Overview

- CBIR has two phases:
  - Database Population phase
    - Image/video shot extraction
    - Feature extraction
  - Retrieval phase
    - Similarity measure

What do users look for?

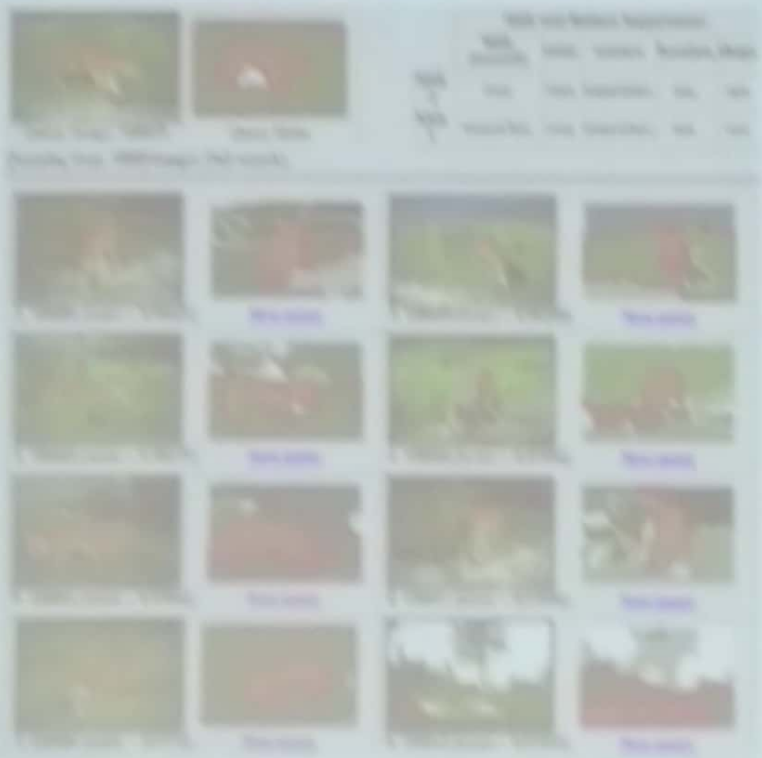
- *In CBIR systems users look for 'things' not 'stuff'.
  - More than global image properties
  - Traditional object recognition will not work
  - Two choices:
    - Rely on text to identify objects
    - Look at regions (objects or parts of objects)

*Chad Carson: Blobworld

Initial Attempts

- Images represented using simple features
  - Color Histograms
  - Color moments
  - Texture descriptors
- A distance metric (e.g., Euclidean) is used to measure similarity

Initial Successful Systems



Blob World: Carson, Belongie, Malik CVPR97

SIMPLIcity: Wang, Li, Wiederhold

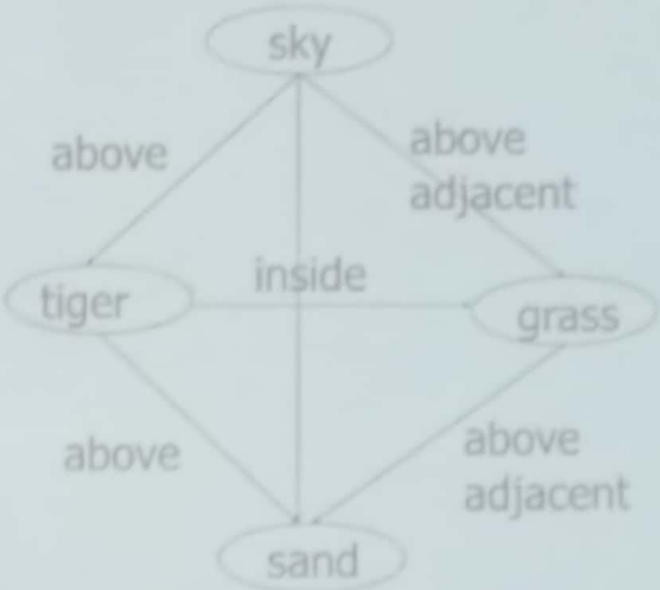
Practical CBIR systems

- Practical large scale deployment of CBIR systems require.
  - Efficient indexing and retrieval of thousands of documents.
  - Flexible framework for retrieval based on various types of features. For example Specialized features for highly specific sub domains like faces, vehicles, monuments.
  - Ability to scale up to millions of images without a significant performance trade off.

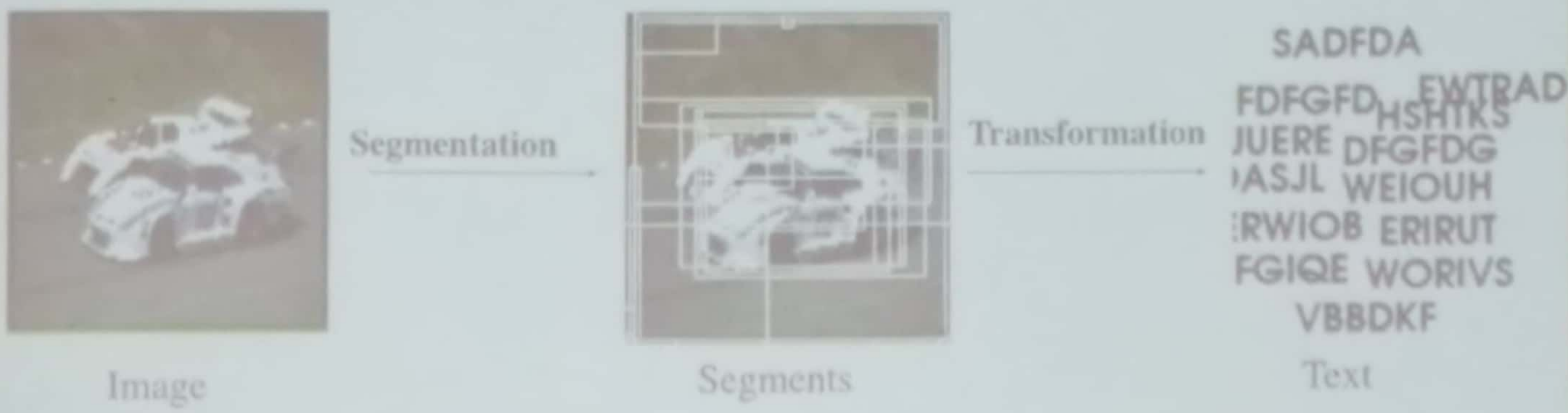
Virtual Graph Representation

image

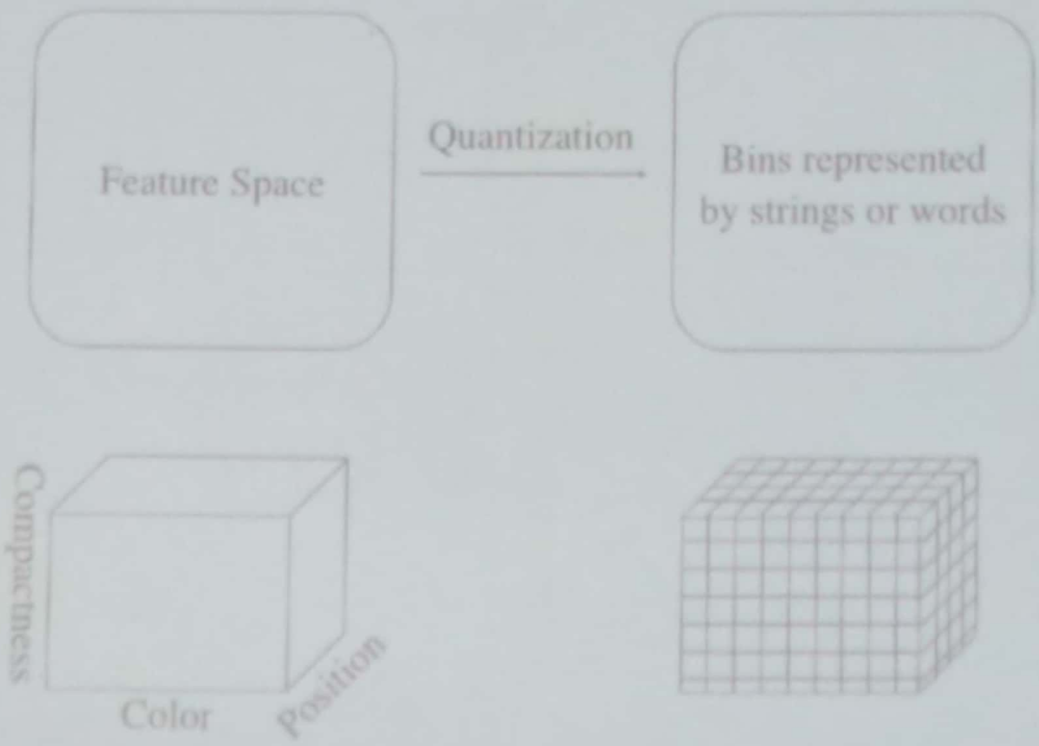
abstract regions



Virtual Textual Representation

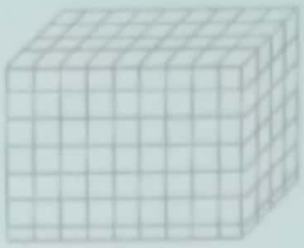


Transformation



Transformation

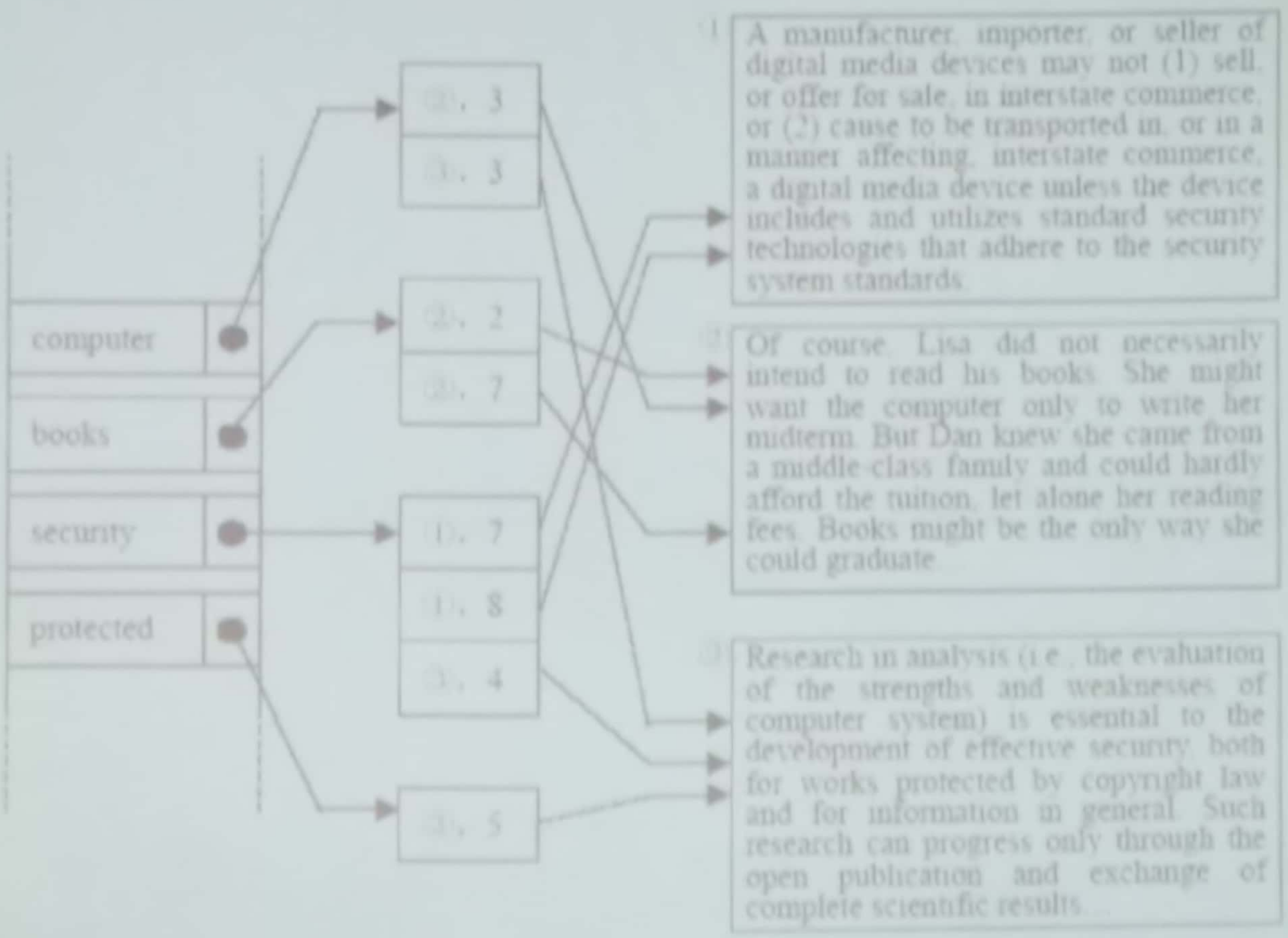
- Quantization can be achieved in a number of ways.
  - Uniform vector space quantization for data set with a uniform feature point distribution.



  - Density based quantization of the feature space can be achieved with simple k-means quantization.

- Irrespective of the quantization applied each cell in the vector space has a representative string.
- Each image segment is assigned to a cell and is assigned the representative string of the cell.

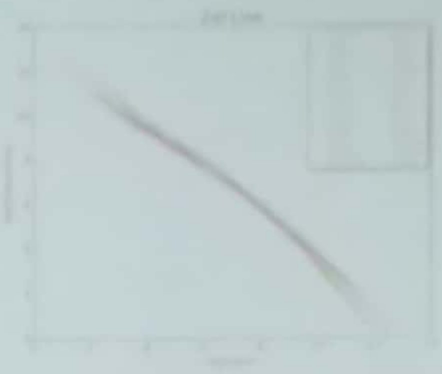
Inverted Index



Documents as Word Histograms

	nova	galaxy	heat	h'wood	film	role	diet	fur
A	10	5	3					
B	5		10					
C				10	8	7		
D				9	10	5		
E							10	10
F							9	10
G	5	7			9			
H		6	10	2	8			
I				7	5		1	3

Assigning Weights to Terms



A plot of the rank versus frequency for the first 10 million words in 30 Wikipedias (dumps from October 2015) in a log-log scale

- Binary Weights
- Raw term frequency
- tf x idf
  - Recall the Zipf distribution
  - Want to weight terms highly if they are
    - frequent in relevant documents … BUT
    - infrequent in the collection as a whole

Zipf's law states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, etc. - Wiki

Computing TF*IDF

$$w_{ik} = tf_{ik} * \log(N / n_k)$$

T_k = term k

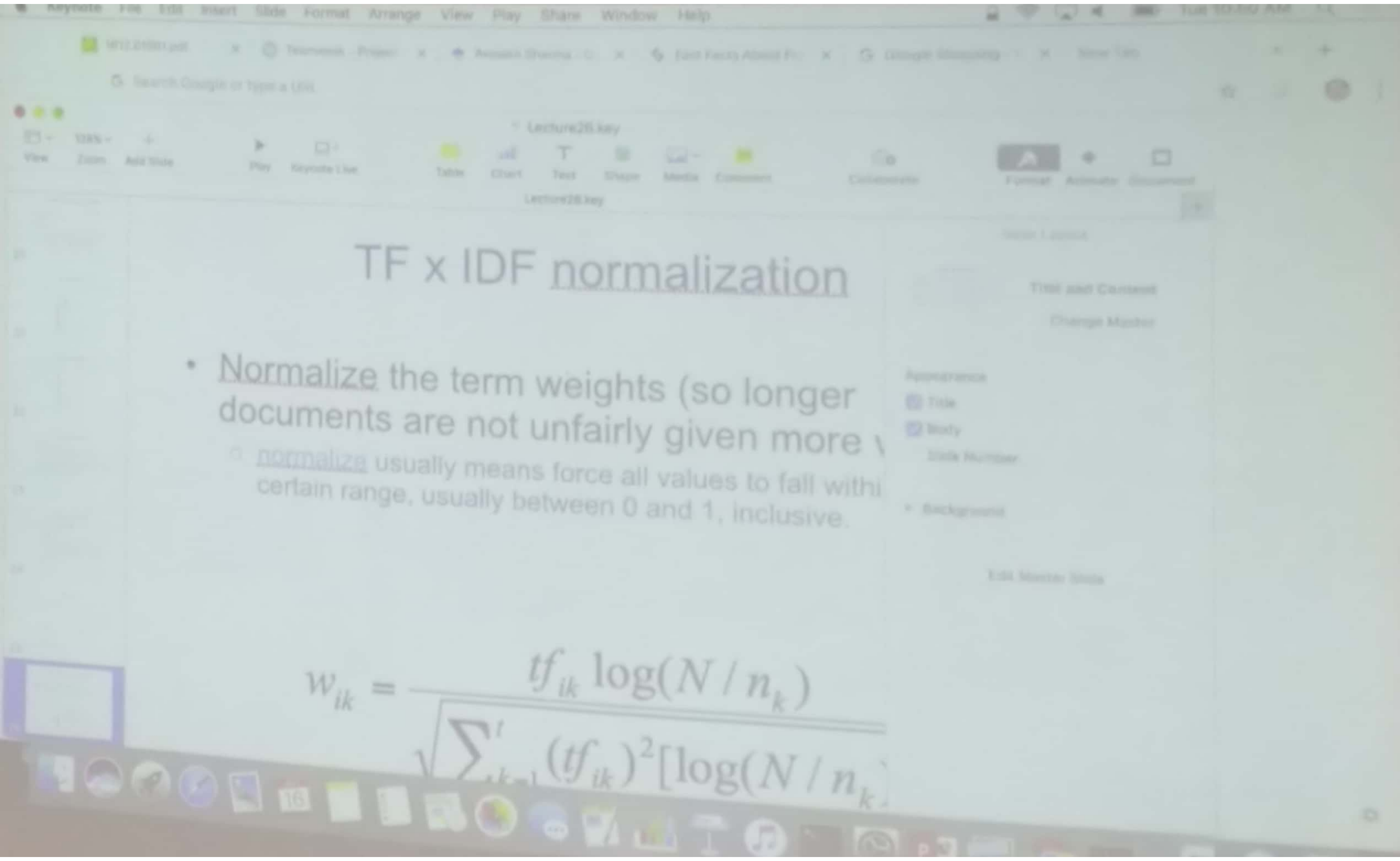
tf_{ik} = frequency of term T_k in document D_i

idf_k = inverse document frequency of term T_k in C

N = total number of documents in the collection C

n_k = the number of documents in C that contain T_k

$$idf_k = \log\left(\frac{N}{n_k}\right)$$


TF x IDF normalization
Normalize the term weights (so longer
documents are not unfairly given more
normalize usually means force all values to fall withi
certain range, usually between 0 and 1, inclusive.
$w_{ik} = \frac{tf_{ik} \log(N / n_k)}{\sqrt{\sum_{k=1}^{t} (tf_{ik})^2 [\log(N / n_k)}}$
Lecture26.key
Title and Content
Change Master
Edit Master Slide

Feature Descriptors

Local/Patch

- SIFT
- SURF
- FAST
- BRIEF
- ORB (≈FAST+BRIEF)
- GLOH

Global/Object

- HOG
- GIST
- Shape Context

Speeded Up Robust Features (SURF)
Features from accelerated segment test (FAST)
Binary Robust Independent Elementary Features (BRIEF)
Oriented FAST and rotated BRIEF (ORB)
Gradient Location and Orientation Histogram (GLOH)

Speeded Up Robust Features (SURF)

- Partially inspired from SIFT
- Several times faster than SIFT
- Claimed to be more robust than SIFT
- Feature detection: SURF uses an integer approximation of the determinant of Hessian blob detector, which can be computed with 3 integer operations using a precomputed integral image.
- Feature descriptor: SURF is based on the sum of the Haar wavelet response around the point of interest. These can also be computed with the aid of the integral image.

Detection

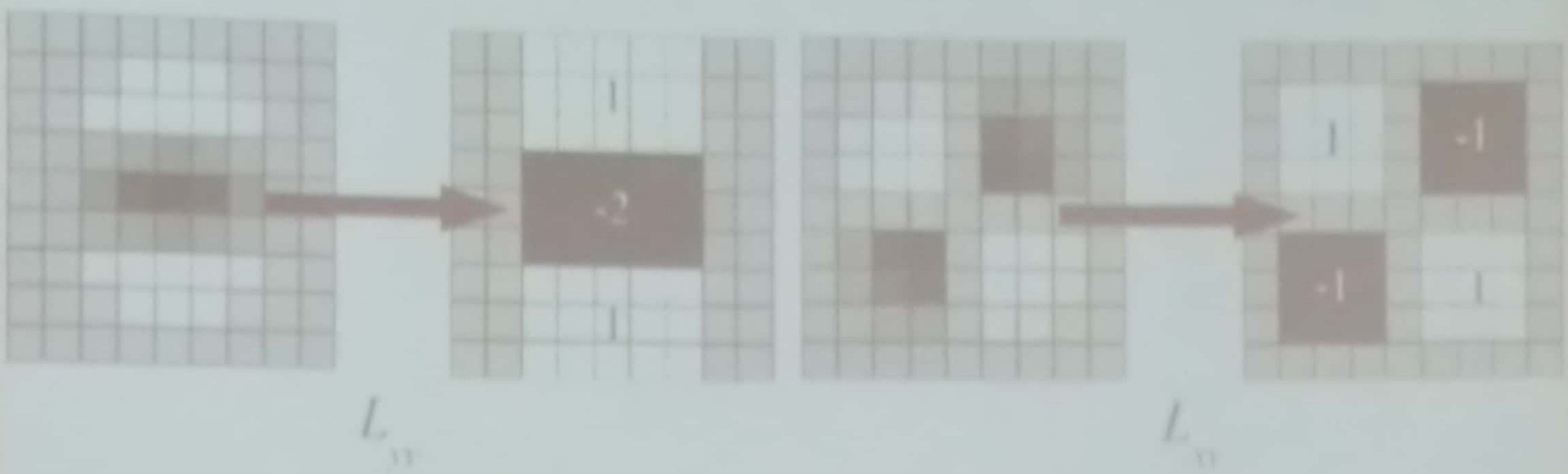
- Hessian-based interest point localization

$$H = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{bmatrix}$$

- $L_{xx}(x,y,\sigma)$ is the **Laplacian of Gaussian** of the image
- It is the convolution of the *Gaussian* second order derivative with the image
- Lindeberg showed Gaussian function is optimal for scale-space analysis
- This paper argues that Gaussian is overrated since the property that no new structures can appear while going to lower resolution is not proven in 2D case

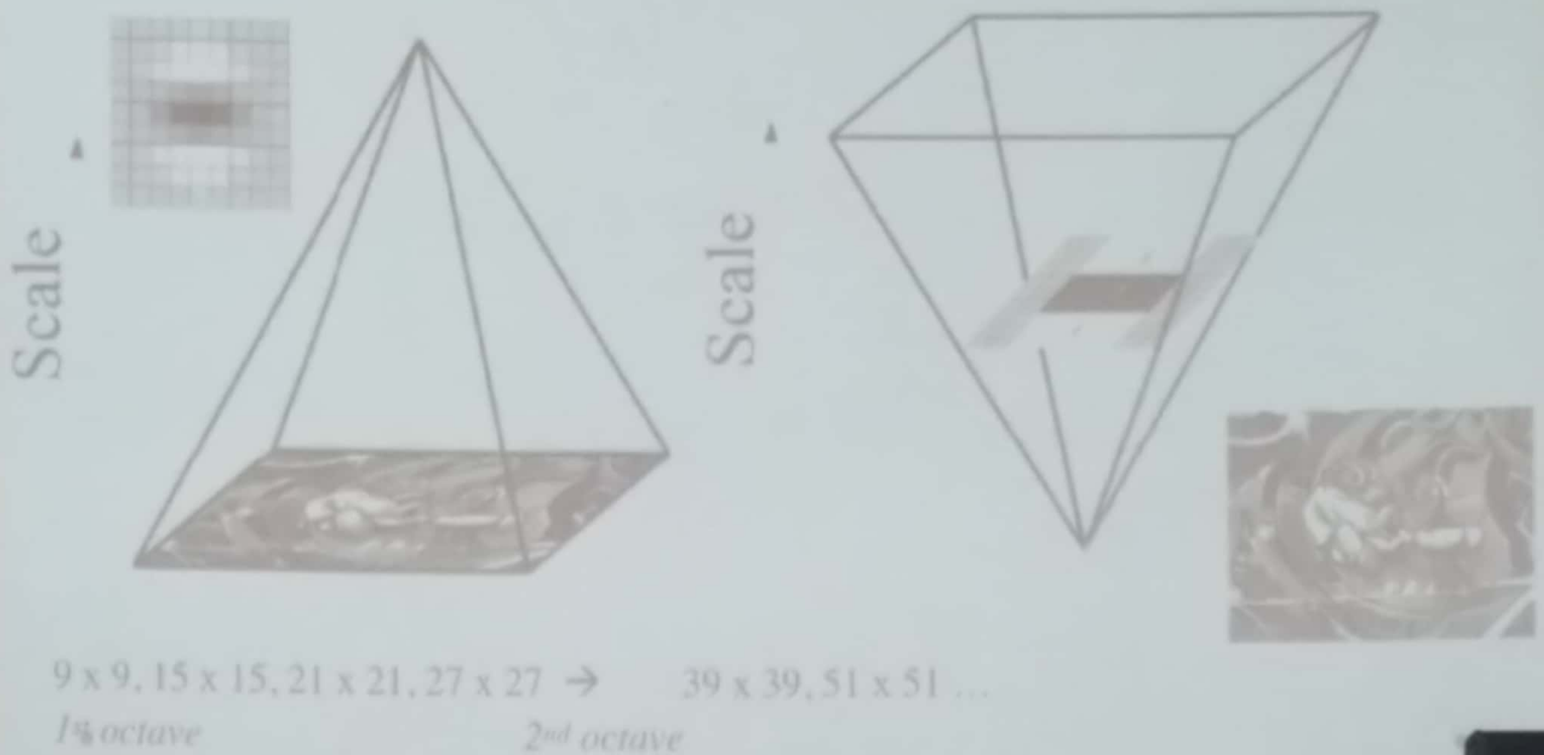
Detection

- Approximated second order derivatives with box filters (mean/average filter)



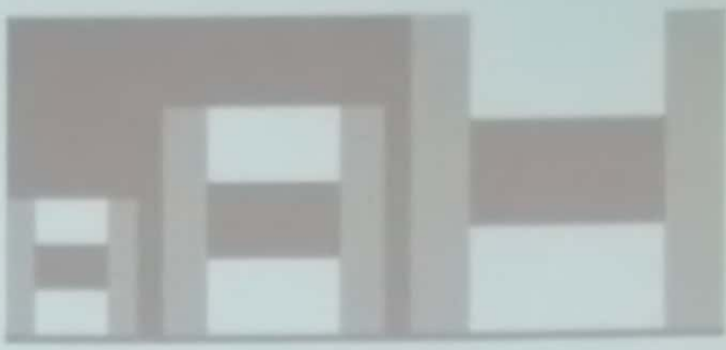
Detection

- Scale analysis with constant image size



Scale Space

- Divided into octaves – series of filter response maps with double increments on higher octave
- Begins with 9x9 filter, corresponding to σ=1.2
- Increment of 6 or higher needed for preservation of filter structure



Feature Description

Interest point *descriptor*:

- Divide window into 4x4
  (16 subwindows)
- Compute Haar wavelet outputs
- Within each subwindow, compute

$$v_{subregion} = \left[\sum dx, \sum dy, \sum |dx|, \sum |dy|\right]$$

- This yields a 64-element descriptor

(Only implement USURF – no rotation)

Orientation Computation

- A window rotates around the origin that is 60 degrees wide
- Add up the responses within the window as the vector's length
- Longest vector is the interest point's orientation

Matching

- **Fast indexing through the sign of the Laplacian for the underlying interest point**

The sign of trace of the Hessian matrix

$$\mathbf{Trace} = \mathbf{L_{xx} + L_{yy}}$$