Smart Cameras

as Embedded Systems

Prajakta S. Apte

Smart Camera?

Image Capture + Application Specific Information + Event Description + Making Decisions

Applications

- Human and Animal Detection
- Surveillance
- Motion Analysis
- Facial Identification

Advantages

- Low Power
- Low Cost

Major Focus – Real Time Analysis

Detecting Human Movement and Gestures

Detect Movement and Recognize Human Actions

→ Walk, Stand, Wave

Processing at Two Levels

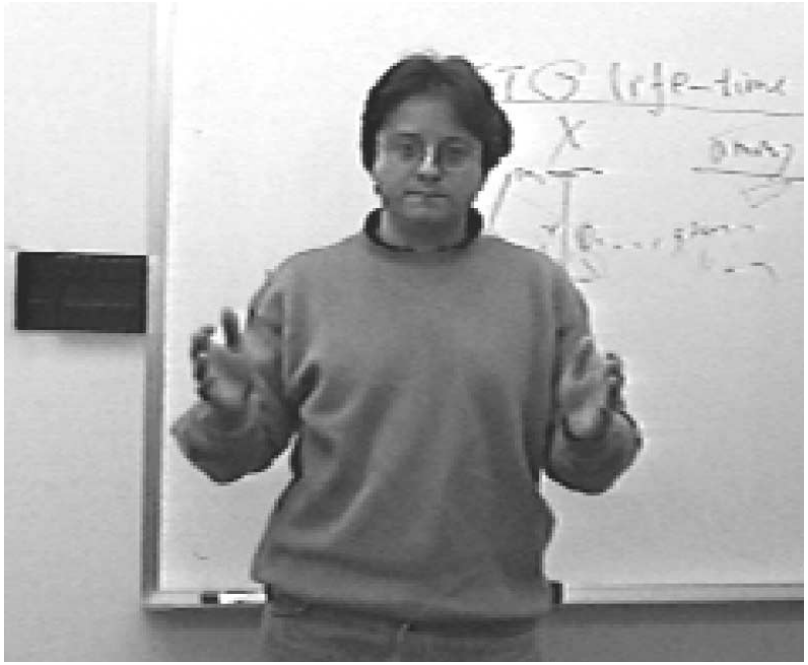
Low Level Processing → Identify Body Parts

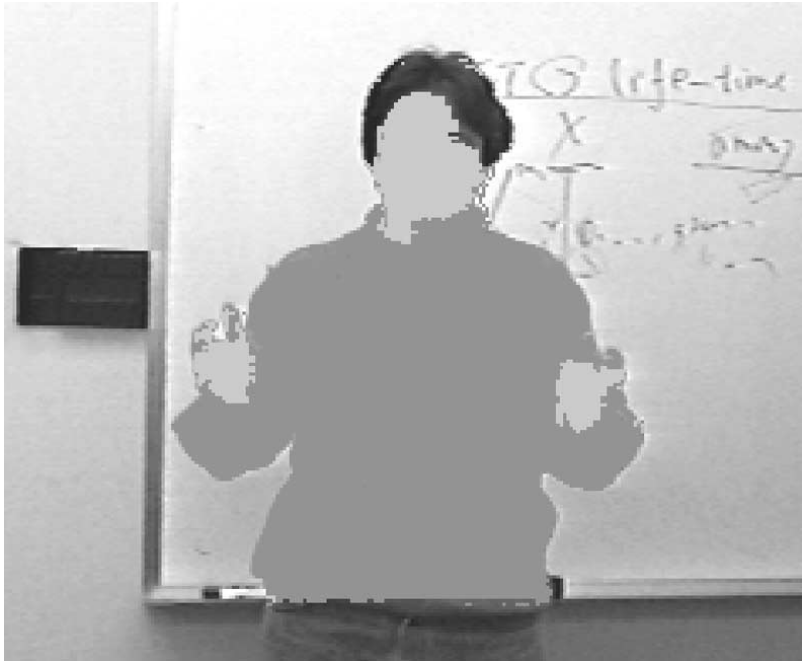
→ Categorize Movement

High Level Processing → Gestures

→ Overall Activity (Based on Scenario)

Low Level Processing – Region Extraction





Input → Video (MPEG/ JPEG, Compressed / Uncompressed)

Transform Pixel to Bitmap

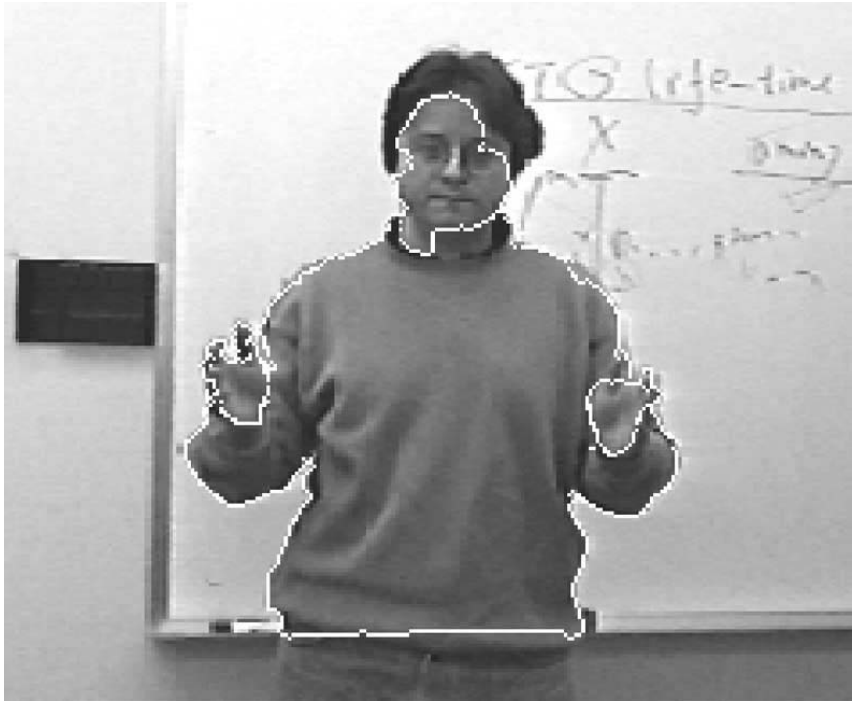
Remove Background

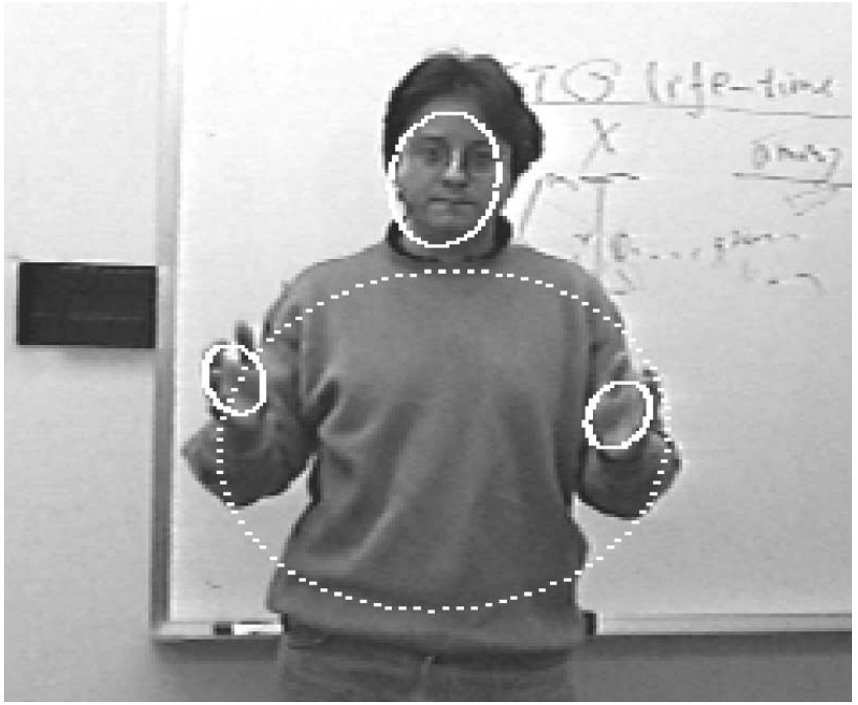
Detect Skin → YUV Color Model

Segment Regions

Combine Meaningfully

Contour Following and Ellipse Fitting





Contour Following

- Link Pixels into Contours
- Use Filter

Ellipse Fitting

- Correct Deformation
- Fit Ellipses to Pixel Regions

Graph Matching

Extracted Region Modeled with Ellipses Corresponds to a Node in a Graphical Representation of the Human Body

- Uses Bayes Classifier to Compute Feature Vectors
- Matches Meaningfully to Body Parts
- Begins Matching With Face to Increase Speed

Low Level Processing is Common to all Systems

High Level Processing needs to be Adapted

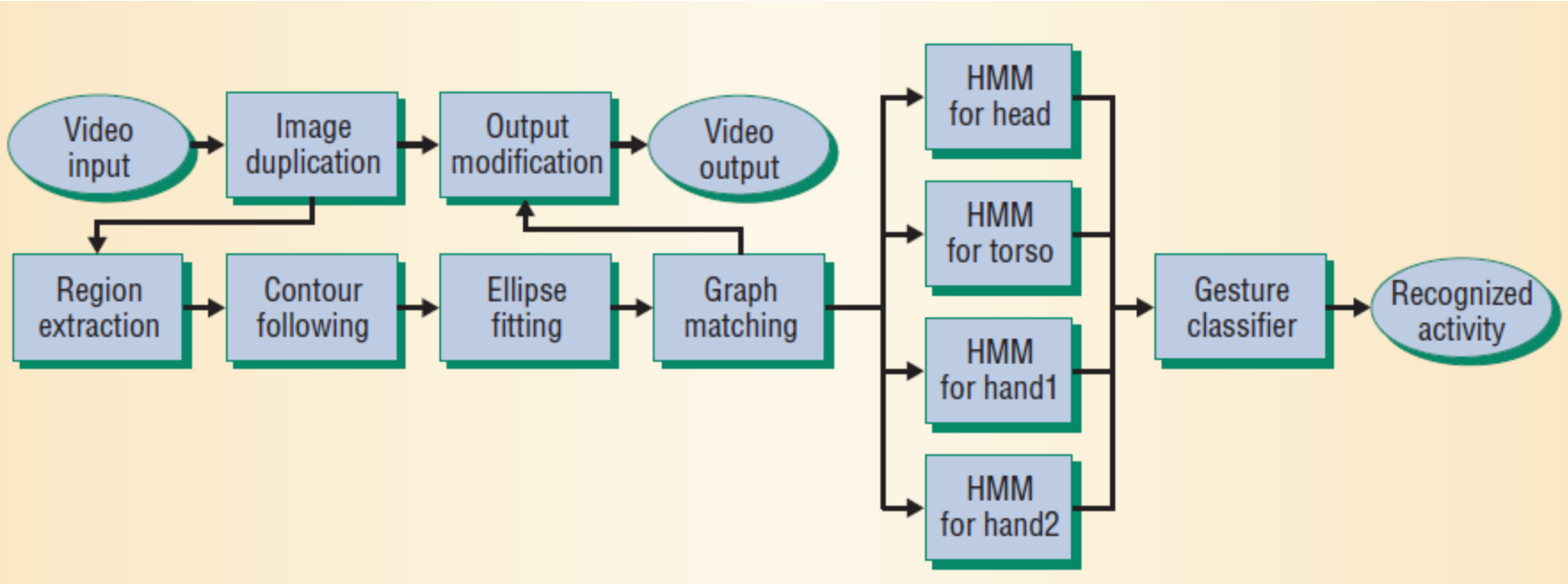
High Level Processing

Use Hidden Markov Models

→ System is a Markov Process (Unobserved States)

→Compare Motion Pattern of Body Part in Sequence of Feature Vectors in Space and Time

→ Evaluate Overall Activity Using Known Gesture



Finding Patterns

Discrete HMMs to Generate 8 Directional Code Words

→Check the Up, Down, Left, Right, and Circular Movement of Each Body Part

Current Motion Pattern Combined With Old Pattern

New Pattern Probability Calculated

Old & New Probabilities Compared to Identify Actions

Gaps Indicate End of Action

Reference Models Generated

Embedded System Development

Used MATLAB To Run Algorithms

→ Much Slower Than Embedded Implementation

No Real-Time Video Processing Possible

Solution: Ported MATLAB to C Code → Runs on VLIW Video processor

Achieved Real-Time Operation?

- Frame Rate
- Latency

Memory Requirement?

Components

Processor: 100-MHz PhilipsTriMedia TM-1300 Video Processor

Features:

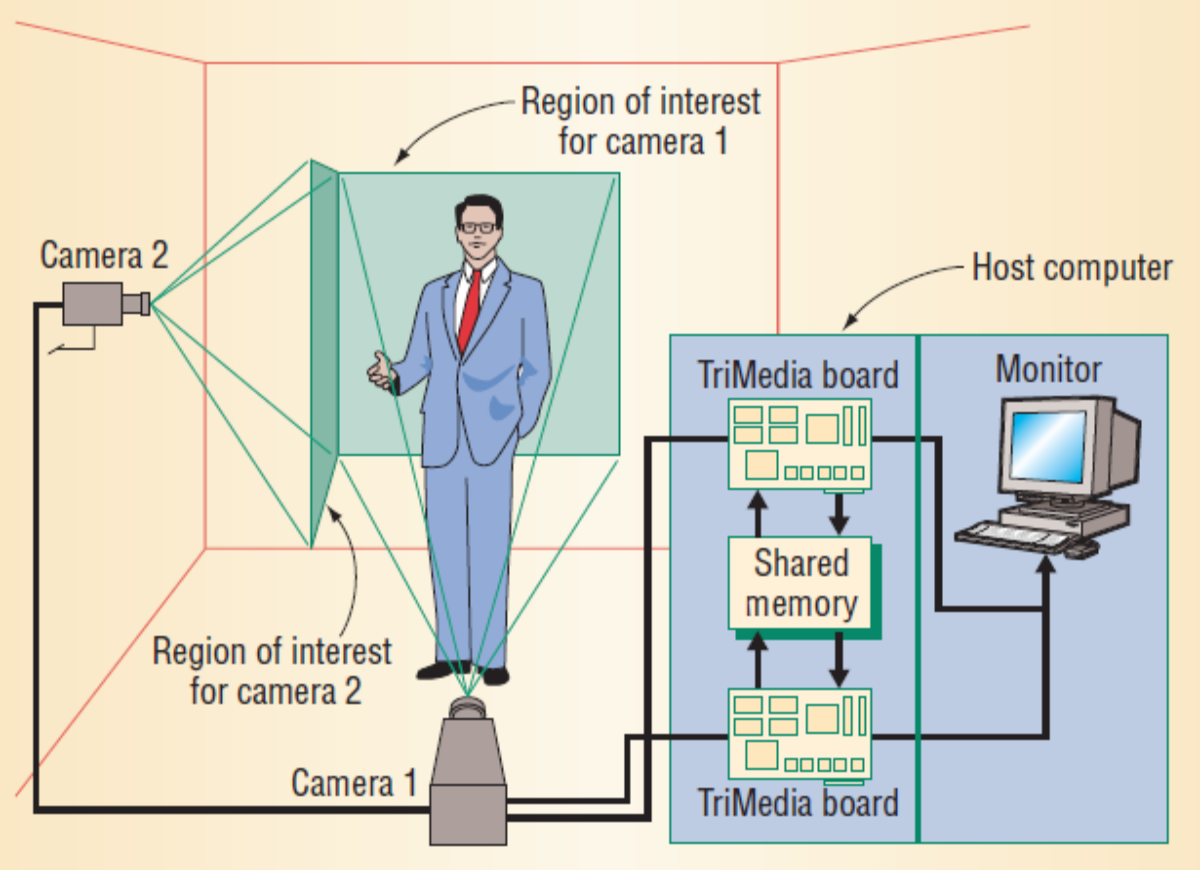
32-bit fixed- and floating-point VLIW processor

Dedicated Image Coprocessor

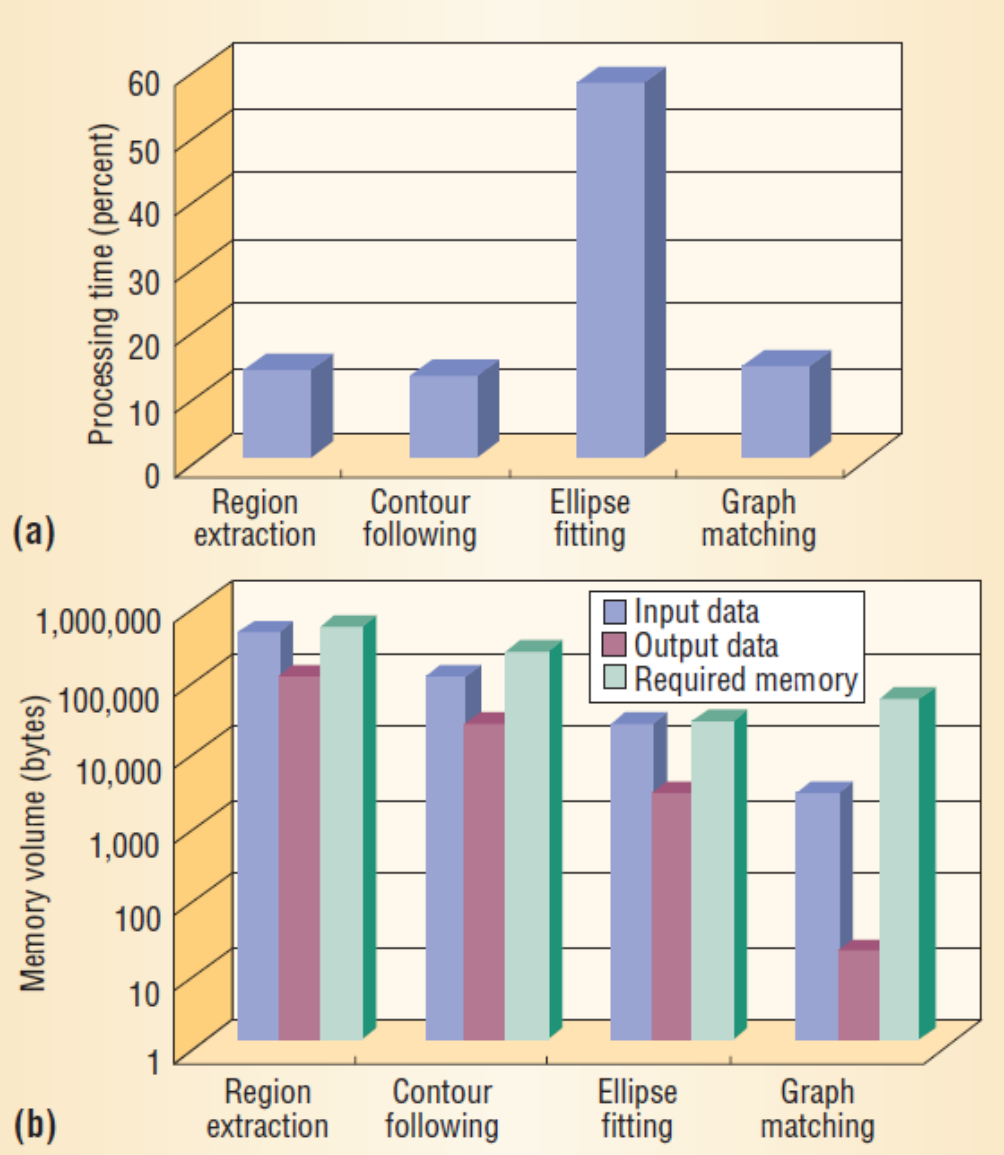
Variable Length Decoder

Optimizing C/C++ Compiler

MPEG, motion JPEG, 2D text and graphics



Experiments and Optimizations



Measured the CPU Times of Low Level Processing Step

Optimized Implementation by Substituting New Algorithms Suited to Real-Time

Used TriMedia Library Routines→ Replaced C-level Code

Levenberg-Marquardt Fitting Rrocedure – Low Execution Time

Contour Following Performed in Parallel

Modifications, Results and Development

Optimization Benefits:

- Improved CPU Performance
- Frame Rate Increased From 5-31 Frames / Second
- Latency Decreased from 340 to 40-60 Milliseconds per Frame

Currently Useful for Some Applications

VLSI System will Enable High Volume Embedded Computing Products

Most Efficient Network Type for Multi-Camera Implementation

Not Determined

Questions?