Image Features

KOM4520 Fundamentals of Robotic Vision

Most of the contents are adopted from
Robotics, Vision and Control: Fundamental Algorithms in MATLAB, Peter Corke, 2nd ed., Springer Verlag. 2017
R. C. Gonzalez, R. E. Woods, *Digital Image Processing*, 2nd edition, Prentice Hall, 2008
Digital Image Processing: An Algorithmic Introduction using Java, W Burger, Mark J. Burge, Springer Verlag, 2008

Outline

- Image Processing vs Image Feature Extraction
- Region Features
  - Segmentation
- Line Features
  - Hough Transform
- Point Features
  - Harris Corner Detector
  - SURF

Image Processing vs Image Feature Extraction

- We learned that images are simply large arrays of pixel values but for robotic applications images have too much data and not enough information.
- We need measurements obtained from the image and which we call image features. Features are the essence of the scene and the raw material that/ we need for robot control.

- The image processing operations from the last chapter operated on one or more input images and returned another image.
- In contrast feature extraction operates on an image and returns one or more image features. Image feature extraction is a necessary first step in using image data to control a robot. It is an information concentration step.

- we will discuss several classes of feature:

  regions, lines and interest points

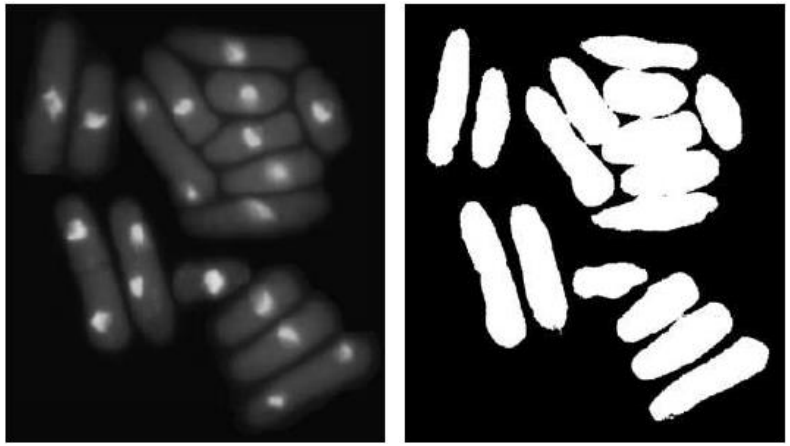
Region Features

- Image **segmentation** refers to the process of partitioning a digital image into multiple segments.
- The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.
- Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics.
- The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image (see edge detection). Each of the pixels in a region are similar with respect to some characteristic or computed property, such as color, intensity, or texture

Some of the practical applications of image segmentation are:

- Medical Imaging
- Measure tissue volumes
- Computer-guided surgery
- Locate objects in satellite images (roads, forests, etc.)
- Face recognition
- Fingerprint recognition
- Traffic control systems
- Brake light detection
- Machine vision

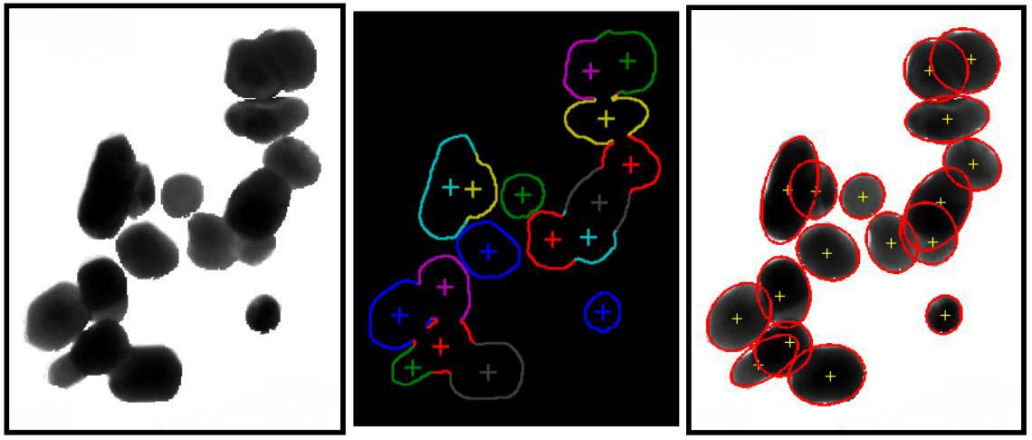
Segmentation



Otsu's Segmentation Method

From a single color and depth image, bottom-up segmentation (top-right), long range completion(bottom-left), semantic segmentation (bottom-middle) and contour classification (bottom-right)

Saurabh Gupta, Pablo Arbelaez, Jitendra Malik, Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images, Computer Vision and Pattern Recognition (CVPR), 2013



Radial symmetry based method for segmentation of clustered partially overlapping objects with a shape that can be approximated using an ellipse.

Sahar Zafari, Tuomas Eerola, Jouni Sampo, Heikki Kälviäinen, and Heikki Haario, Segmentation of Overlapping Elliptical Objects in Silhouette Images In IEEE Transactions on Image Processing 24 (12), 5942 - 5952, 2015

Segmentation

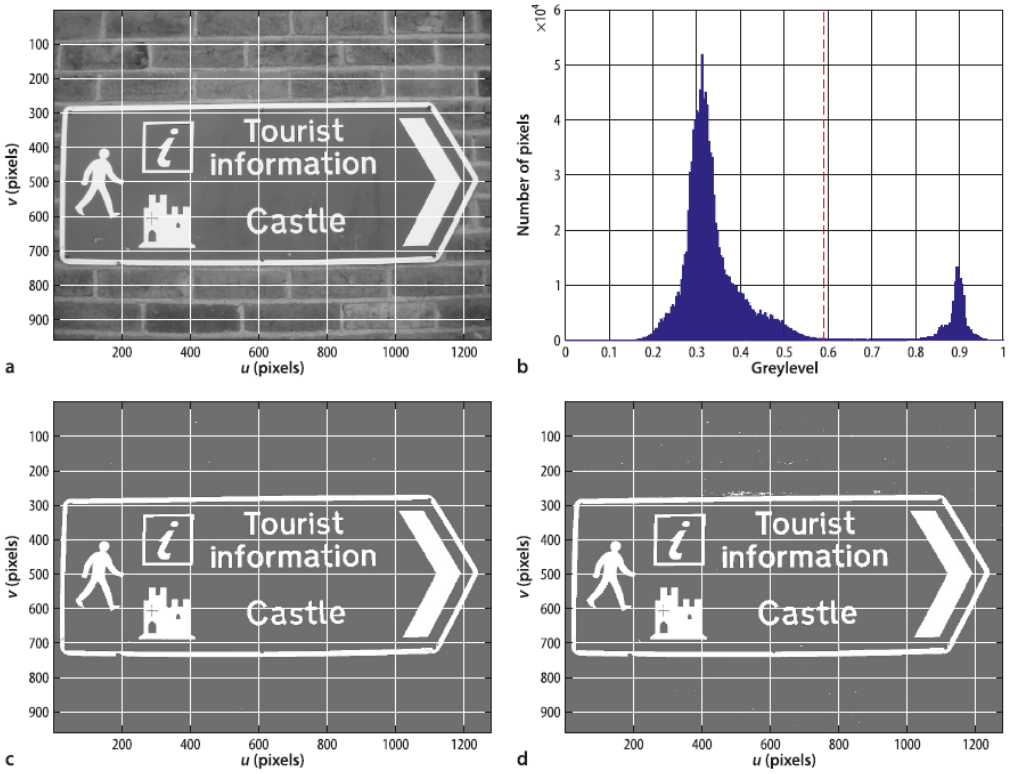
- Image segmentation is considered as three sub-problems.

- The first is *classification* which is a decision process applied to each pixel that assigns the pixel to one of *C* classes $c \in \{0 \ldots C - 1\}$.
- Commonly we use *C* = 2 which is known as binary classification or binarization.
  - The pixels have been classified as object (*c* = 1) or not-object (*c* = 0) which are displayed as white or black pixels respectively. The classification is *always application specific* – for example the object corresponds to pixels that are bright or yellow or red or moving.
- The underlying *assumption* is that regions are *homogeneous* with respect to some characteristic such as brightness, color or texture. In practice we accept that this stage is imperfect and that pixels may be misclassified – subsequent processing steps will have to deal with this.

- The second step in the segmentation process is *representation* where adjacent pixels of the same class are *connected* to form spatial sets $S_1 \ldots S_m$.

- In the third and final step, the sets S_i are *described* in terms of compact scalar or vector-valued *features* such as size, position, and shape.

Segmentation



$$c[u,v] = \begin{cases} 0, & \text{if } I[u,v] < t \\ 1, & \text{if } I[u,v] \geq t \end{cases} \quad \forall (u,v) \in I$$

Binary classification.
a Original image;
b histogram of greyscale pixel values, threshold values indicated, Otsu in *red*;
c binary classification with threshold of 0.7; `idisp(castle >= 0.7)`

d binary classification with Otsu Threshold

```
th = otsu (inim) →  0.5898
idisp(castle >= th)
```

where did the threshold value of 0.7 come from? The most common approach is trial and error.

Segmentation

We make an assumption about the scene. The characters are approximately 100 pixels tall, to choose a window half-width of 50 pixels

```
th = niblack(inim, -0.3, 50);
idisp(th)
```

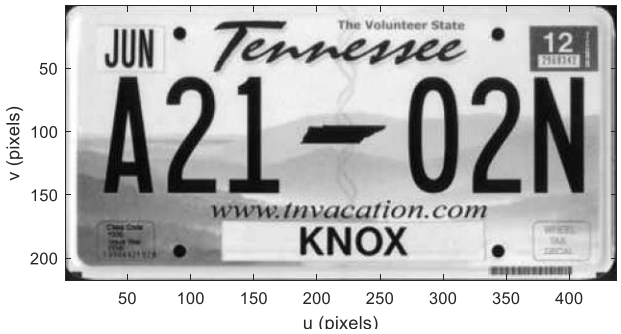
where *k*=−0.3. threshold pixel-wise to the original image

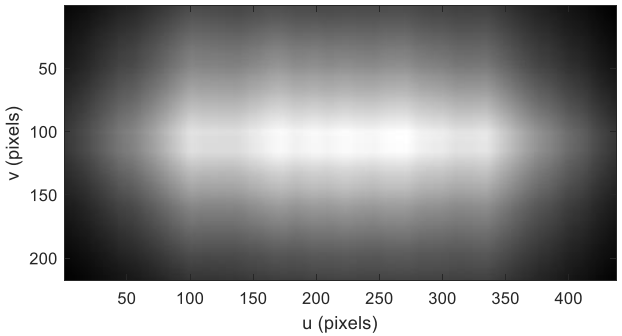
```
idisp(castle >= th)
```

- The Niblack algorithm is widely used in optical character recognition systems and computes a local threshold

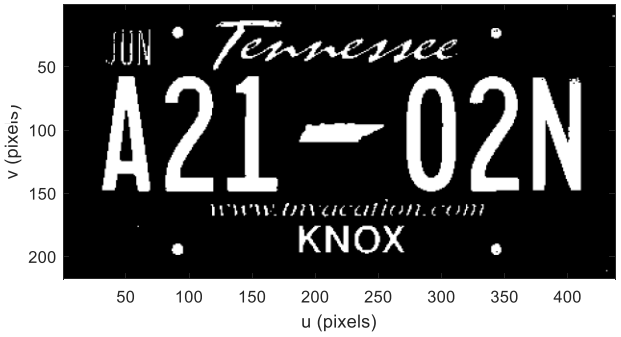
$$t[u, v] = \mu(W) - k\sigma(W)$$

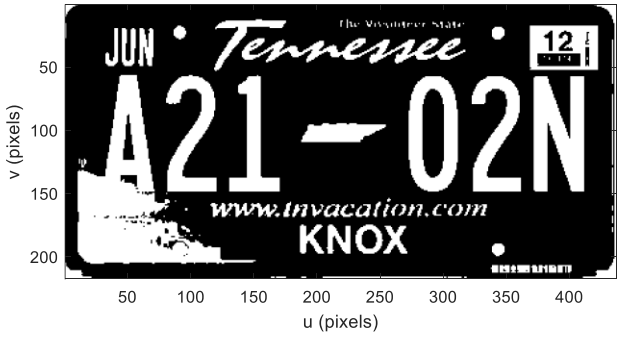
- where W is a region about the point (*u*, *v*) and μ (·) and σ (·) are the mean and standard deviation respectively. The size of the window W is a critical parameter and should be of a similar size to the objects we are looking for.





binary classification with Otsu Threshold





Segmentation

Color Classification:

- Color is a powerful cue for segmentation
- for an indoor UAV landing experiment :

```
[cls, cab,res] = colorkmeans(in_im, 2, 'ab');

cab =
-0.8190 0.4783
57.6140 -4.1910

showcolorspace(cab, 'ab');
res =
2.8897e+03

>> idisp(cls, 'colormap', flag(2), 'bar')
```

Subsequently we can assign pixels to their closest cluster relatively cheaply

```
cls = colorkmeans(in_im, cab, 'ab');
```

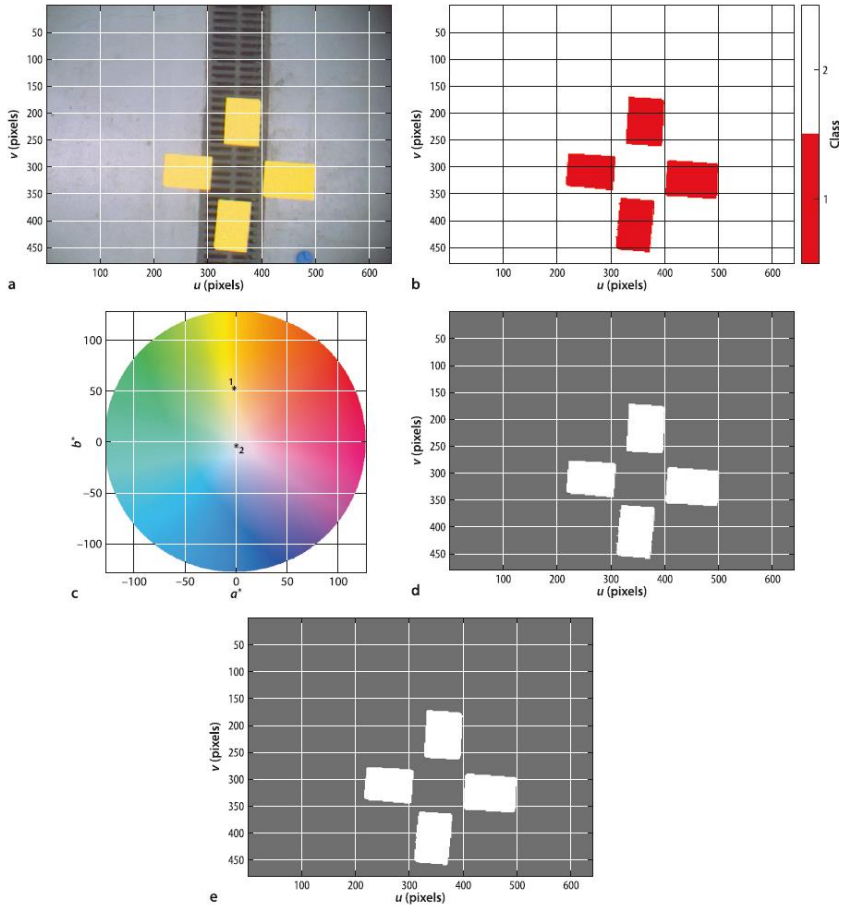
The pixels belonging to class 1 can be selected

```
cls1 = (cls == 1);
```

which is a *logical image* that can be displayed

idisp(cls1)

```
im_binary = iopen(cls1, kcircle(2));
```

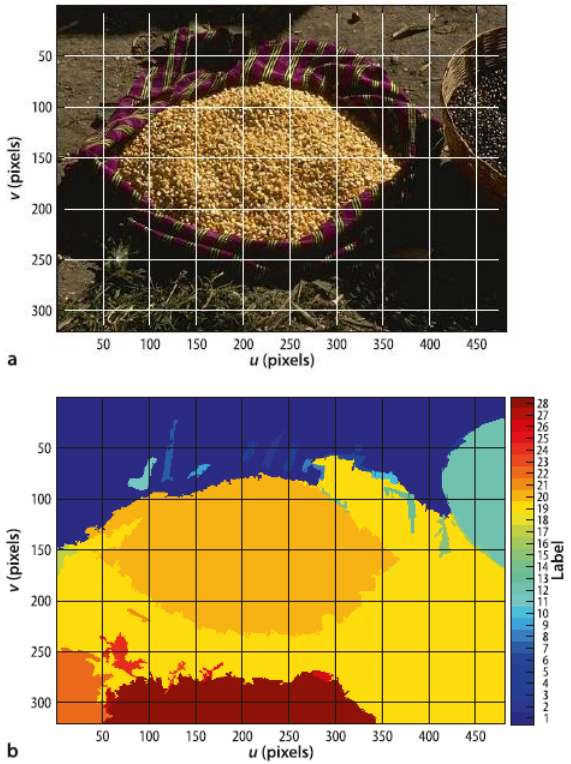


Target image example.
a Original image; **b** pixel classification (*C*=2) shown in false color; **c** cluster centers in the *a***b**-chromaticity space; **d** all pixels of class *c*=1; **e** after morphological opening with a circular structuring element (radius 2)

Segmentation

- Graph-Based Segmentation

```
[label, m] = igraphseg(im, 1500, 100, 0.5);
m =
28
idisp(label, 'colormap', 'jet')
```



- The image can be represented as a graph where each pixel is a vertex and has 8 edges connecting it to its neighboring pixels.
- The weight of each edge is a nonnegative measure of the dissimilarity between the two pixels – the absolute value of the difference in color.

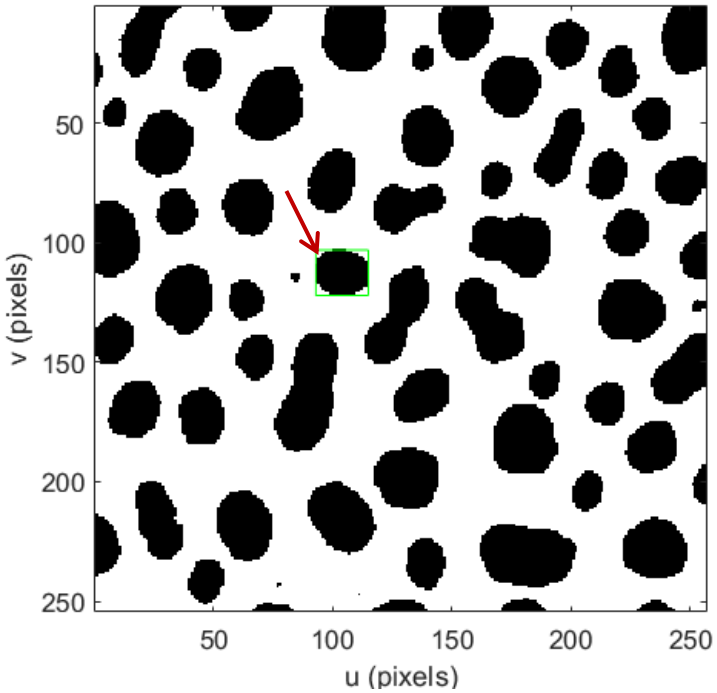
- The algorithm starts with every vertex assigned to its own set.
- At each iteration the edge weights are examined and if the vertices are in different sets but the edge weight is below a threshold the two vertex sets are merged.
- The threshold is a function of the size of the set and a global parameter *k* which sets the scale of the segmentation – a larger value of *k* leads to a preference for larger connected components.

Segmentation - Description

- We learned so far how to find connected components in the image and how to isolate particular components.
- However this representation of the component is still just an image with logical pixel values rather than a concise *description* of its size, position and shape.

```
im_in = iread('blobs.tif');
[label, m] = ilabel(im_in);
blob = (label == 29);
sum(blob(:))
[v,u] = find(blob);

umin = min(u)
umax = max(u)
vmin = min(v)
vmax = max(v)
idisp(im_in)
hold on
plot_box(umin, vmin, umax, vmax, 'g')
```



Segmentation - Description

- **Moments**
- are a rich and computationally cheap class of image features which can describe region size and location as well as shape.
- ***Ordinary moment*** of the order **p, q** for a discrete (image) function **I (u,v)** is
- Moments are concrete physical properties of a region

$$m_{pq} = \sum_{(u,v)\in\mathcal{R}} I(u,v)\cdot u^p v^q$$

Taking the p^{th} moment in u direction
And q^{th} moment in v direction

- **Centroid** can be expressed as

$$\bar{x} = \frac{1}{|\mathcal{R}|}\cdot\sum_{(u,v)\in\mathcal{R}} u^1 v^0 = \frac{m_{10}(\mathcal{R})}{m_{00}(\mathcal{R})}$$

$$\bar{y} = \frac{1}{|\mathcal{R}|}\cdot\sum_{(u,v)\in\mathcal{R}} u^0 v^1 = \frac{m_{01}(\mathcal{R})}{m_{00}(\mathcal{R})}$$

Central moments:

- **Order p, q central moments** can be calculated as:

$$\mu_{pq}(\mathcal{R}) = \sum_{(u,v)\in\mathcal{R}} I(u,v)\cdot(u-\bar{x})^p\cdot(v-\bar{y})^q$$

- For binary image with $I(u,v)$ = 1 or 0

$$\mu_{pq}(\mathcal{R}) = \sum_{(u,v)\in\mathcal{R}} (u-\bar{x})^p\cdot(v-\bar{y})^q$$

Segmentation - Description

- ***Normalized central moments:***

$$\bar{\mu}_{pq}(\mathcal{R}) = \mu_{pq} \cdot \left(\frac{1}{\mu_{00}(\mathcal{R})}\right)^{(p+q+2)/2} \quad (p + q) >= 2$$

The inertia of the region about axes parallel to the *u*- and *v*-axes and intersecting at the centroid of the region is given by the symmetric matrix:

$$\mathbf{J} = \begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix}$$

```
u20 = upq(blob, 2, 0); u02 = upq(blob, 0,
2); u11 = upq(blob, 1, 1);
J = [ u20 u11; u11 u02]
J =
1.0e+06 *
7.8299 -2.9169
-2.9169 4.7328
plot_ellipse(4*J/m00, [uc, vc], 'b');
lambda = eig(J)
lambda =
1.0e+06 *
2.9788
9.5838
```

The maximum and minimum radii of the equivalent ellipse are

$$a = 2\sqrt{\frac{\lambda_2}{m_{00}}}, \quad b = 2\sqrt{\frac{\lambda_1}{m_{00}}}$$

```
>> a = 2 * sqrt(lambda(2) / m00)
a =
70.4313
>> b = 2 * sqrt(lambda(1) / m00)
b =
39.2663
```

in units of pixels. These lengths are characteristic of this particular shape and are invariant to rotation. The aspect ratio of the region

```
>> b/a
ans =
0.5575
```

Segmentation - Description

```
[x,lambda] = eig(J);
x =
-0.5153 -0.8570
-0.8570 0.5153
```

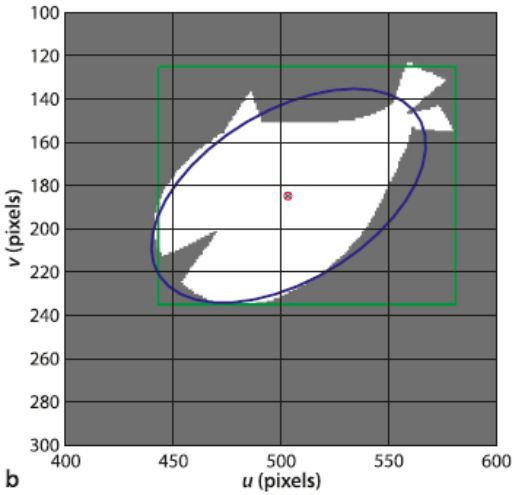
MATLAB returns eigenvalues in increasing order v is always the *last* column of the returned eigenvector matrix

```
v = x(:,end);
```

The angle of this vector with respect to the horizontal axis is and for our example this is

```
atand( v(2)/v(1) )
ans =
-31.0185
```

degrees which indicates that the major axis of the equivalent ellipse is approximately 30 degrees above horizontal.



Segmentation - Description

$$
\begin{aligned}
H_1 &= \bar{\mu}_{20} + \bar{\mu}_{02} \\
H_2 &= (\bar{\mu}_{20} - \bar{\mu}_{02})^2 + 4\,\bar{\mu}_{11}^2 \\
H_3 &= (\bar{\mu}_{30} - 3\,\bar{\mu}_{12})^2 + (3\,\bar{\mu}_{21} - \bar{\mu}_{03})^2 \\
H_4 &= (\bar{\mu}_{30} + \bar{\mu}_{12})^2 + (\bar{\mu}_{21} + \bar{\mu}_{03})^2 \\
H_5 &= (\bar{\mu}_{30} - 3\,\bar{\mu}_{12}) \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot \left[(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - 3(\bar{\mu}_{21} + \bar{\mu}_{03})^2\right] \\
&\quad + (3\,\bar{\mu}_{21} - \bar{\mu}_{03}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}) \cdot \left[3\,(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2\right] \\
H_6 &= (\bar{\mu}_{20} - \bar{\mu}_{02}) \cdot \left[(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2\right] \\
&\quad + 4\,\bar{\mu}_{11} \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}) \\
H_7 &= (3\,\bar{\mu}_{21} - \bar{\mu}_{03}) \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot \left[(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - 3\,(\bar{\mu}_{21} + \bar{\mu}_{03})^2\right] \\
&\quad + (3\,\bar{\mu}_{12} - \bar{\mu}_{30}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}) \cdot \left[3\,(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2\right]
\end{aligned}
$$

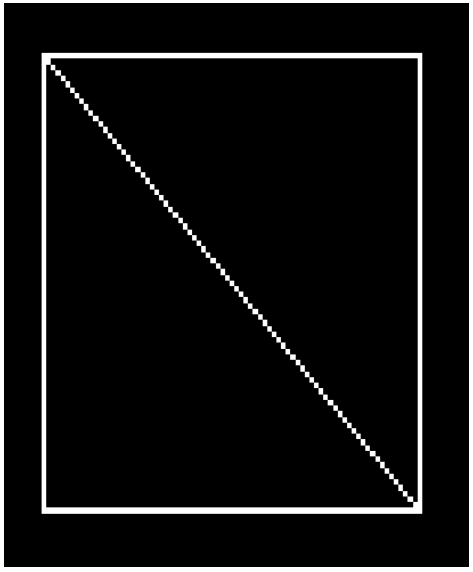
	Translation	Rotation	Scale
Area	✓	✓	×
Centroid	×	✓	✓
Orientation θ	✓	×	✓
Aspect ratio	✓	✓	✓
Circularity	✓	✓	✓
Hu moments	✓	✓	✓

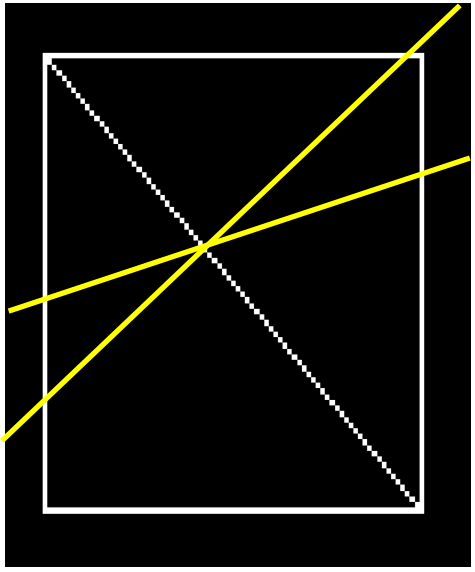
Moments called **Hu's moments** (seven combinations of normalized central moments) are invariant to translation, scaling and rotation

Line Features

- Lines are distinct visual features that are particularly common in man-made environments for example the edges of roads, buildings and doorways.
- We discussed how image intensity gradients can be used to find edges within an image, and now will be concerned with fitting line segments to such edges.

- The Hough transform is a common approach to finding parameterized line segments (here straight lines)
- Each straight line in this image can be described by an equation
- Each white point if considered in isolation could lie on an infinite number of straight lines





Line Features - Hough Transform

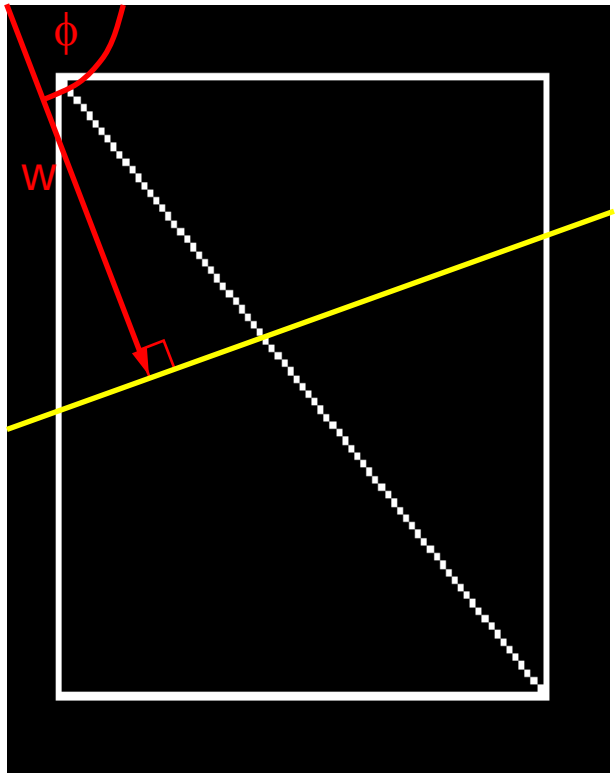
- How can we find and describe more complex features?
- The Hough transform is a common approach to finding parameterized line segments (here straight lines)

- In the Hough transform each point votes for every line it could be on .
- The lines with the most votes win.

- Any line can be represented by two numbers
- Here we will represent the yellow line by (w, ϕ)

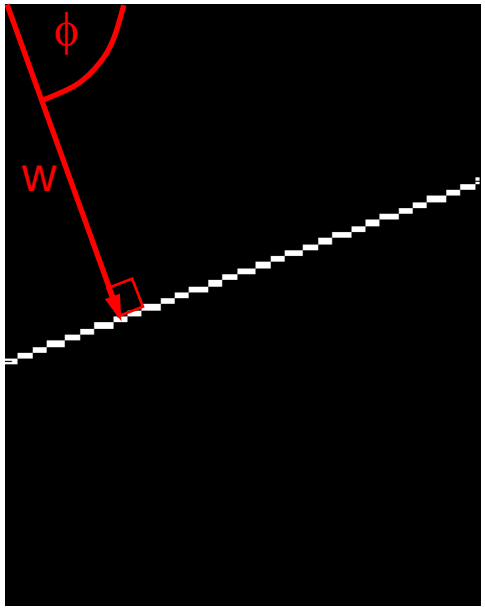
- In other words we define it using

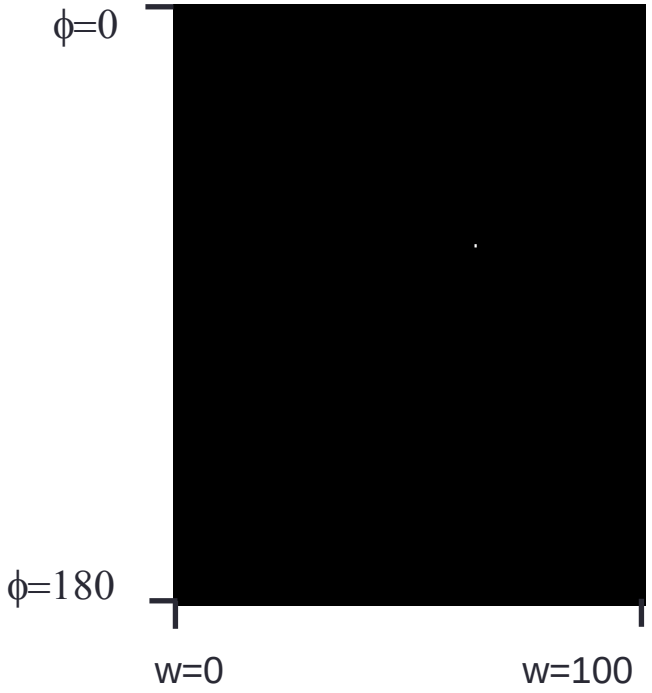
- a line from an agreed origin
- of length w
- at angle ϕ to the horizontal



Line Features - Hough Transform

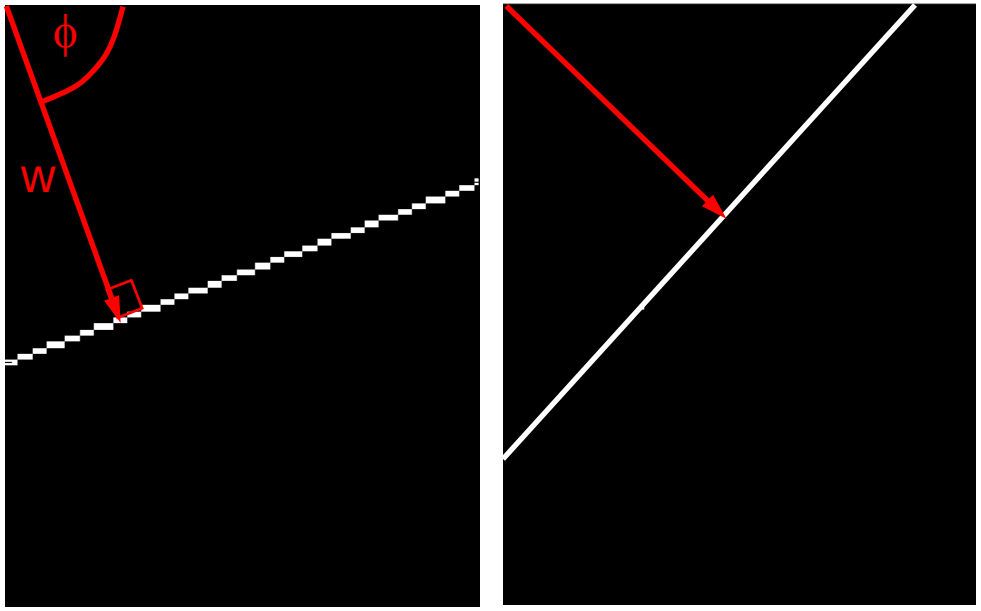
- Since we can use (w, ϕ) to represent any line in the image space
- We can represent any line in the image space as a point in the plane defined by (w, ϕ)
- This is called Hough space

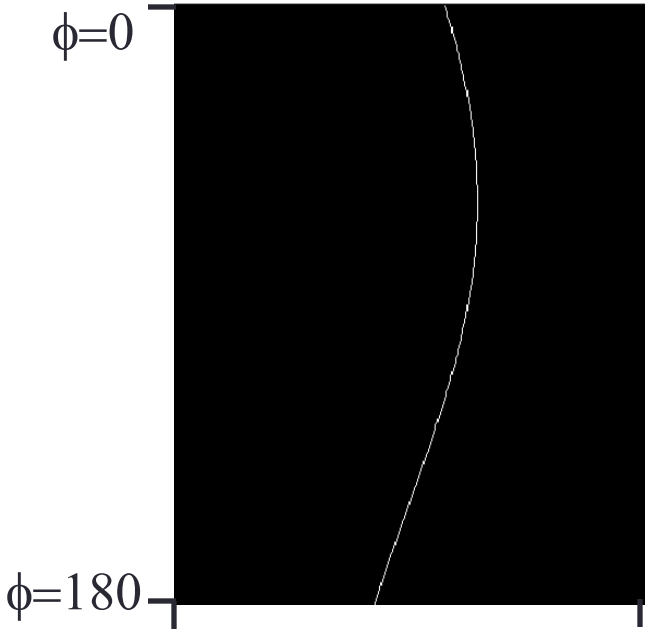




Line Features - Hough Transform

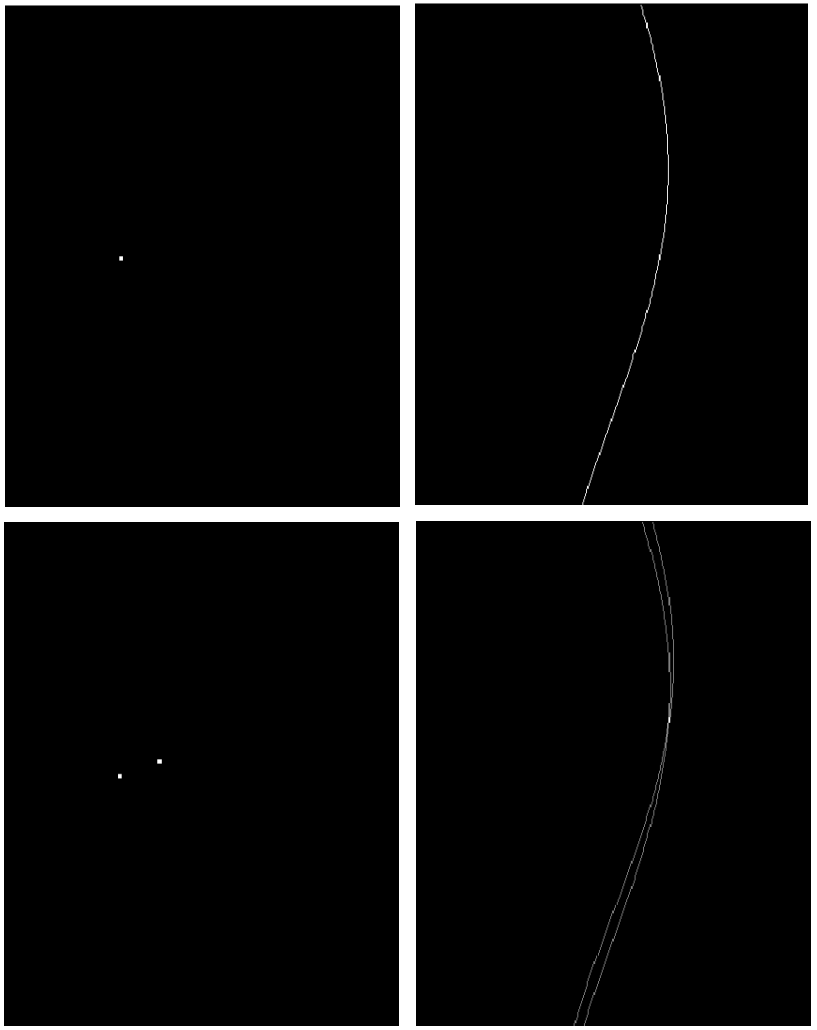
$$w = x\cos(\phi) + y\sin(\phi)$$





Line Features - Hough Transform

- One point in image space corresponds to a sinusoidal curve in image space
- Two points correspond to two curves in Hough space
- The intersection of those two curves has “two votes”.
- This intersection represents the straight line in image space that passes through both points



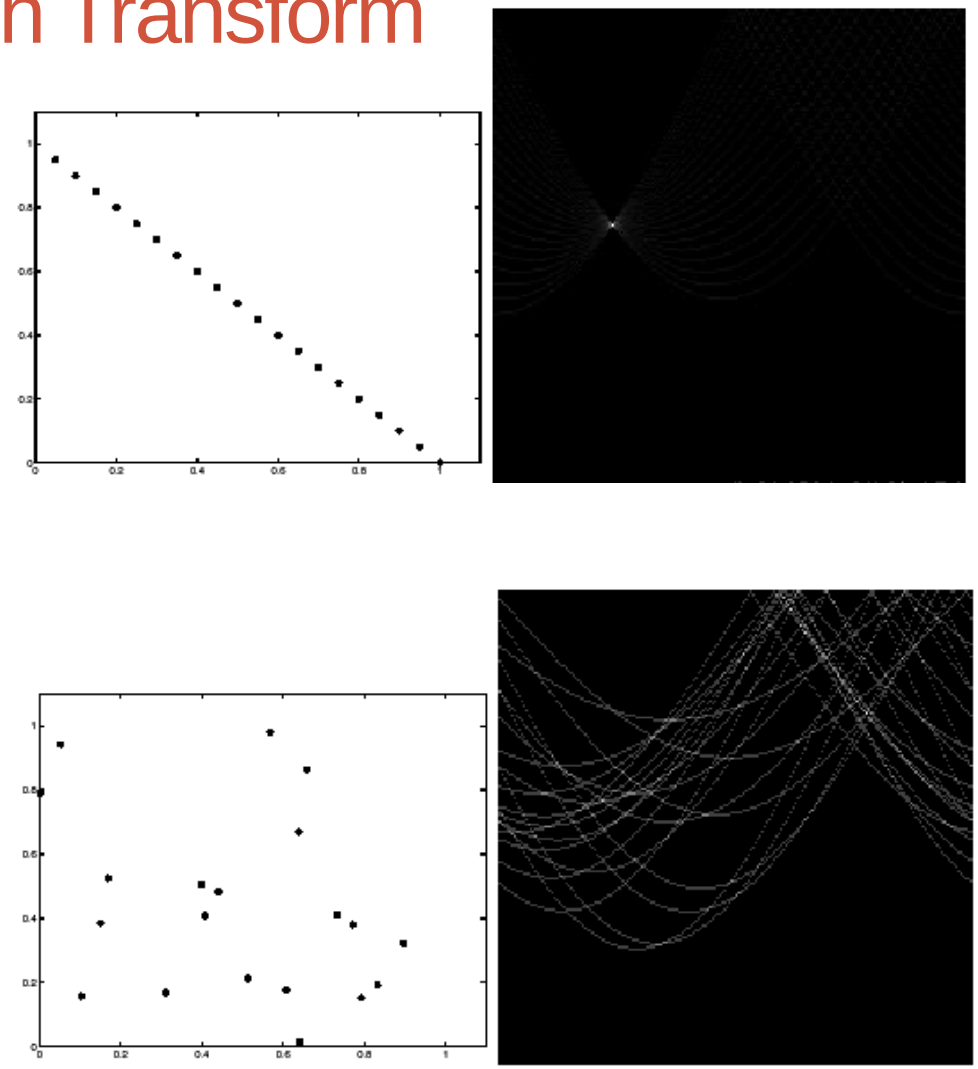
Line Features - Hough Transform

```
Create ϕ and w for all possible lines
Create an array A indexed by ϕ and w

for each point (x,y)
  for each angle ϕ
          w = x*cos(ϕ)+ y*sin(ϕ)
          A[ϕ,w] = A[ϕ,w]+1
  end
end

where A > Threshold return a line
```

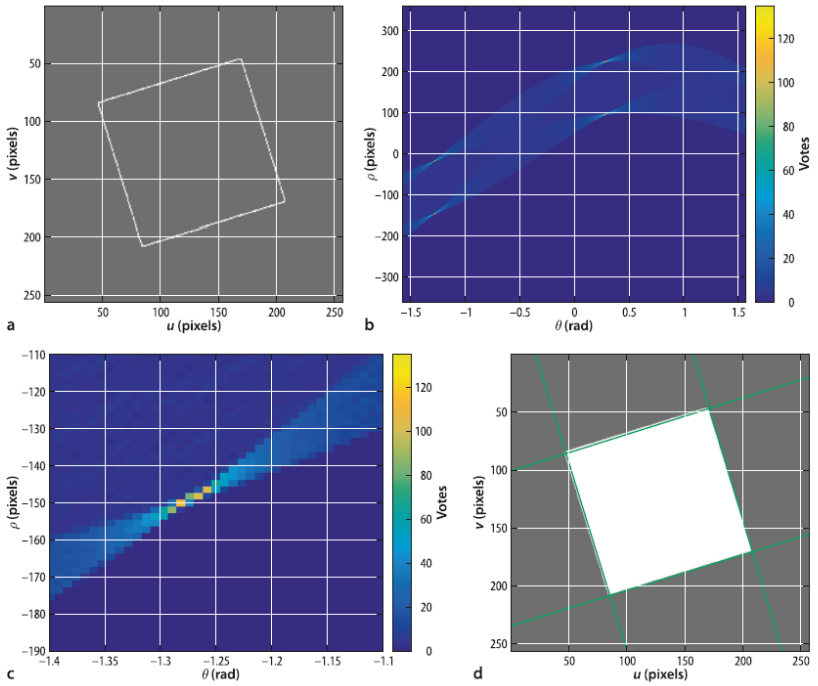
- There are generalized versions for ellipses, circles
- For the straight line transform we need to supress non-local maxima
- The input image could also benefit from edge thinning
- Single line segments not isolated
- Will still fail in the face of certain textures



Votes
Horizontal axis is θ, vertical is rho.

Line Features - Hough Transform

```
edges = icanny(in_im);
h = Hough(edges, 'suppress', 10);
lines = h.lines();
icanny(in_im, 'dark');
lines(1:10).plot('g.');
```



Hough transform for
a rotated square. **a** Edge image; **b** Hough accumulator; **c** closeup view of the Hough accumulator;
d estimated lines overlaid on the original image

Point Features

- These are visually distinct points in the image that are known as interest points, salient points, keypoints or corner points.
- We will introduce some classical techniques for finding interest points and more recent scale-invariant techniques as well.

Moravec's interest operator

$$s(u,v,\delta_u,\delta_v) = \sum_{(i,j)\in\mathcal{W}} \left(\boldsymbol{I}[u+\delta_u+i, v+\delta_v+j] - \boldsymbol{I}[u+i, v+j]\right)^2$$

- SSD Similarity is evaluated for displacements in eight cardinal directions the minimum value is the interest measure which has a large value only if all the displaced patches are different to the original
- patch.

$$C_M(u,v) = \min_{(\delta_u,\delta_v)\in\mathcal{D}} s(u,v,\delta_u,\delta_v)$$

- We can generalize the approach by defining the similarity as the weighted sum of squared differences between the image region and the displaced region as

$$s(u,v,\delta_u,\delta_v) = \sum_{(i,j)\in\mathcal{W}} \boldsymbol{W}[i,j]\left(\underbrace{\boldsymbol{I}[u+\delta_u+i, v+\delta_v+j]} - \boldsymbol{I}[u+i, v+j]\right)^2$$

where $\boldsymbol{W}$ is a weighting matrix that emphasizes points closer to the center of the window $\mathcal{W}$

Point Features

- The indicated term can be approximated by a truncated Taylor series

$$I[u+\delta_u, v+\delta_v] \approx I[u,v] + I_u[u,v]\delta_u + I_v[u,v]\delta_v$$

where I_u and I_v are the horizontal and vertical image gradients respectively

$$\begin{aligned} s(u,v,\delta_u,\delta_v) &= \sum_{(i,j)\in\mathcal{W}} W[i,j]\left(I_u[u+i,v+j]\delta_u + I_v[u+i,v+j]\delta_v\right)^2 \\ &= \delta_u^2 \sum_{(i,j)\in\mathcal{W}} W[i,j] I_u^2[u+i,v+j] + \delta_v^2 \sum_{(i,j)\in\mathcal{W}} W[i,j] I_v^2[u+i,v+j] \\ &\quad + \delta_u\delta_v \sum_{(i,j)\in\mathcal{W}} W[i,j] I_u[u+i,v+j] I_v[u+i,v+j] \end{aligned}$$

$$A = \begin{pmatrix} \Sigma W[i,j] I_u^2[u+i,v+j] & \Sigma W[i,j] I_u[u+i,v+j] I_v[u+i,v+j] \\ \Sigma W[i,j] I_u[u+i,v+j] I_v[u+i,v+j] & \Sigma W[i,j] I_v^2[u+i,v+j] \end{pmatrix}$$

$$A = \begin{pmatrix} G(\sigma_I) * I_u^2 & G(\sigma_I) * I_u I_v \\ G(\sigma_I) * I_u I_v & G(\sigma_I) * I_v^2 \end{pmatrix} \leftarrow$$

If the weighting matrix is a Gaussian kernel $W = G(\sigma I)$ and we replace the summation by a convolution

Point Features

- A is a symmetric 2 × 2 matrix referred to variously as the structure tensor, autocorrelation matrix or second moment matrix.
- It captures the intensity structure of the local neighborhood and its eigenvalues provide a rotationally invariant description of the neighborhood.
- The elements of the ***A*** matrix are computed from the image gradients, squared or multiplied, and then smoothed using a weighting matrix.
- The latter reduces noise and improves the stability and reliability of the detector. The gradient images $\boldsymbol{I}_u$ and $\boldsymbol{I}_v$ are typically calculated using a derivative of Gaussian kernel method.

- The **Harris detector** is based on this same insight but defines corner strength.

- The Hessian is the matrix of second-order gradients at a point

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{I}_{uu} & \boldsymbol{I}_{uv} \\ \boldsymbol{I}_{uv} & \boldsymbol{I}_{vv} \end{pmatrix}$$

$$\boldsymbol{I}_{uu} = \partial^2 \boldsymbol{I} / \partial u^2, \boldsymbol{I}_{vv} = \partial^2 \boldsymbol{I} / \partial v^2$$

$$\boldsymbol{I}_{uv} = \partial^2 \boldsymbol{I}^2 / \partial u \partial v.$$

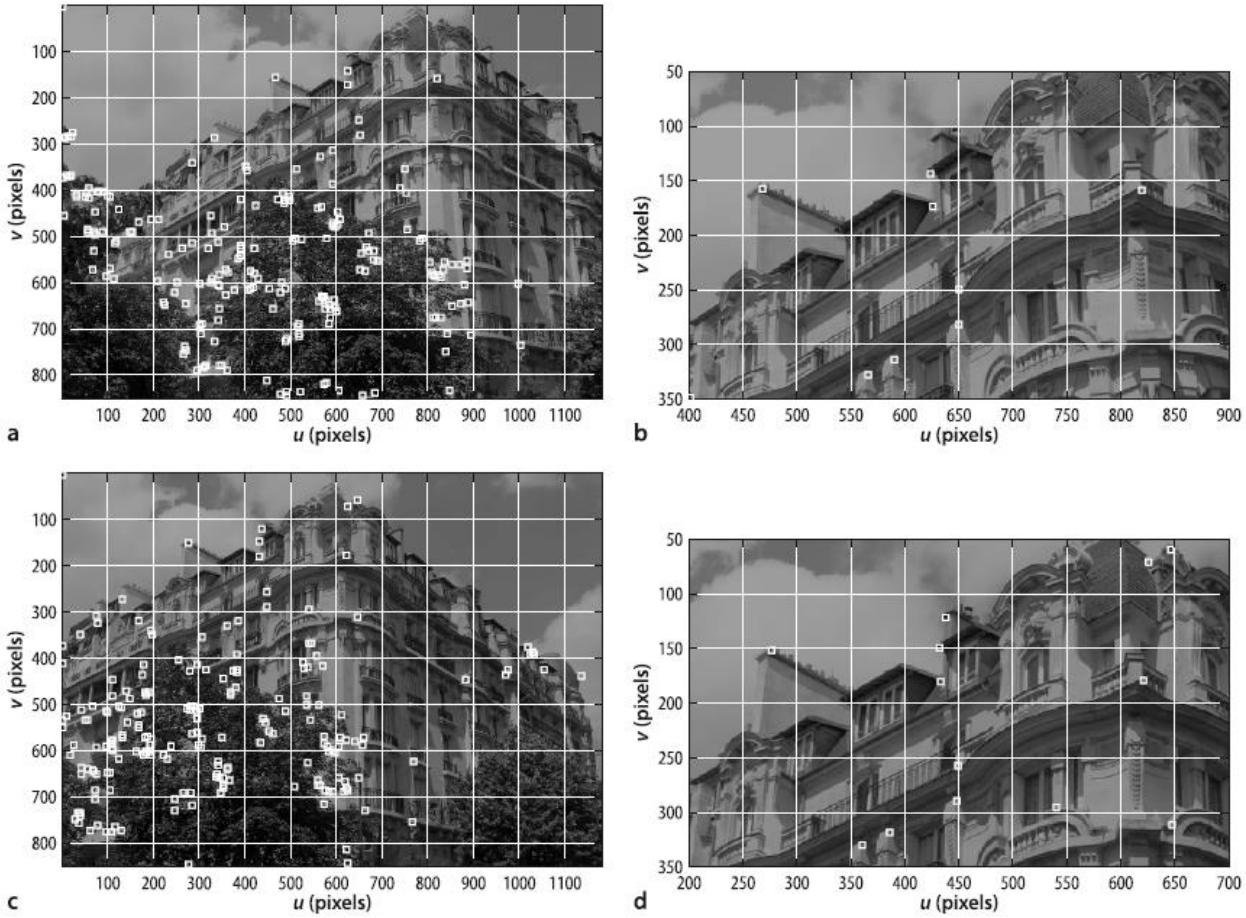
```
b1 = iread('building2-1.png', 'grey', 'double');
idisp(b1)
```

The Harris features are computed by

```
C = icorner(b1, 'nfeat', 200);
```

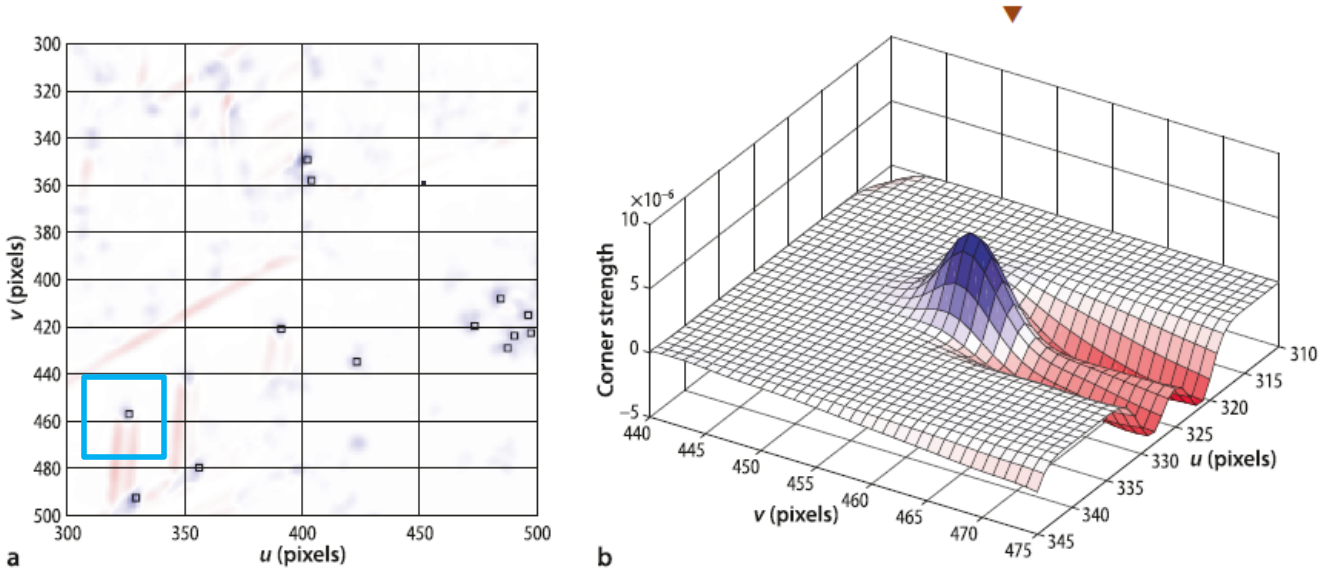
7497 corners found (0.8%), 200 corner features saved which returns a vector of PointFeature objects.

Point Features – Harris Corner detector



Harris corner detector applied to two views of the same building. **a** View one; **b** zoomed in view one; **c** view two; **d** zoomed in view two. Notice that quite a number of the detected corners are attached to the same world features in the two views

Point Features – Harris Corner detector



Harris corner strength.
a Zoomed view of corner strength
displayed as an image (*blue* is positive, *red* is negative); **b** zoomed view of corner strength image displayed as a surface

For many useful applications in robotic vision – such as tracking, mosaicing and stereo vision that we will discuss in the next chapter – it is important that corner features are detected at the same world points irrespective of variation in illumination or changes in rotation and scale between the two views

Point Features

- **Scale-Space Point Feature**

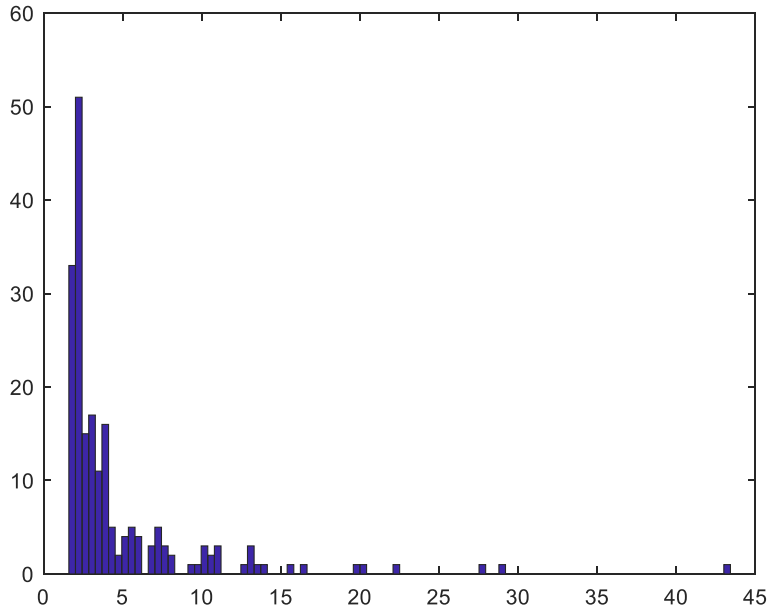
- The scale-space concepts just discussed underpin a number of popular feature detectors which find salient points within an image and determines their scale and also their orientation

- The Scale-Invariant Feature Transform (SIFT) is based on the maxima in a difference of Gaussian sequence. The Speeded Up Robust Feature (SURF) is based on the maxima in an approximate Hessian of Gaussian sequence.

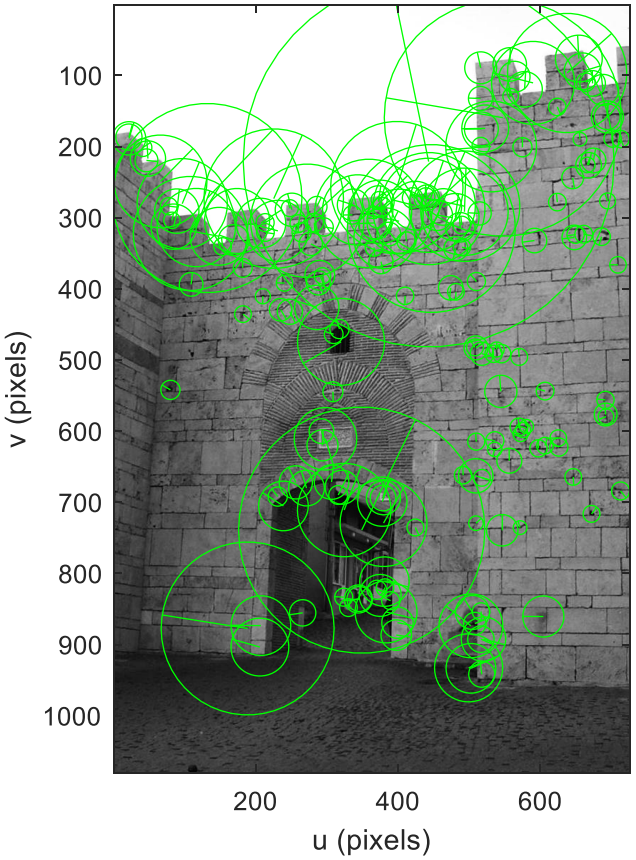
```
im = iread('gate.tif', 'double');
sf1 = isurf(im, 'nfeat', 200)

idisp(im, 'dark');
sf1.plot_scale('g', 'clock')
hold off
hist(sf1.scale, 100);
```

Point Features - SURF



Histogram of feature scales shown with logarithmic vertical scale



SURF descriptors showing the support region (scale) and orientation as a radial line

Point Features - SURF

Matched SURF features : 7

ImageJ SURF

to conclude …

- In this chapter we have studied the extraction of features from an image.
- Instead of considering the image as millions of independent pixel values we succinctly describe regions within the image that correspond to distinct objects in the world.
- For instance we can find regions that are homogeneous with respect to intensity or color and describe them in terms of features such as a bounding box, centroid, equivalent ellipse, aspect ratio, circularity and perimeter shape.

- Features have invariance properties with respect to translation, rotation about the optical axis and scale which are important for object recognition. Straight lines are common visual features in man-made environments and we showed how to find and describe distinct straight lines in an image using the Hough transform.

- We also showed how to find interest points that can reliably *associate* to particular points in the world irrespective of the camera view.