mathbf{grad}\ J$$

and

$$I(W(q)) - J(W_{\mathbf{a}}(q)) = I(W(q)) - J(q + \mathbf{a})$$

Altogether (for Translation)

$$
\begin{aligned}
m_{\mathbf{a}}^{\top} &= H^{-1} \sum_{q} \left[\mathbf{grad}\ J \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}} \right]^{\top} [I(W(q)) - J(W_{\mathbf{a}}(q))] \\
&= \left[\sum_{q} \left[\begin{array}{cc} \left(\frac{\partial J}{\partial x}\right)^2 & \frac{\partial J^2}{\partial x \partial y} \\ \frac{\partial J^2}{\partial x \partial y} & \left(\frac{\partial J}{\partial y}\right)^2 \end{array} \right] \right]^{-1} \sum_{q} [\mathbf{grad}\ J]^{\top} [I(W(q)) - J(q + \mathbf{a})]
\end{aligned}
$$

1. Approximate derivatives in image J around the current pixel locations in window W
2. This defines the Hessian and the gradient vector
3. Then a sum of differences for identifying the shift vector $m_{\mathbf{a}}$

Lucas-Kanade Algorithm

Given is an image I, its gradient image **grad** I, and a local template W (i.e. a window) containing (e.g.) the disk of influence of a keypoint

```
1: Let $\mathbf{a}$ be an initial guess for a dissimilarity vector
2: while STOP CRITERION = false do
3:     For the given vector $\mathbf{a}$, compute the optimum shift $m_{\mathbf{a}}$ as
       defined above
4:     Let $\mathbf{a} = \mathbf{a} + m_{\mathbf{a}}$
5: end while
```

Line 3

Line 3 in the algorithm requires calculations for all pixels q defined by template W; basically the main steps:

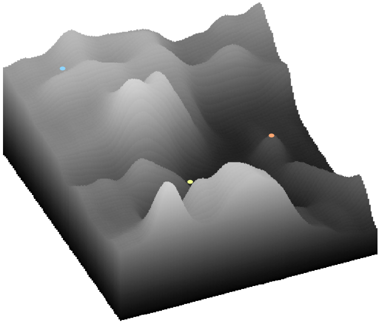
1. Warp W in I into $W_{\mathbf{a}}(q)$ in J
2. Calculate the Jacobian matrix and its product with **grad** J
3. Compute the Hessian matrix

The algorithm performs magnitudes faster than an exhaustive search algorithm for an optimized vector $\mathbf{a}$

Program for Lucas-Kanade algorithm available in `OpenCV`

Dents and Hills

Assume that error values are defined on the plane, and for different values of **a** they describe a "hilly terrain", with local minima, possibly a uniquely defined global minimum, local maxima, and possibly a uniquely defined global maximum



Blue point "cannot climb" to global maximum; red point is already at local maximum; yellow dot can iterate to global peak

Drift

There is also the possibility of a *drift*

The individual local calculation can be accurate, but the composition of several local moves may result in significant errors after some time, mainly due to the discrete nature of the data

Copyright Information

This slide show was prepared by Reinhard Klette
with kind permission from Springer Science+Business Media B.V.

The slide show can be used freely for presentations.
However, *all the material* is copyrighted.

R. Klette. Concise Computer Vision.
©Springer-Verlag, London, 2014.

In case of citation: just cite the book, that's fine.