

# Machine Vision: Recognizing objects and scenes

October 5, 2007

# Introduction

- ▶ Robots need vision for many different goals
- ▶ Humans are excellent in visual tasks
- ▶ Computer vision is difficult: images are large, noisy, and are taken from different viewpoints and lighting conditions
- ▶ There can be a huge amount of different scenes or objects
- ▶ Computer vision techniques rely on analyzing:
  - ▶ Color
  - ▶ Shape
  - ▶ Texture

# Overview

- ▶ Object and Shape Recognition with neural architectures
- ▶ Scene Recognition with texture and color descriptors
- ▶ vision on the i-Cat

# Computer Vision

- ▶ Computer Vision has been researched for a long time
- ▶ Some applications are:
  - ▶ Handwritten text recognition
  - ▶ Face recognition
  - ▶ Object recognition
  - ▶ Car and pedestrian recognition
- ▶ There are different approaches:
  1. Machine learning techniques
  2. 2D or 3D (geometrical) shape models
  3. Image features and distance functions
  4. Local descriptors

# Shape and Object Recognition

- ▶ Shapes play an important role for recognizing objects
- ▶ Shapes are invariant to color intensities and different textures
- ▶ In this presentation we will look at an object recognition approach using neural architectures
- ▶ This research was carried out by Jelmer De Vries

# Scene Recognition

- ▶ Scenes are important for self localization
- ▶ Different kinds of scenes: e.g. a bus on a street, a market
- ▶ We will concentrate on Content Based Image Retrieval (CBIR) techniques
- ▶ CBIR techniques have the goal to retrieve similar scenes from an image query
- ▶ E.g. if you submit a picture of a rose, the computer gives you back 10 other pictures of roses
- ▶ This research was done by Azizi Abdullah

# Overview: Object and Shape Recognition

- ▶ Introduction
  - ▶ The Problem
  - ▶ Difficulties
  - ▶ Ideas and Inspirations
- ▶ Steps
  - ▶ Shape Extraction
  - ▶ Descriptors
  - ▶ Interpreting Shapes
  - ▶ Combining all steps
- ▶ Experiments and Results
- ▶ Conclusion

*To what extent can a neural system combined with a shape-based approach be used to recognize objects efficiently and effectively?*

# The Problem

- ▶ Recognizing objects in pictures
- ▶ Using artificial intelligence (machine learning) techniques
- ▶ With a shape-based approach
- ▶ Bound by some constraints
  - ▶ Single object
  - ▶ Start with shapes, follow 2D, finish 3D

## General Difficulties

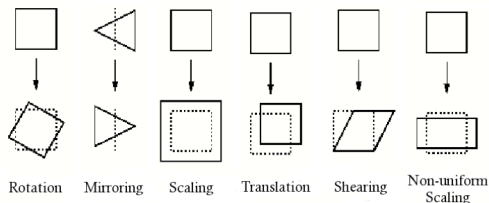
- ▶ A computer only sees pixels
  - ▶ From pixels to objects is a long way
- ▶ Working with pixels is computationally expensive
  - ▶ Several runs over all pixels are often necessary
- ▶ Working with images requires huge amounts of generalization
- ▶ Preparing data for a neural system is not always trivial

# Appearance Difficulties

- ▶ Affine transformations
  - ▶ Rotation, mirroring, translation and scaling
  - ▶ Shearing and non-uniform scaling
  - ▶  $d = b = q$  ?

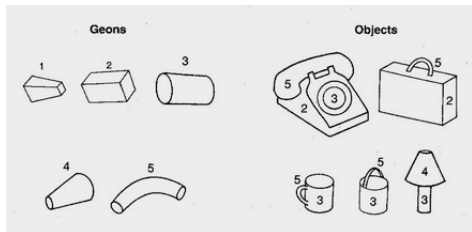
- ▶ Occlusion,  
lighting

- ▶ Multiple Objects



# Recognition by Components

- ▶ Geons, entities that make up objects
- ▶ Test based on response times
- ▶ Connections are important
- ▶ Limited role for colors and texture



# Using shapes instead of objects

- ▶ Looking at parts instead of the whole thing at once
- ▶ Many advantages
- ▶ What are these shapes?
  - ▶ Don't have to look perfect, have to be constant
- ▶ Not all perfectly analyzed; we have to put them back together

# Why neural networks?

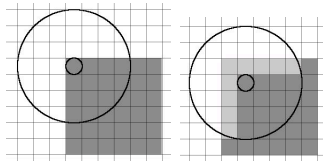
- ▶ Generalization
  - ▶ Good to overcome the small distortions
- ▶ Expandability
  - ▶ New samples hardly require any physical change
- ▶ Representing multiple samples and objects
- ▶ Memory saving
  - ▶ All the information is in the structure
- ▶ Recognition speed

# Steps

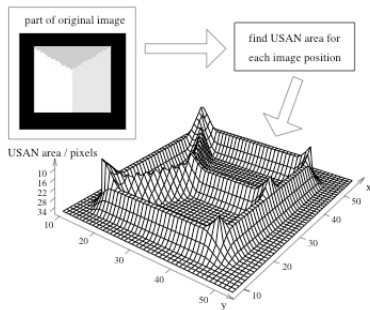
1. Extract Shapes
  - ▶ Edge extraction
  - ▶ Extract the shapes
2. Represent Shapes
3. Interpret Shapes
4. Recombine Shapes
5. Recognize Object

## Edge Extraction

- ▶ Edges found using *SUSAN Edge Detection*
- ▶ Algorithm finds *USAN* for each pixel
  - ▶ Compares pixel to its surroundings
  - ▶ Surroundings chosen to be circular
  - ▶ Pixels in surrounding that are *similar* the bigger the USAN



# Visual on SUSAN



- ▶ Resulting values cut using a threshold

# From Image to Shapes

- ▶ Dilate edges, add a pixel on all sides
- ▶ Subtract edges, take the original out
- ▶ Extract shapes, find those pixels that are connected



# Steps

1. Extract Shapes
2. Represent Shapes
  - ▶ From pixels to neural input
3. Interpret Shapes
4. Recombine Shapes
5. Recognize Object

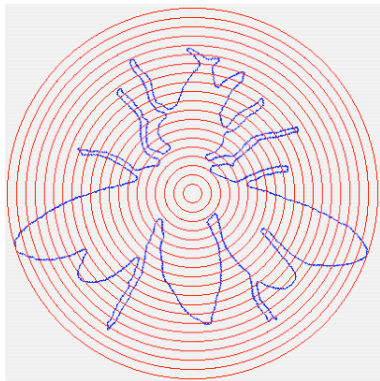
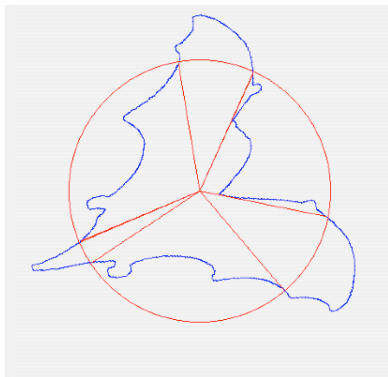
# Shape Descriptors

- ▶ Pixel / Grid
  - ▶ Take the shapes as they are, some extra preprocessing
- ▶ Orientation neurons ✓
- ▶ Distance representation
  - ▶ Measure distance from the centroid, put them in a histogram
- ▶ Radial representation ✓
- ▶ Curvature scale space representation
  - ▶ Popular approach for describing shapes

# Orientation Neurons

- ▶ Based on neurons in the brain
- ▶ Neurons that are sensitive to the orientation of a stimulus
- ▶ Two neurons interpret four by four pixels
  - ▶ One activates on horizontal lines
  - ▶ One activates on vertical lines
  - ▶ Activation is Gaussian inspired by simple cell findings

## Radial Representation



# Radial Representation

- ▶ Using circles to find an invariant solution
- ▶ Angles of crossings are invariant
- ▶ What information?
  - ▶ Biggest and smallest angle
  - ▶ Number of crossings
  - ▶  $D_{radial} = (max_1, min_1, cross_1, \dots, max_n, min_n, cross_n)$

# Steps

1. Extract Shapes
2. Represent Shapes
3. Interpret Shapes
  - ▶ **The first neural component**
4. Recombine Shapes
5. Recognize Object

# Interpreting Shapes

- ▶ 'Distances to vantage shapes'
- ▶ Selecting vantage shapes
- ▶ Neural network to learn vantage shapes
- ▶ Results in five values

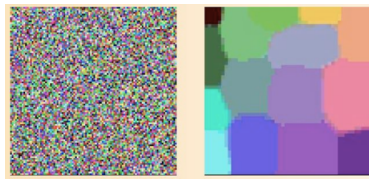


# Steps

1. Extract Shapes
2. Represent Shapes
3. Interpret Shapes
4. Recombine Shapes
  - ▶ Putting the shapes back together to represent the object
5. Recognize Object

# Self-Organizing Map

- ▶ 2D Map of nodes
- ▶ Every node has a number of weights
- ▶ Trained using vector
  - ▶ Winner takes a lot, neighbors a little
- ▶ Points mapped using winner take all



## Mapping it to the SOM

- ▶ How to recombine the shapes to make the object
- ▶ Use a Self-Organizing Map (2D)
- ▶ Five values don't include relational information
  - ▶ Map them in pairs
  - ▶  $combi_{ij} = (s_1^i, \dots, s_5^i, s_1^j, \dots, s_5^j, o_{ij}, r_{ij}, d_{ij})$
- ▶ Results in a 3D landscape

# Steps

1. Extract Shapes
2. Represent Shapes
3. Interpret Shapes
4. Recombine Shapes
5. Recognize Object
  - ▶ A final neural network to determine the object

## Categorizing the Object

- ▶ Somehow read the SOM
- ▶ Solution: Another neural network
- ▶ One input neuron for every node on the SOM
- ▶ How to deal with  $N$  objects?

## Quick Recap

1. Extract Shapes
  - ▶ Shapes in the form of pictures
2. Represent Shapes
  - ▶ Shape descriptors
3. Interpret Shapes
  - ▶ Five distance values
4. Recombine Shapes
  - ▶ Combinations on a SOM
5. Recognize Object
  - ▶ Categorize the structure

# Experiments

- ▶ Three different tests (with 70, 10, 10 categories)
- ▶ N-Fold testing, for mean-worthy results



# Results

- ▶ Results on shape recognition very good
  - ▶ Up to 65% on 70 different shapes
  - ▶ Especially Orientation, Pixel and Radial Representation
- ▶ Results on object recognition good
  - ▶ All done with radial representation
  - ▶ 75% on cards, 63% on 3D objects
  - ▶ Clearly learns to recognize

# In Conclusion

- ▶ Results show clear recognition rates
  - ▶ Over 65% on 70 shapes
  - ▶ Almost 65% on 10 3D objects
- ▶ Still much room for further work
  - ▶ More different objects
  - ▶ More objects in one image
  - ▶ Comparing to other techniques

# Scene Recognition

- ▶ There are lots of different scenes
  - ▶ Division into: Beach, Mountains, City, Inside, Road, etc.
  - ▶ Division into: Buses, Flowers, Horses, all in natural surroundings
  - ▶ Division into: nearby room A, in corridor B, before printer
- ▶ The task requires to construct features and a similarity metric
- ▶ The goal is to have a large similarity between similar scenes, or a large distance between different scenes

# CBIR

- ▶ CBIR stands for content based image retrieval
- ▶ The goal is to retrieve similar images as a query image
- ▶ The problem is that the semantics of a scene are too complicated to handle
- ▶ Therefore CBIR methods often use two steps:
  1. Extract features of the query image
  2. Compare the features with features of other images using a distance function

## Some Features

- ▶ Color Histogram

$$H_I(c_i) = \frac{\sum_x \sum_y \text{color}(I_{x,y}) = c_i}{|I|} \quad (1)$$

- ▶ Color Histogram Entropy
- ▶ Intensity Histogram
- ▶ Edge Histogram: directions of edges
- ▶ Color Correlogram

$$C^{\delta\varphi}(c_i, c_j) = \mathbf{P}(\text{color}(I_{x,y}) = c_i \wedge \text{color}(I_{x',y'}) = c_j \\ \wedge d((x, y), (x', y')) = (\delta, \varphi))$$

- ▶ Use features to compare images using a distance function
- ▶ Manhattan ( $L_1$ ) distance function:

$$d(x, y) = \sum_i |x_i - y_i|$$

- ▶ Euclidean distance  $L_2$ :

$$d(x, y) = \sum_i (x_i - y_i)^2$$

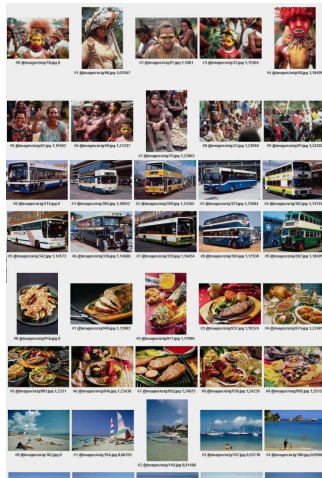
- ▶ Compute distances between a query image and all other images, then compute rank of each image

# CIRE

- ▶ We have extended previous approaches using regions.
- ▶ Each region is processed with a color or edge histogram
- ▶ Then the histograms are separately clustered
- ▶ The cluster indices are used to make cluster correlograms



# Query results of CIRE



## Overall results of CIRE

- ▶ Analyze percentage of returned images of same category when retrieving the  $N$  closest images.
- ▶ There are 10 categories each one consisting of 100 images

	100 queries per category				
	10	20	30	40	50
Africans	0.75	0.70	0.65	0.63	0.60
Beaches	0.69	0.63	0.58	0.55	0.52
Buildings	0.66	0.56	0.51	0.46	0.43
Buses	0.94	0.91	0.89	0.87	0.84
Dinosaurs	1.00	1.00	1.00	1.00	1.00
Elephants	0.72	0.60	0.54	0.49	0.45
Flowers	0.96	0.93	0.91	0.89	0.87
Horses	0.96	0.92	0.90	0.88	0.84
Mountains	0.72	0.66	0.62	0.59	0.57
Foods	0.84	0.78	0.75	0.71	0.66
Total	0.82	0.77	0.74	0.71	0.68

## Results of MPEG 7 features

	100 queries per category				
	10	20	30	40	50
Africans	0.68	0.57	0.51	0.46	0.42
Beaches	0.52	0.44	0.40	0.37	0.35
Buildings	0.49	0.41	0.37	0.35	0.33
Buses	0.68	0.62	0.57	0.53	0.50
Dinosaurs	1.00	1.00	0.99	0.99	0.91
Elephants	0.58	0.44	0.37	0.34	0.30
Flowers	0.88	0.81	0.78	0.74	0.70
Horses	0.86	0.78	0.72	0.67	0.63
Mountains	0.47	0.40	0.34	0.32	0.29
Foods	0.58	0.49	0.44	0.40	0.38
Total	0.67	0.60	0.55	0.52	0.48

## Results of Color Correlogram

	100 queries per category				
	10	20	30	40	50
Africans	0.78	0.69	0.65	0.62	0.59
Beaches	0.53	0.44	0.41	0.38	0.36
Buildings	0.63	0.55	0.49	0.45	0.42
Buses	0.69	0.61	0.57	0.55	0.52
Dinosaurs	1.00	1.00	1.00	1.00	1.00
Elephants	0.68	0.55	0.49	0.44	0.41
Flowers	0.89	0.79	0.74	0.70	0.66
Horses	0.97	0.92	0.88	0.84	0.79
Mountains	0.47	0.41	0.38	0.36	0.35
Foods	0.77	0.70	0.65	0.61	0.57
Total	0.74	0.67	0.63	0.60	0.57

## Extensions of CIRE

- ▶ Currently CIRE only uses color and texture features, but no shape information
- ▶ We want to use segmentation to separate objects and background in an image
- ▶ Then the shapes are used to construct shape features
- ▶ We also want to use relational shape features such as: shape A is close to shape B
- ▶ Extend CIRE with wavelets and principal component analysis for describing textures and shapes

## Final Goals

- ▶ The goal of our project is to use vision techniques on robots
- ▶ Extend particular SLAM algorithms such as DP-SLAM and FastSlam to work with visual descriptors
- ▶ Building a map using distance sensors is much easier
- ▶ Similar to object avoidance: an easy task with distance sensors, a less trivial task with camera's
- ▶ The good thing of using vision is that its range is far, it is less expensive than LRFs and more accurate than Sonars
- ▶ Furthermore, with vision we can do many different things, such as approaching an object of a particular color, etc.

# Questions

