Corner (Interest Point) Detection

COMP 4102A

Winter 2015

Gerhard Roth

Version 1

Motivation: Corners for Recognition

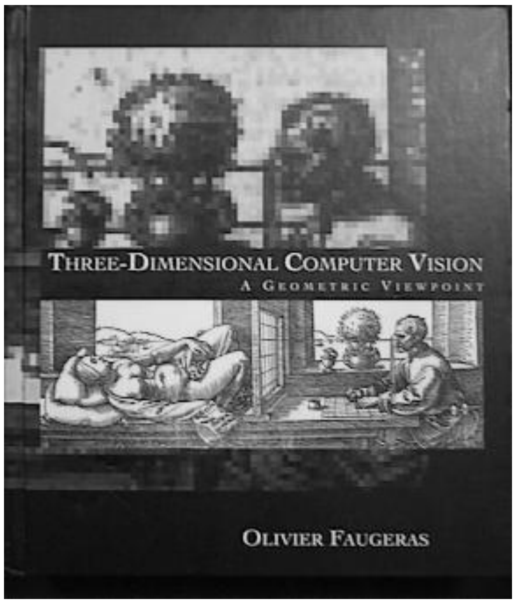
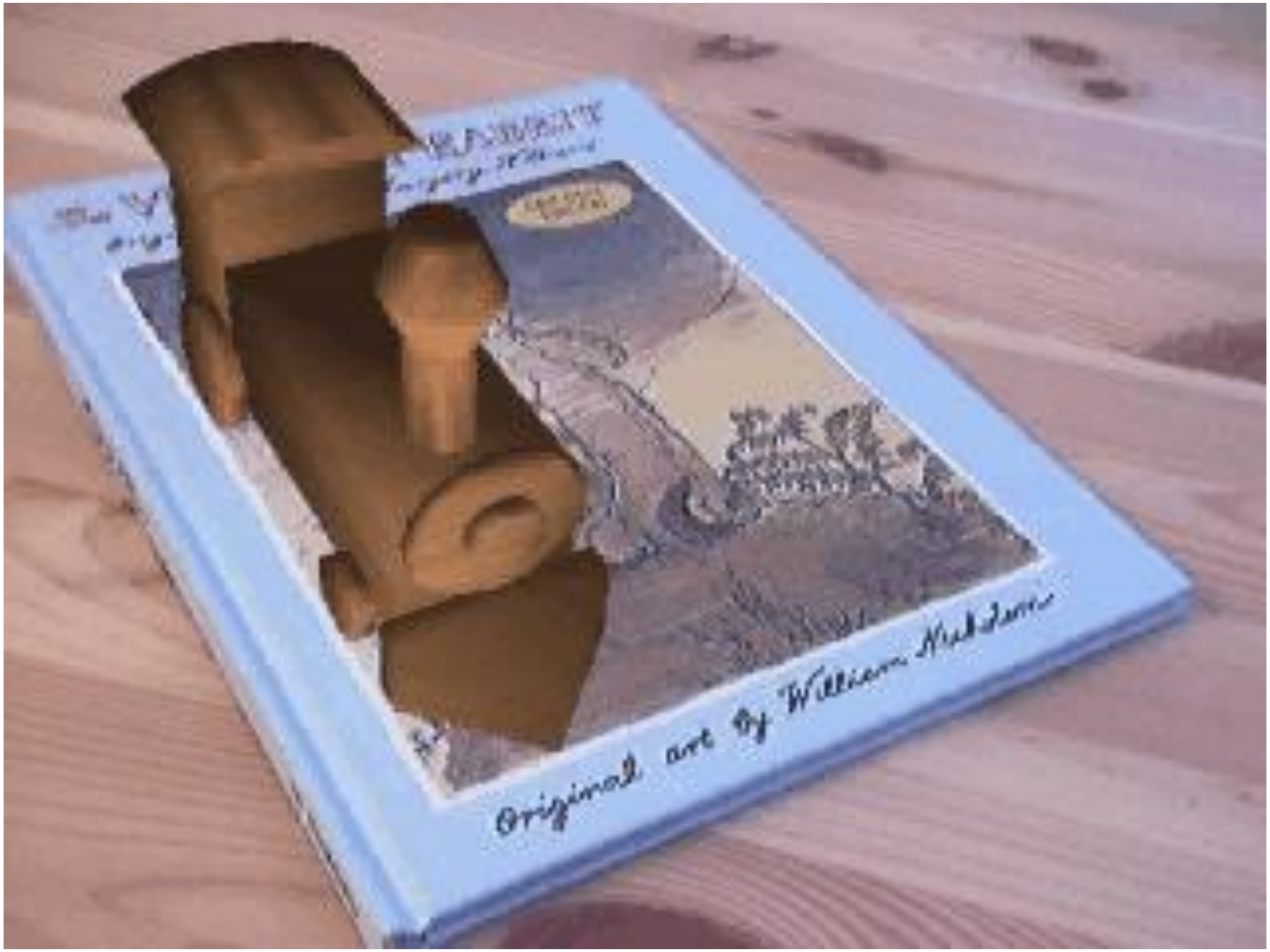


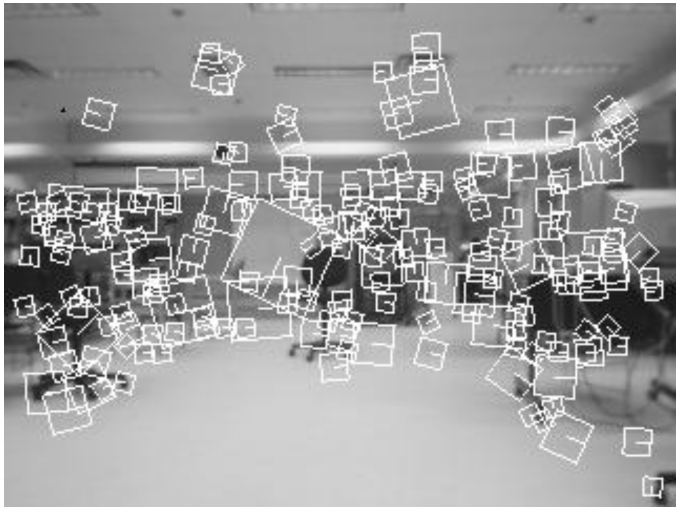
Image search: find the book in an image.

Motivation: Corners for Recognition

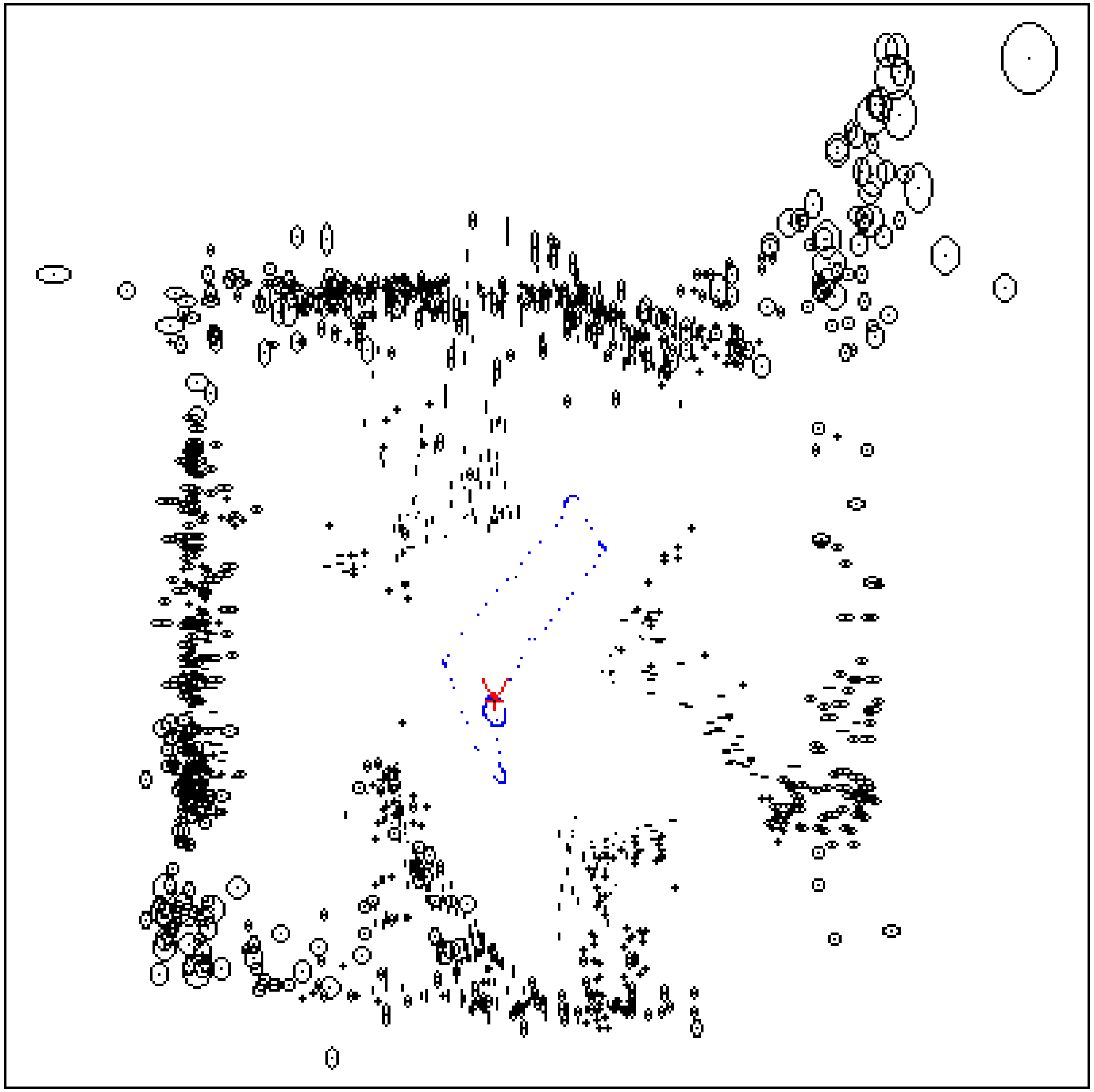
Corners for Augmented Reality



Motivation: Corners for Robotics



Motivation: 2D map built using corners



Motivation: Build a panorama

Input images

Output panorama 1

Motivation: Build a Panorama

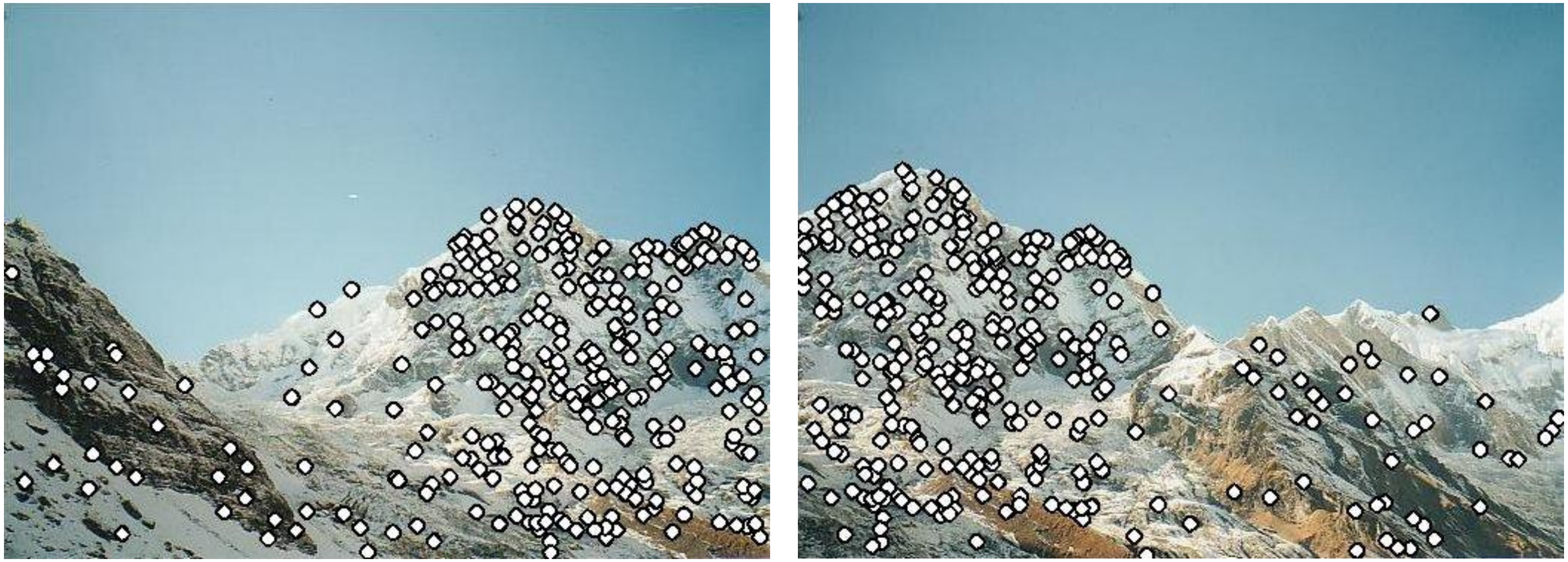
How do we build panorama?

We need to match (align) images

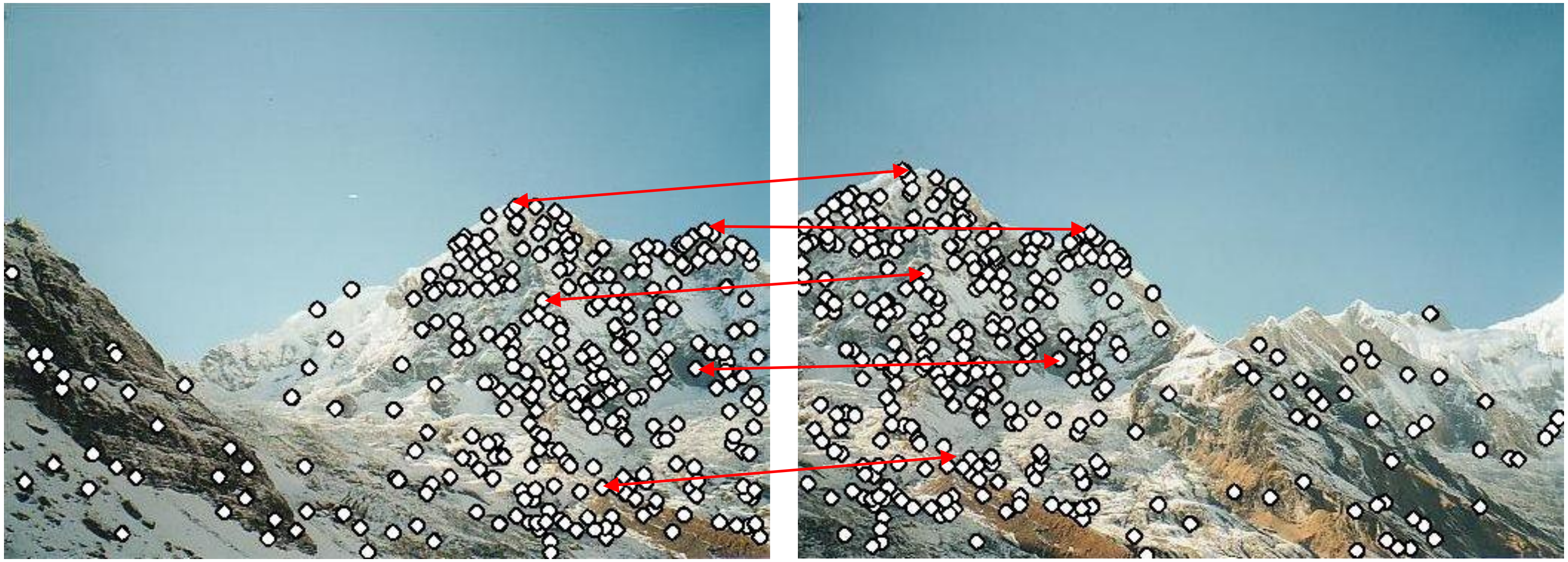
Matching with Corners

- Detect corner points in both images



Matching with Corners

- Detect corner points in both images
- Find corresponding corner pairs by comparing the corner descriptors



Matching with Corners

- Detect feature points in both images
- Find corresponding corner pairs by comparing the corner descriptors
- Use these pairs to align images

More motivation…

Corner points are used also for:

- Image alignment (homography, fundamental matrix)
- 3D reconstruction, Object recognition, Indexing and database retrieval, Robot navigation, … other

Corners define repeatable points for matching

Not just intersection of two lines (pure corner) but pixels which have a “corner like” structure

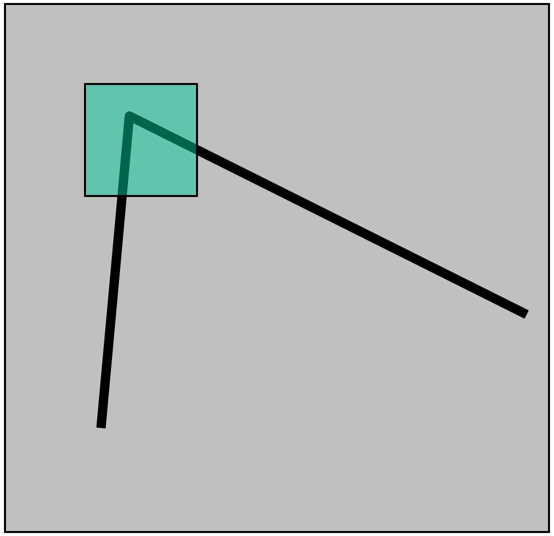
Corners sometimes called interest points because pixels that are “corner like” are interesting

Observe that in the region around a corner the gradient has two or more distinct values

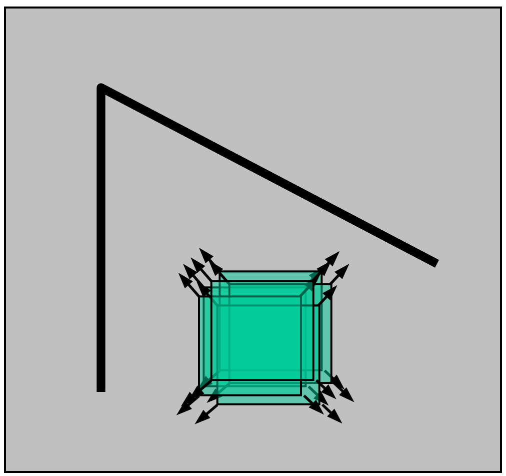
Corner Feature

Corners are image locations that have large intensity changes in more than one direction

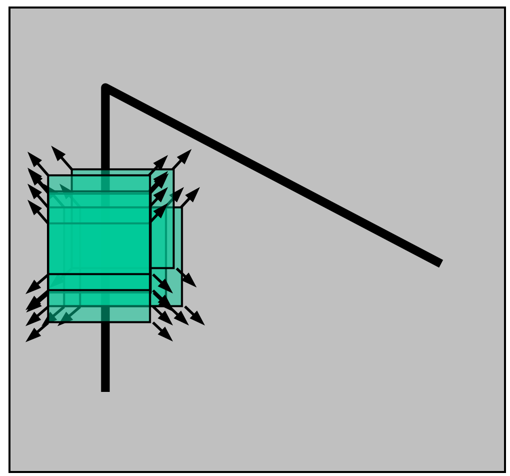
For a pixel which is a corner shifting a window centered on that pixel in *any direction* should give *a large change* in the average intensity in that window.



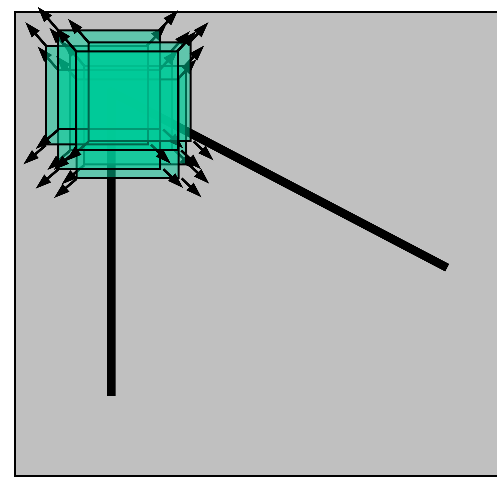
Harris Detector: Basic Idea



“flat” region:
no change in
all directions



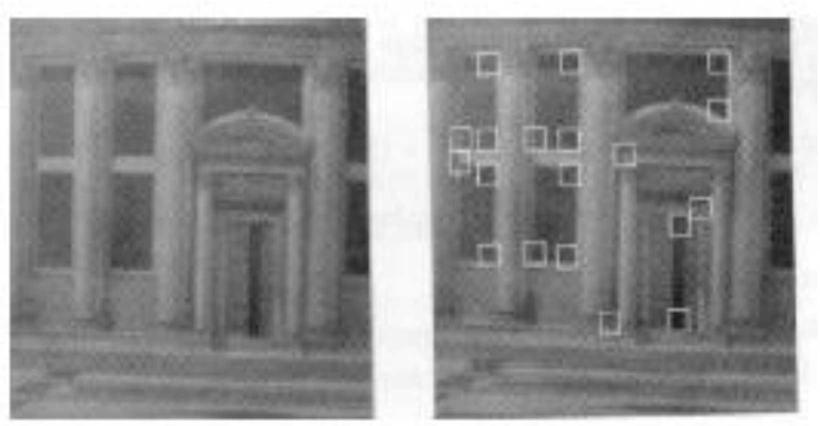
“edge”:
no change along
the edge direction

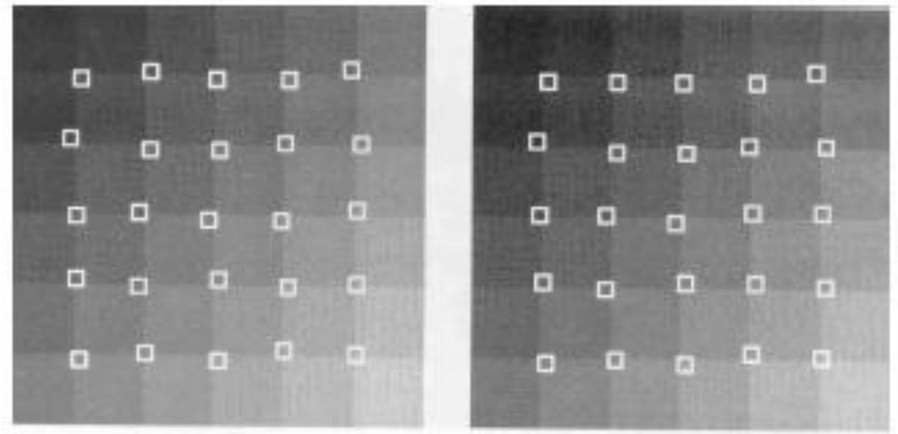


“corner”:
significant change
in all directions

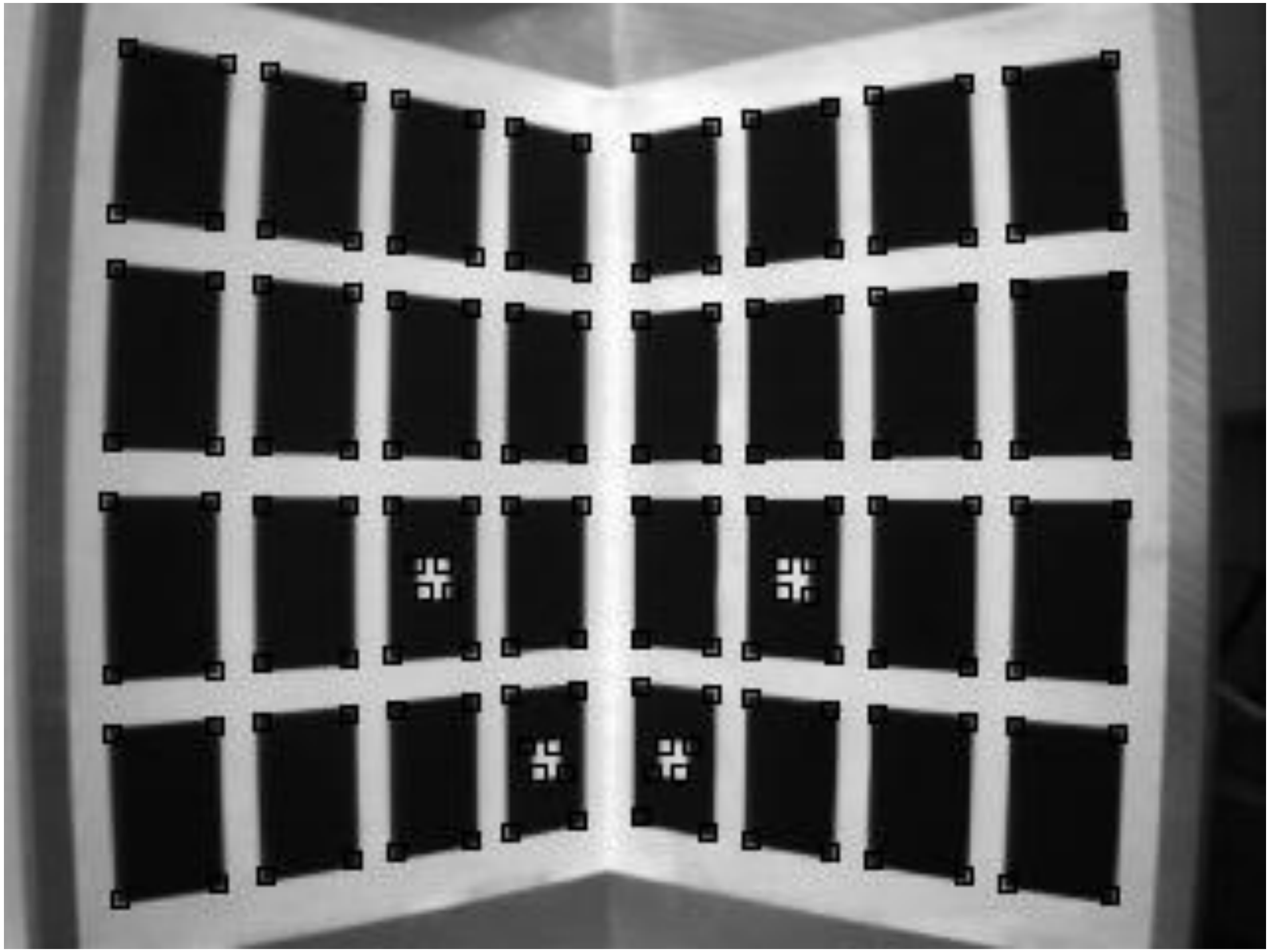
C.Harris, M.Stephens. “A Combined Corner and Edge Detector”. 1988

Examples of Corner Features

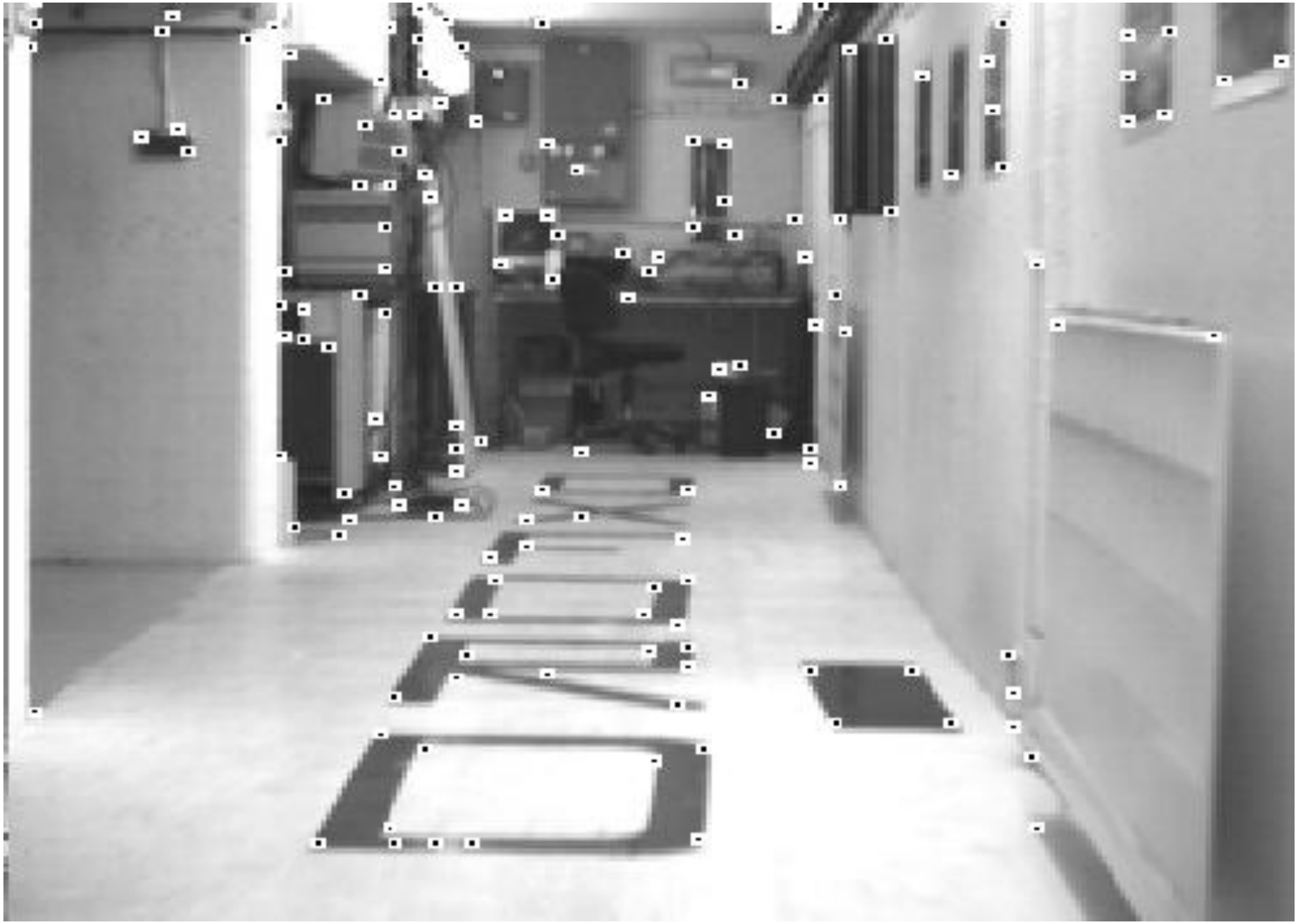




Corners in Calibration pattern



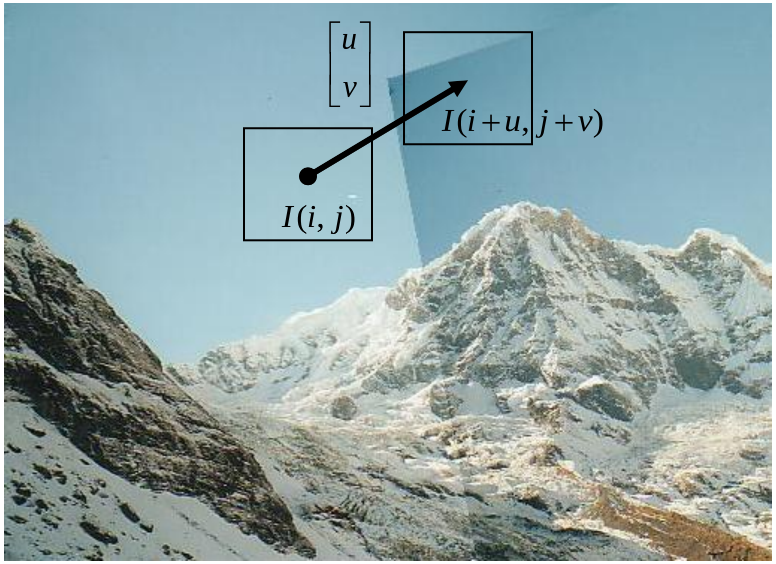
Corners in a Basement room



Change of Intensity

The intensity change at a given pixel in the direction (u,v) can be quantified by sum-of-squared-difference (SSD) of all pixels in a nbhd of that window, and the associated pixel shifted by (u,v).

$$D(u,v) = \sum_{i,j} \left(I(i+u, j+v) - I(i,j) \right)^2$$



Here i, j ranges over all the pixels in the nbhd.
The difference between the original pixel and shifted pixel in nbhd is summed.
Different D(u,v) function exists for every pixel in the nbhd.

Change Approximation

If *u* and *v* are small, by Taylor's theorem:

$$I(i+u, j+v) \approx I(i, j) + I_{x(i,j)} u + I_{y(i,j)} v$$

where $I_{x(i,j)} = \frac{\partial I}{\partial x}$ *and* $I_{y(i,j)} = \frac{\partial I}{\partial y}$

Therefore for any *i*, *j* and *u*, *v*

$$\begin{aligned}
\left(I(i+u, j+v) - I(i, j)\right)^2 &= \left(I(i, j) + I_{x(i,j)} u + I_{y(i,j)} v - I(i, j)\right)^2 \\
&= \left(I_{x(i,j)} u + I_{y(i,j)} v\right)^2 \\
&= I_{x(i,j)}^2 u^2 + 2 I_{x(i,j)} I_{y(i,j)} uv + I_{y(i,j)}^2 v^2 \\
&= \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_{x(i,j)}^2 & I_{x(i,j)} I_{y(i,j)} \\ I_{x(i,j)} I_{y(i,j)} & I_{y(i,j)}^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}
\end{aligned}$$

Gradient Variation Matrix

$$D(u,v) = \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

This function is a rotated ellipse. Ellipse *D(u,v)* = const

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

Matrix *C* characterizes how intensity changes in a certain direction. Each entry is computed by summing the appropriate values over every pixel in the neighbourhood around the given pixel

Eigenvalue Analysis – simple case

First, consider case of a corner/edge which is aligned with the x and y axis so we have:

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means dominant gradient directions align with x or y axis

If either λ is close to 0, then this is **not** a corner, so look for locations where both are large.

The bigger the smallest λ the more "corner like" is that pixel in the image

Slide credit: David Jacobs

Eigenvalue Analysis – simple case

$$D(u,v) = \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$D(u,v) = \lambda_1 u^2 + \lambda_2 v^2$$

Here λ_1 is the rate of change in direction of u

λ_2 is the rate of change in direction of v

If both λ are small, we have a constant region,

If only one λ is large have an edge,

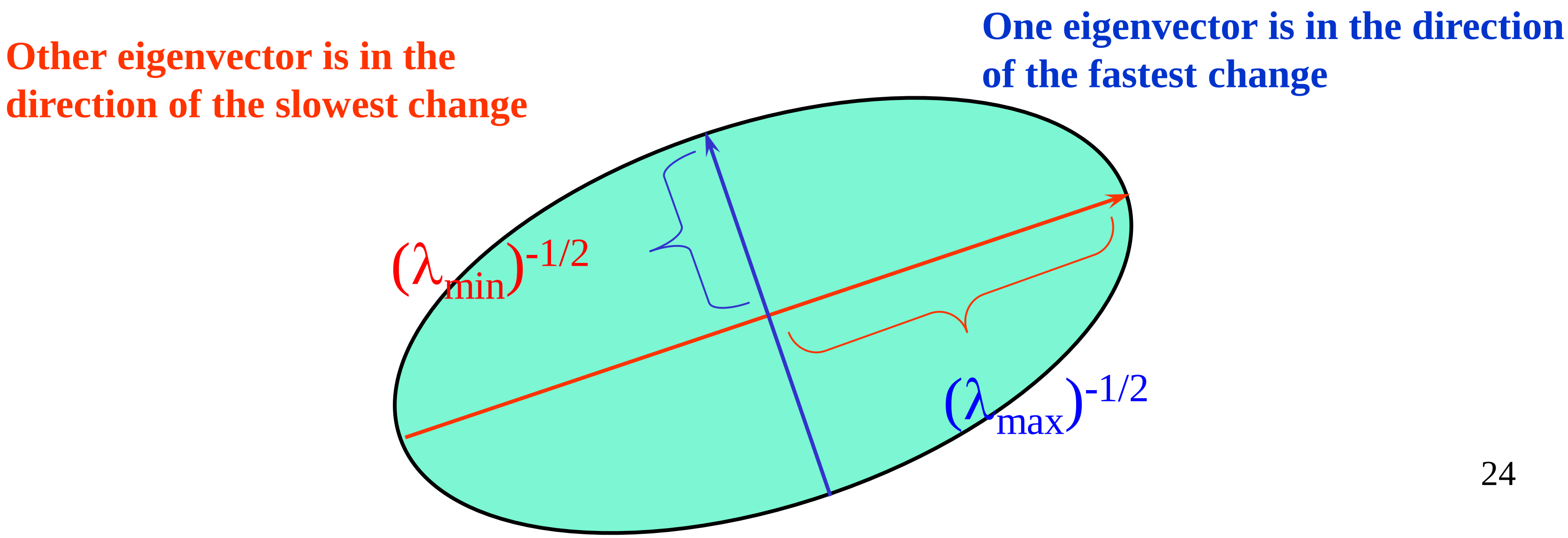
If both λ large is a corner (smallest λ is large)

General Case - Diagonalization

It can be shown that since C is symmetric it can be diagonalized, which means finding matrix Q to rotate and rewrite C as:

$$C = Q^T \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} Q$$

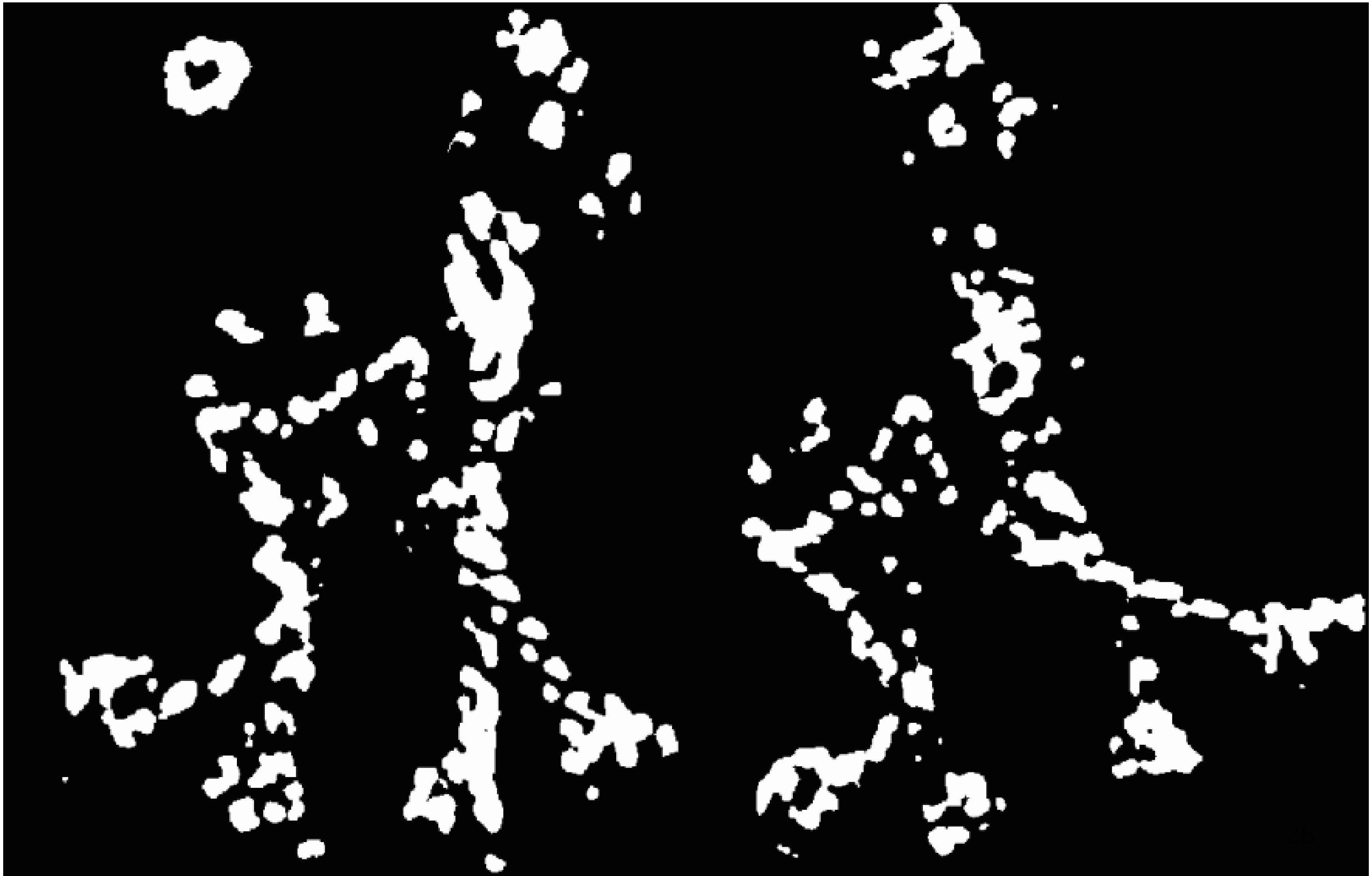
So every C is simply a rotated version of the simple case:



Harris Detector

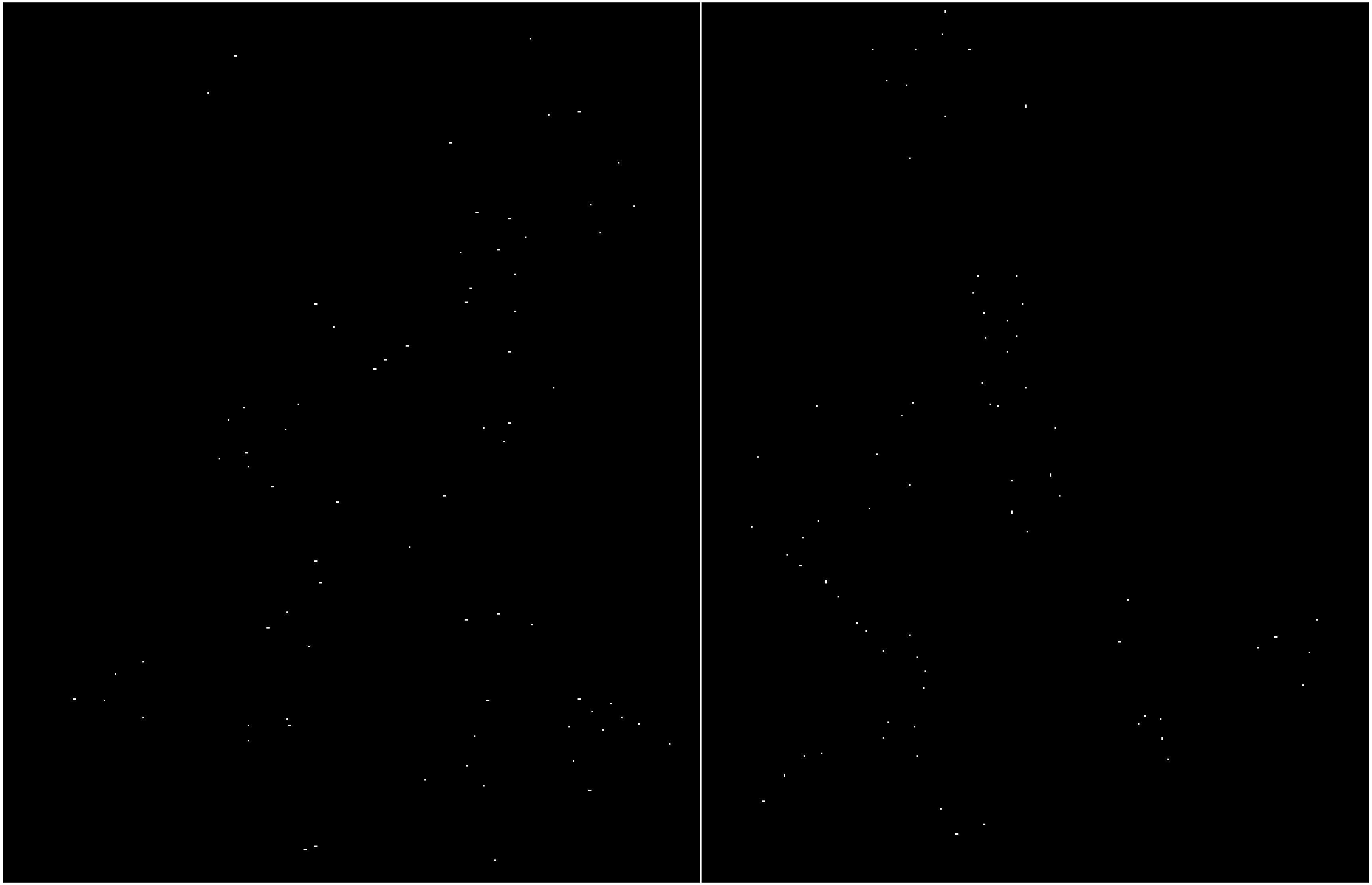
Harris Detector

Find points where smallest eigenvalue is >threshold



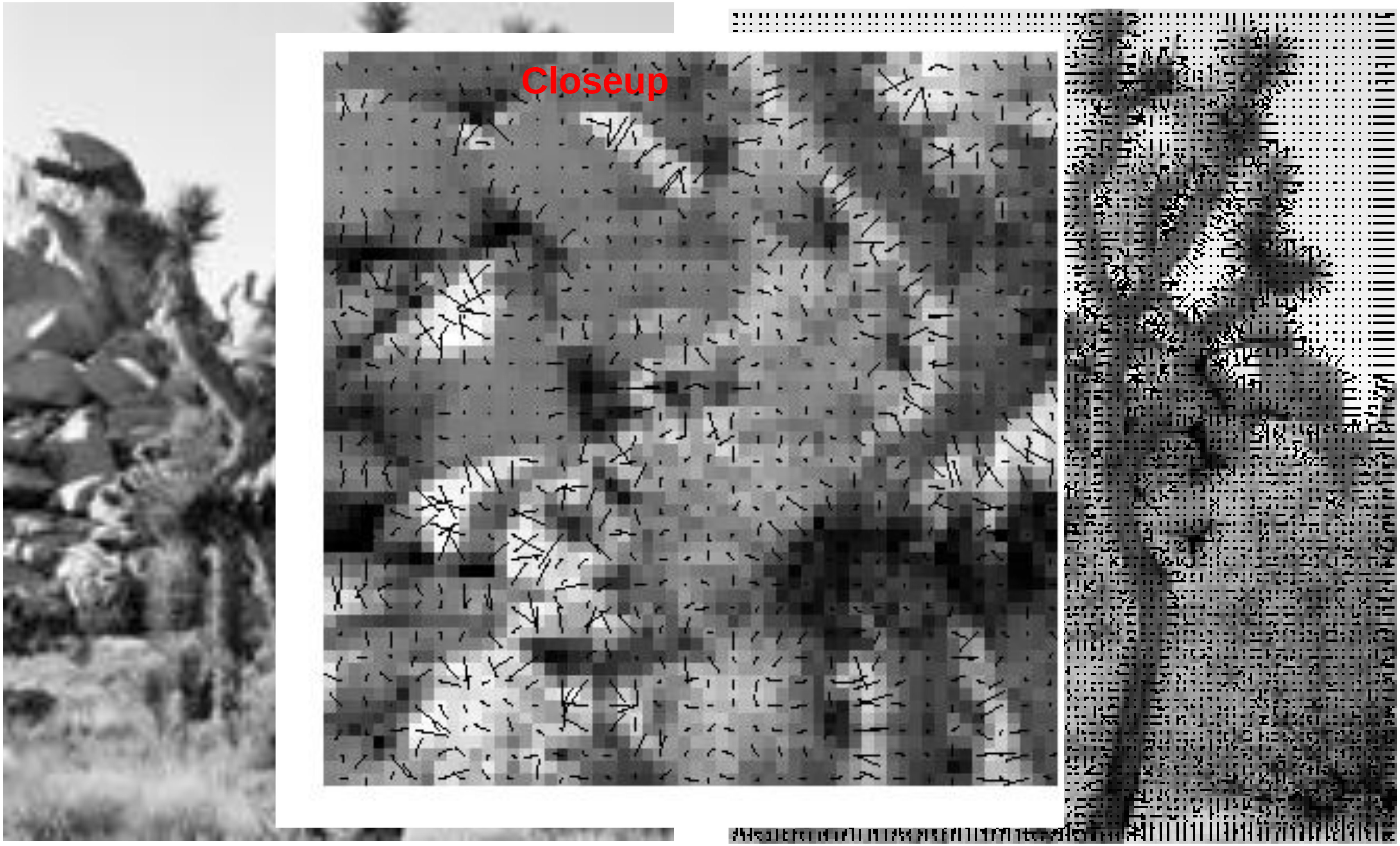
Harris Detector – non maxima superssion

Take only the points of local maxima of the smallest eigenvalue



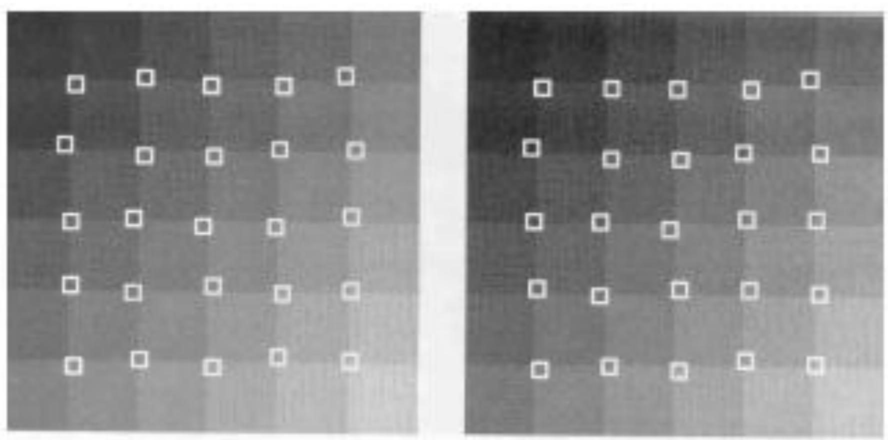
Harris Detector

Gradient Orientation



Corner Detection Summary

- if this is a region of constant intensity, both eigenvalues will be very small.
- if it contains an edge, there will be one large and one small eigenvalue (the eigenvector associated with the large eigenvalue will be parallel to the image gradient).
- if it contains edges at two or more orientations (i.e., a corner), there will be two large eigenvalues (the eigenvectors will be parallel to the image gradients).
- Eigenvectors encode edge directions, eigenvalues edge strength



Corner Detection Algorithm

Algorithm

Input: image f, threshold t for λ_2, size of Q

(1) Compute the gradient over the entire image f

(2) For each image point p:

(2.1) form the matrix C over the neighborhood Q of p
(2.2) compute λ_2, the smaller eigenvalue of C
(2.3) if $\lambda_2 > t$, save the coordinates of p in a list L

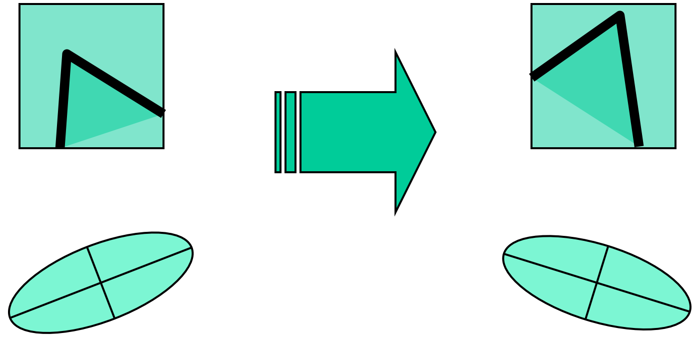
(3) Sort the list in decreasing order of λ_2

(4) Scanning the sorted list top to bottom: delete all the points that appear in the list that are in the same neighborhood Q with p

Step (3) and (4) is a type of non-maxima suppression (can be done in other ways)

Harris Detector Rotation Invariance

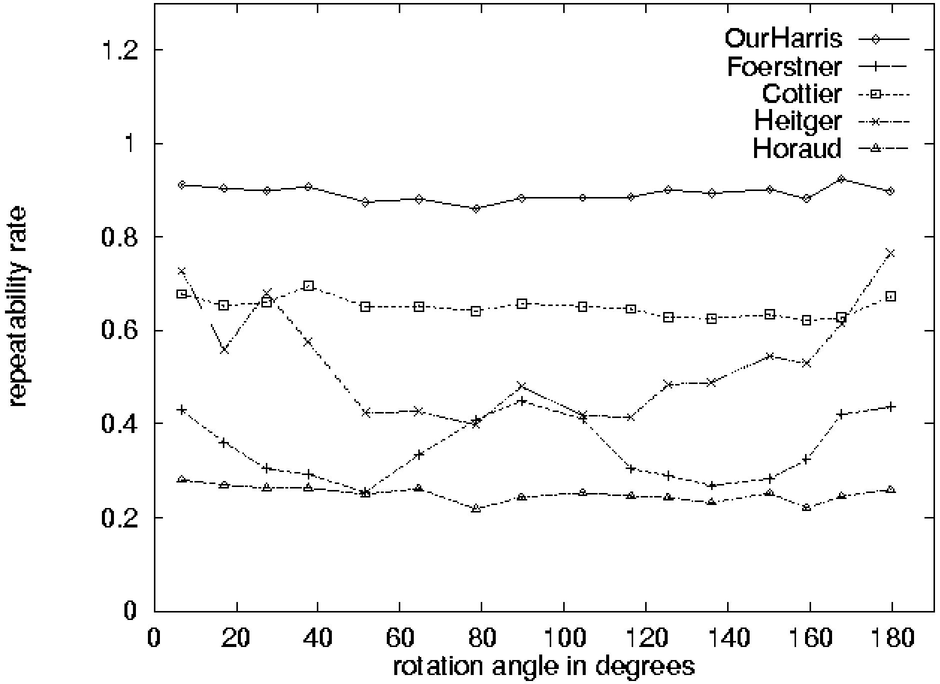
- Detection is invariant to rotation in the image plane (why?)
- These are different than rotations out of the camera plane!



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

Harris Detector Rotation Invariance

- Repeatability with image plane rotation



Comparing corners – Corner descriptor

- To match corners extract a description of the corner (this is also called a corner descriptor)
  - Need this to compare two corners in different images
  - Use the set of pixels in small nbhd around the corner and compute a high dimensional vector from these pixels
  - Up to you what you compute, but it should be invariant!
- Can simply use the pixels in a neighbourhood around each corner as the descriptor
  - To compare two descriptors take the sum of squares difference of pixels in a small window around each corner
- This is a very easy but is not invariant
  - There are better corner descriptors than using raw pixels
  - Want invariance for the corner detection process and for the descriptor associated with each corner

Invariance of Corner Detector

- Invariance is desirable but not easy to get
- Both corner detection process and descriptor must be invariant (these are different things!)
  - Often have some type of invariance (but not every type)
  - Harris corners detection is invariant to rotations and translations in the camera plane
  - The simplest descriptor consisting of the actual pixels in the pixel neighborhood have no rotation invariance
- Adding invariance due to lighting increase/decrease is very easy
  - Just take the average pixel value in neighborhood , and subtract it before comparing the two neighborhood of a descriptor

Invariance to scale and orientation

- These are critical for matching tasks
  - Scale is distance from object, orientation is viewing angle
- Descriptors invariant to scale have been created (called SIFT or SURF)
- Work by creating a scale space and finding the natural scale for the feature
  - Done by smoothing with Gaussian of different size and tracking features across scale
- Implies we can match SIFT/SURF descriptors at different scales and orientations

Blur and Lighting Change

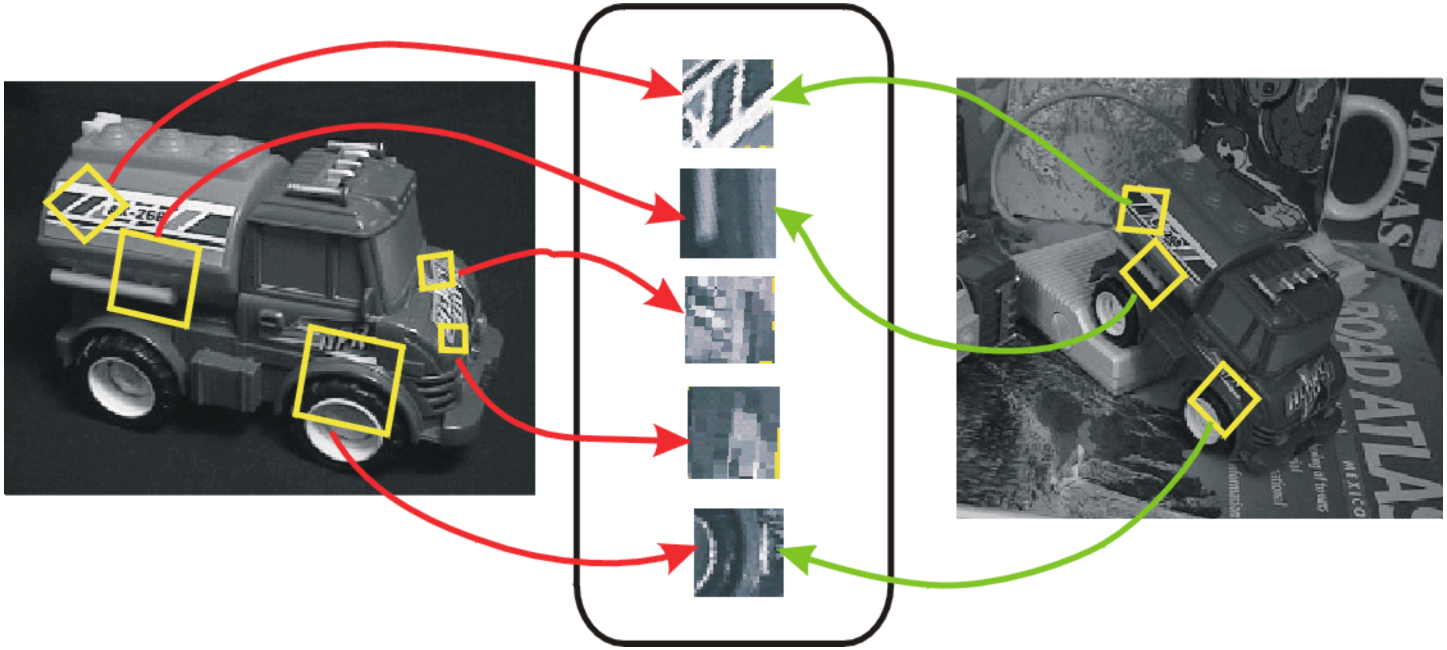
 

Orientation and Zoom/Rotation

SIFT/SURF Features – in OpenCV

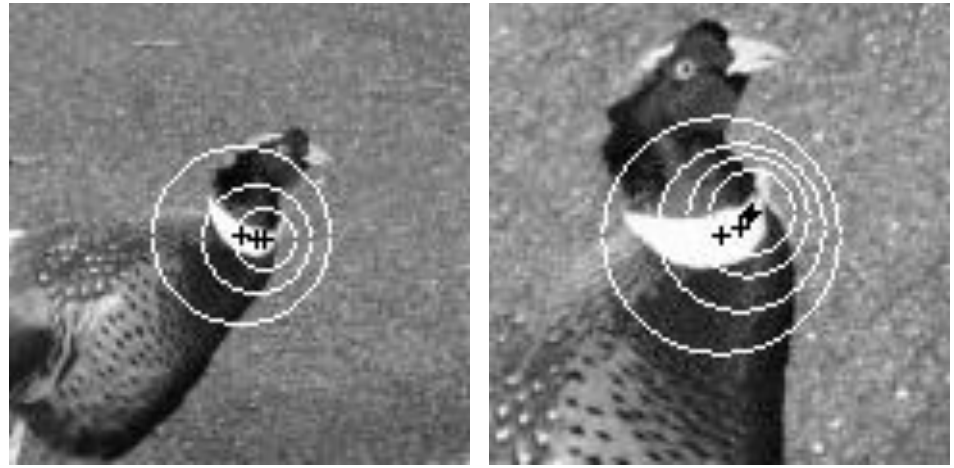
- Bot the feature detection and the feature descriptor are invariant to scale
- Means the pixels used to compute the feature descriptor change with the feature scale



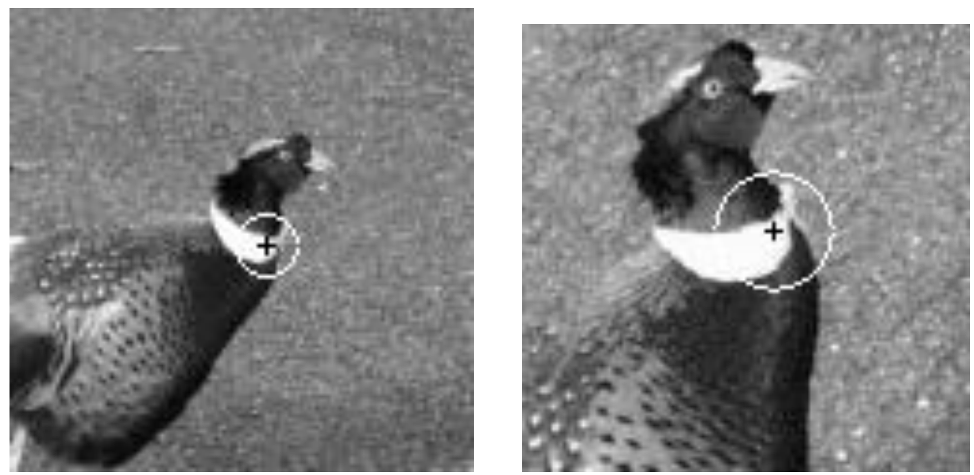
•Box around the feature changes scale appropriately

Scale Invariance – Find natural scale!

Bad Scale choice
means wrong
descriptor



Good scale choice
Means correct
Descriptor



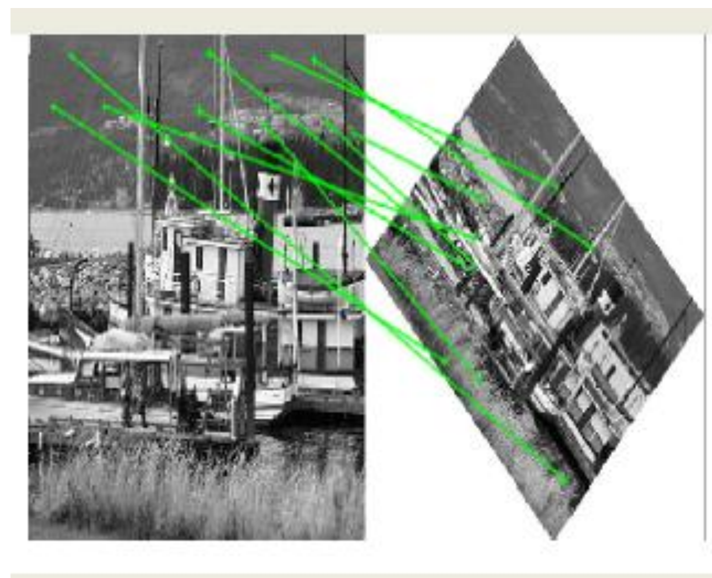
Harris relative to SIFT/SURF features

- For each feature Harris detector returns
  - Pixel location, corner strength and corner orientation
  - Size of nbhd used for the feature descriptor is not specified
- SIFT/SURF are scale invariant so they also
  - Include a scale (size of the nbhd window around the feature)
  - Compute a descriptor of feature from pixels in this nbhd
  - In other words, the nbhd changes to cover the same pixels as we change the distance of the camera to the feature
- So SURF/SIFT descriptor is invariant to scale
  - As we change camera distance still have the same corner descriptor because pixels in the nbhd change appropriately
  - This is scale invariance, not true for Harris corners

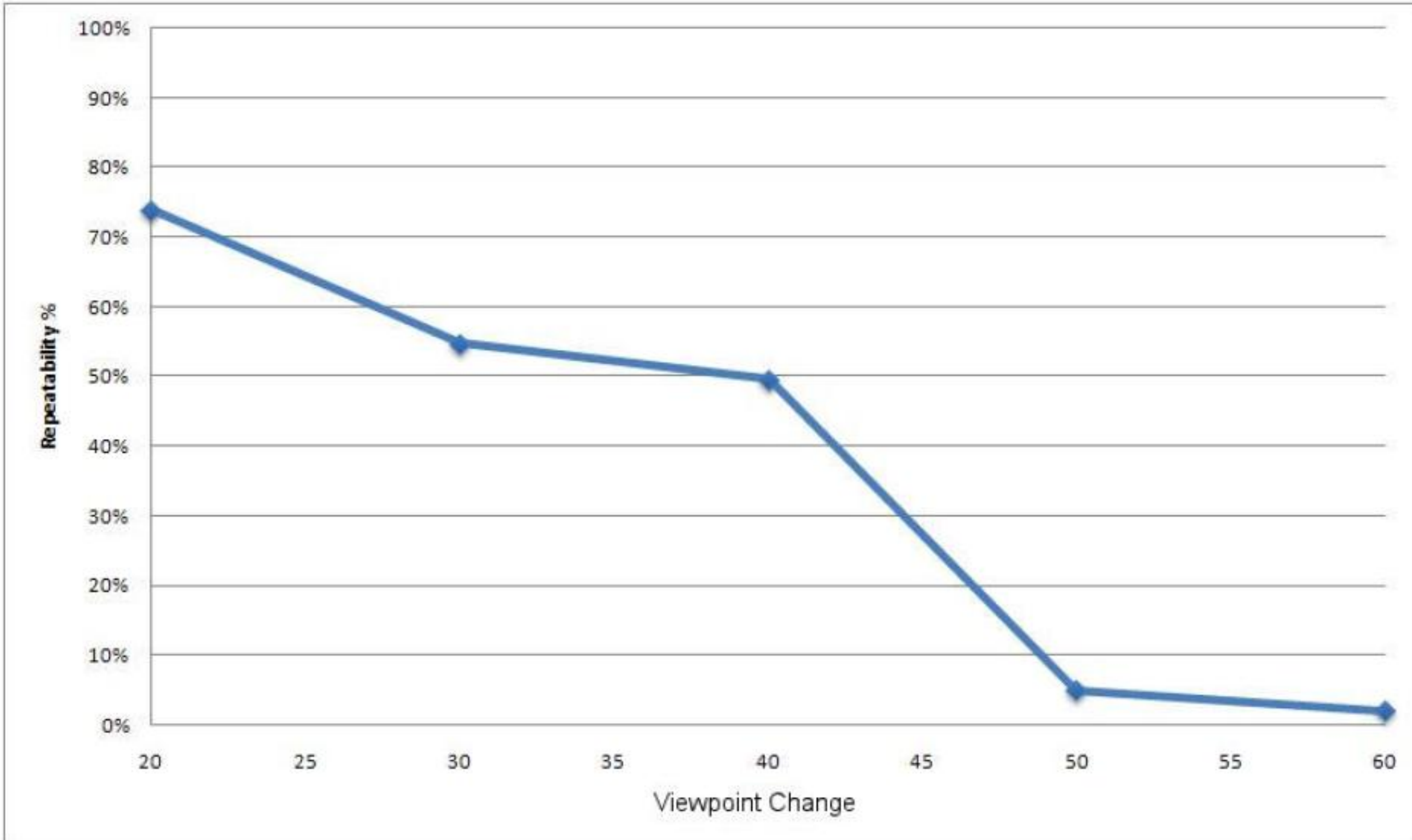
Small motions/large motions

- Consider two images
  - Extract corners (SIFT/SURF or Harris) and then match using some feature descriptor
- Harris features work only for some motions (rotation in camera plane, translation)
  - Descriptors usually pixels in a small nbhd around the corner
- SIFT/SURF features work for larger motions, and for different types of motions
  - Can handle blur, lighting, compression, all motion in the camera plane, and some motions out of the camera plane
  - But they are slower to compute and the matching of the descriptors associated with each feature is also slower

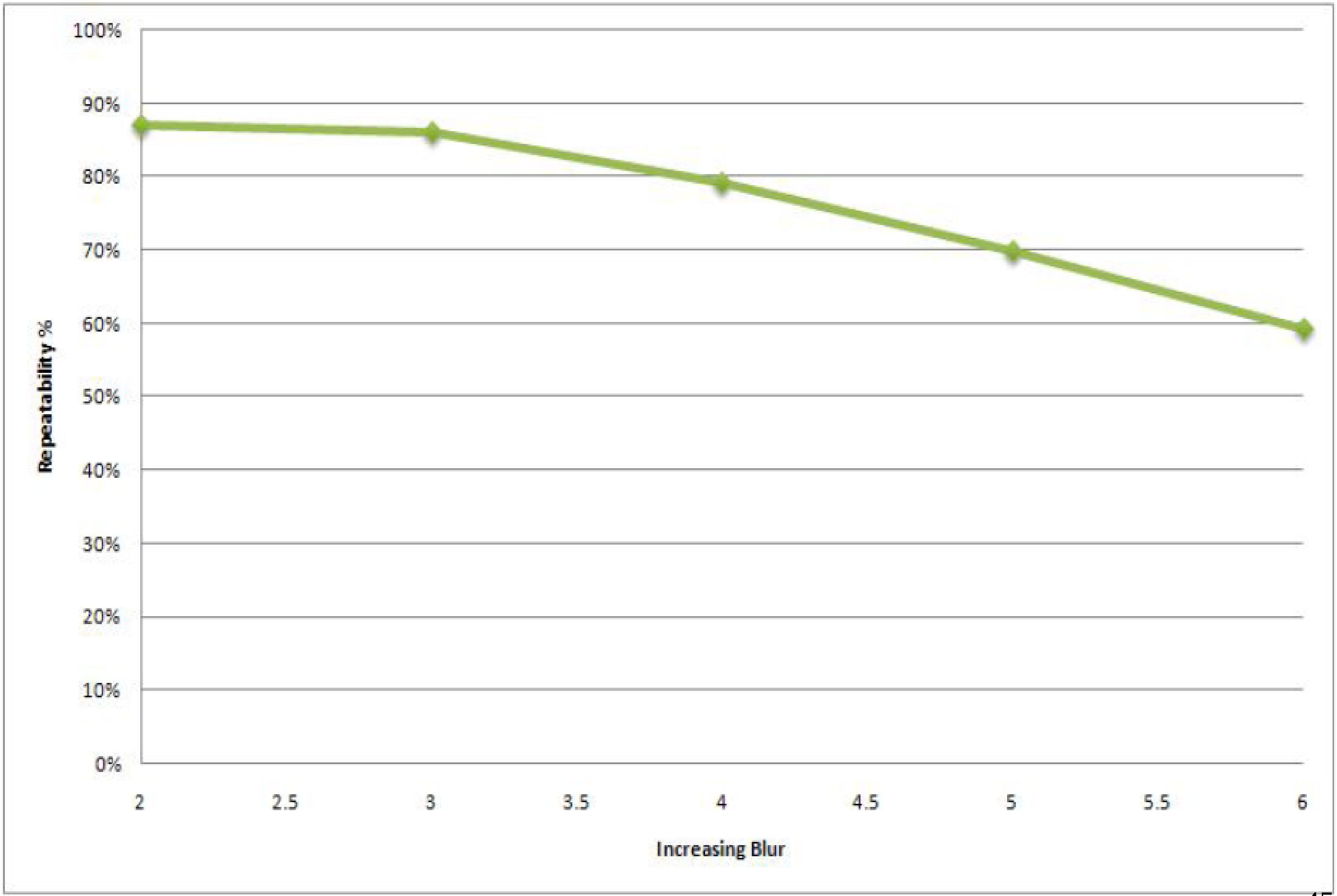
Successful SIFT/SURF matching



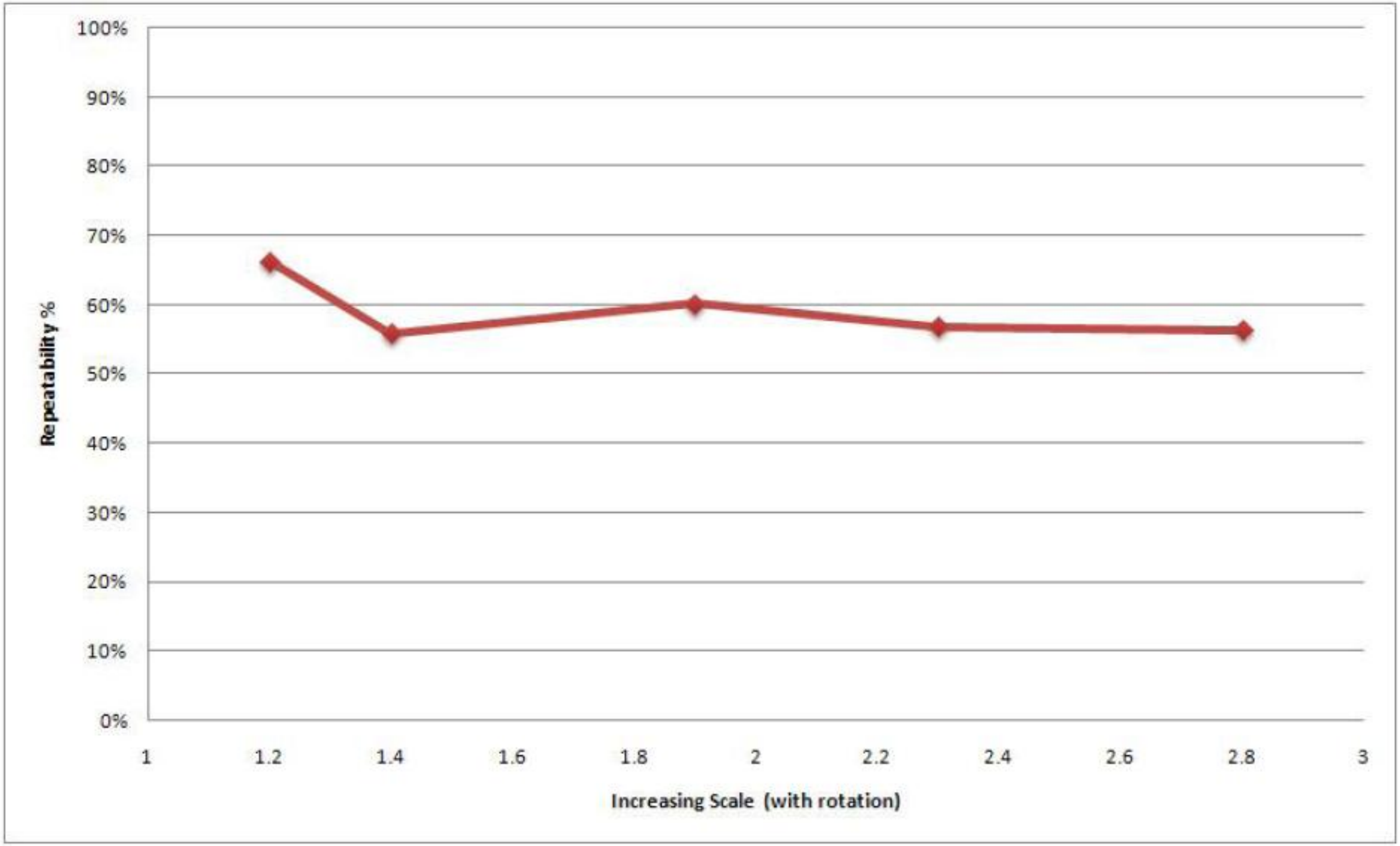
Viewpoint Change - SURF



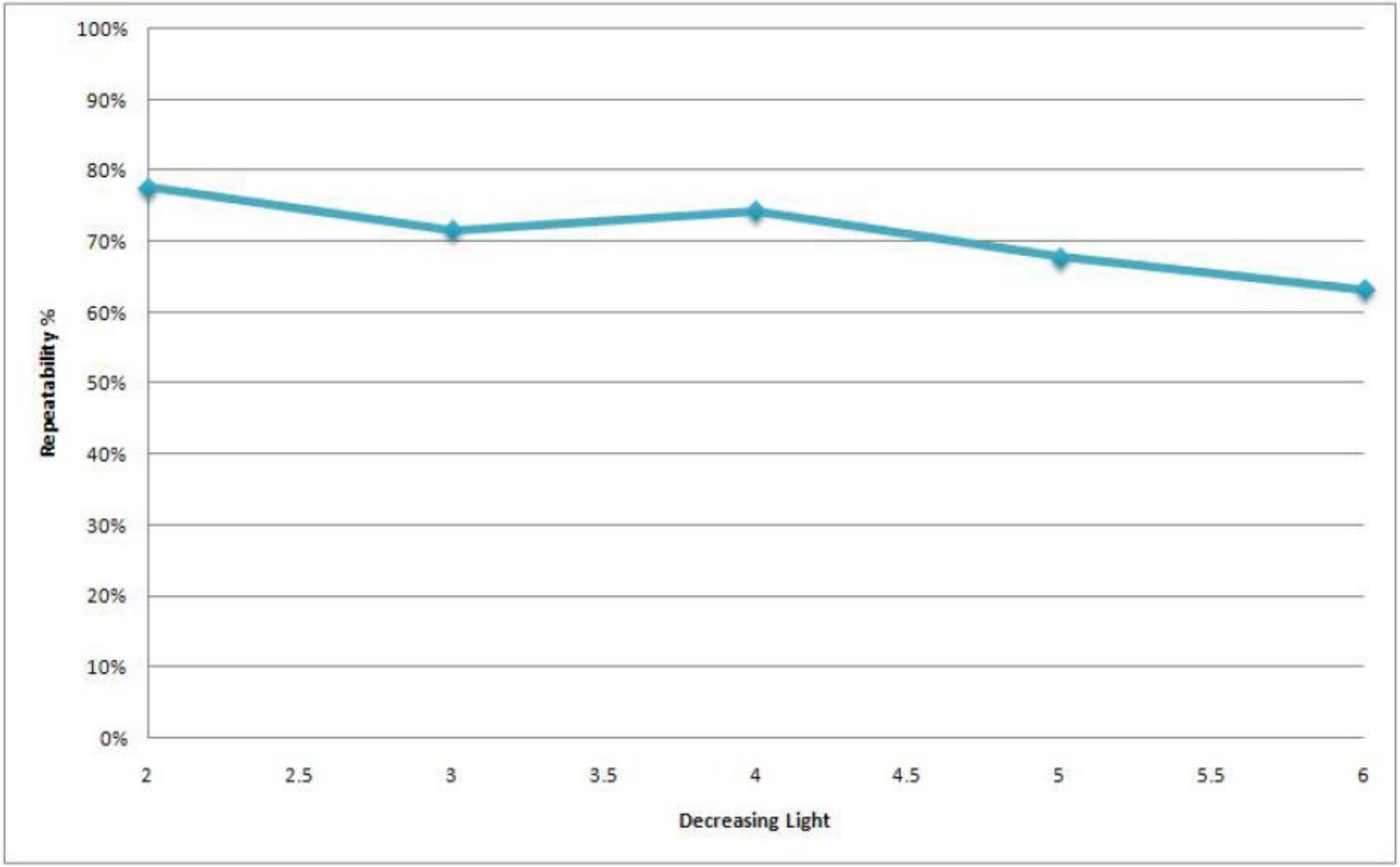
Increasing Blur - SURF



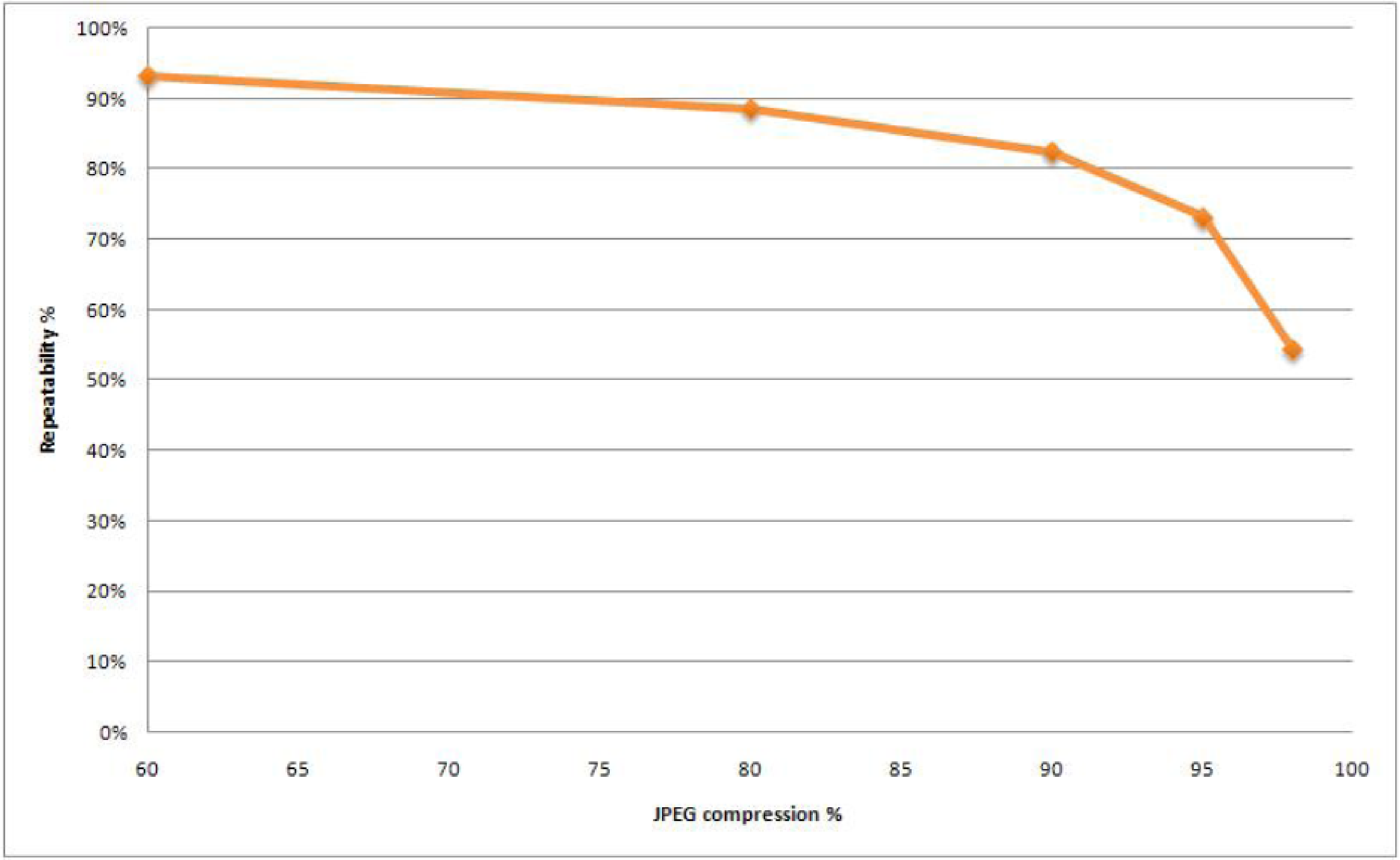
Scale/Rotate in Camera Plane - SURF



Decreasing Brightness - SURF



JPEG Compression - SURF



Harris Corners

- Are the simplest corners to find which
  - Can be extracted in real-time and are they are invariant to translation and rotation in image plane
  - They are not invariant to scale, only good for small motion
- Finding corners is only half the work
  - To match you need a corner descriptor which is a high dimensional vector derived from the pixels in a window centered at that corner pixel (what size is the window?)
- Both the corner location and the corner descriptor should be as invariant as possible
  - Invariant to scale, orientation, lighting, etc.
    - Should find same corners and very similar descriptor
  - Scale invariant corners now exist and are common
    - David Lowe (UBC) created SIFT corners, SURF is faster SIFT