

Object Tracking based on Adaptive Feature Selection

Maarten Corzilius
Martijn Liem
Vladimir Nedovic
Mark Smids

Supervisors:
Frank Aldershoff
Theo Gevers

Contents

- Goal
- Demo
- System
 - Tracking
 - Image transformation
 - Background subtraction
 - Integration
- Conclusion

Goal

- To study on computational methods to track objects in video sequences.

Goal

- To study on computational methods to track objects in video sequences.

Applied in:

- Surveillance
- Video conferencing
- Man-machine interaction
- Traffic control
- Sport matches
- Scientific analysis

Goal

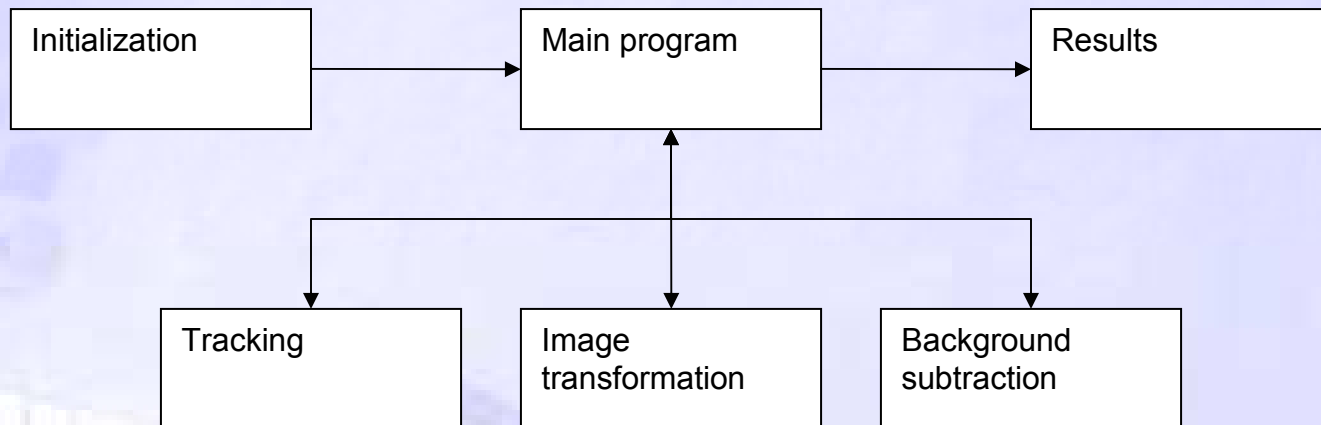
- Build a robust tracker
 - Robust to:
 - Changes in appearance
 - Occlusion
 - Work in real-time
- Applied to manufactured surveillance collection (CAVIAR¹)
 - Added constraint of tracking a single person

¹) <http://homepages.inf.ed.ac.uk/rbf/CAVIAR/>

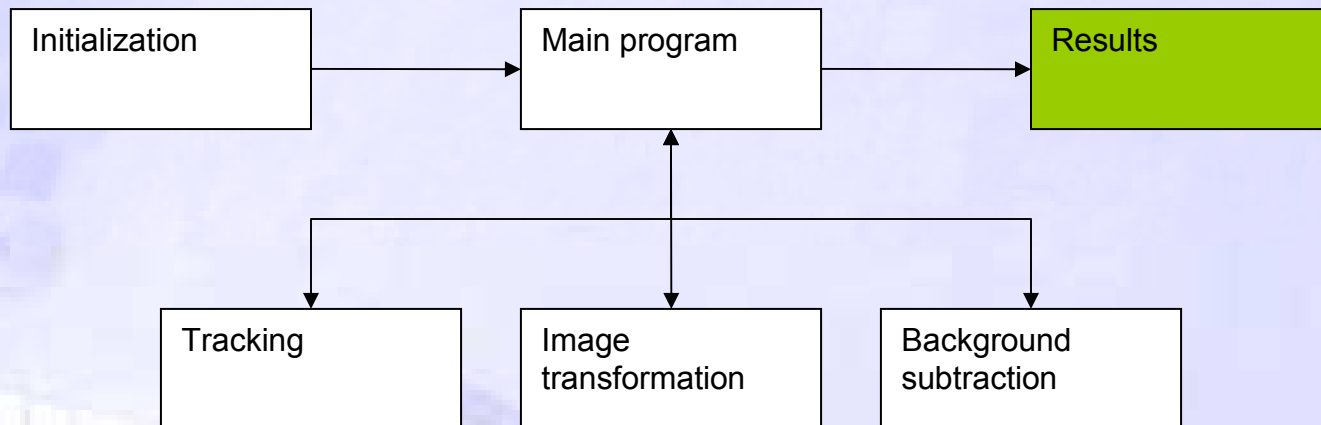
Goal

- Create a background model
 - Detecting objects
 - Transformation of images
- Get trajectories of people
- Warn for suspicious events

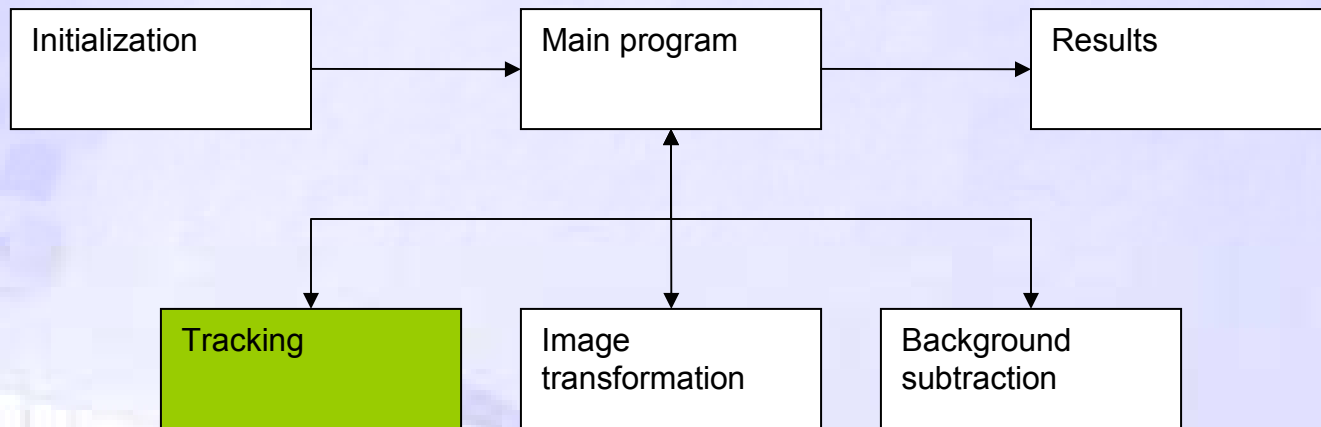
Flow Diagram



Demo



Tracking



Tracking

- Uses features from the frames
 - Color based features
 - R,G,B,r,g,b,H,S,V
 - Texture based features
 - Co-occurrence
 - Gabor filters
 - Wavelet packets

Features - Color

- R,G,B
 - The original color system for input frames
- r,g,b
 - Normalized RGB, robust against changes in object shape and illumination intensity
- H,S,V
 - Hue, saturation, value, additional robustness to highlights

Features – Co-occurrence

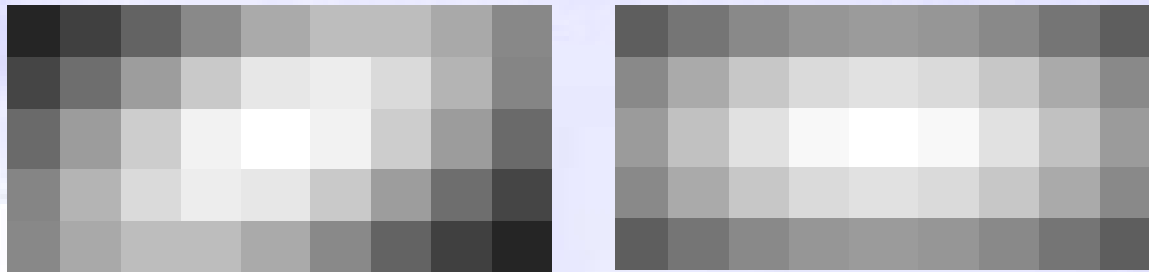
- Co-occurrence
 - Counts the occurrences of colors in some given relation

1	2	2
2	1	3
2	2	3

	1	2	3
1	0	1	1
2	1	2	1
3	0	0	0

Features – Gabor filters

- A Gabor filter is a Gaussian modulated by a complex sinusoid



- Filtering is done by the convolution of the filter with the image

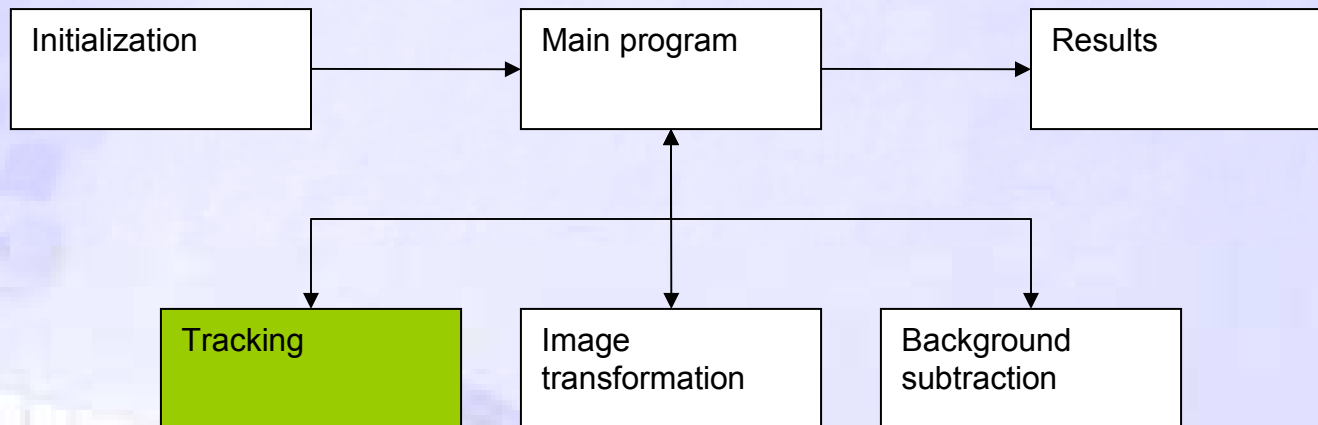
Features – Wavelet packets

- Different textures will have different energy distributions in the frequency domain
- Filter bank can separate the input image in several frequency subbands
- Wavelet packets are such filter banks

Features – Texture



Tracking (continued)

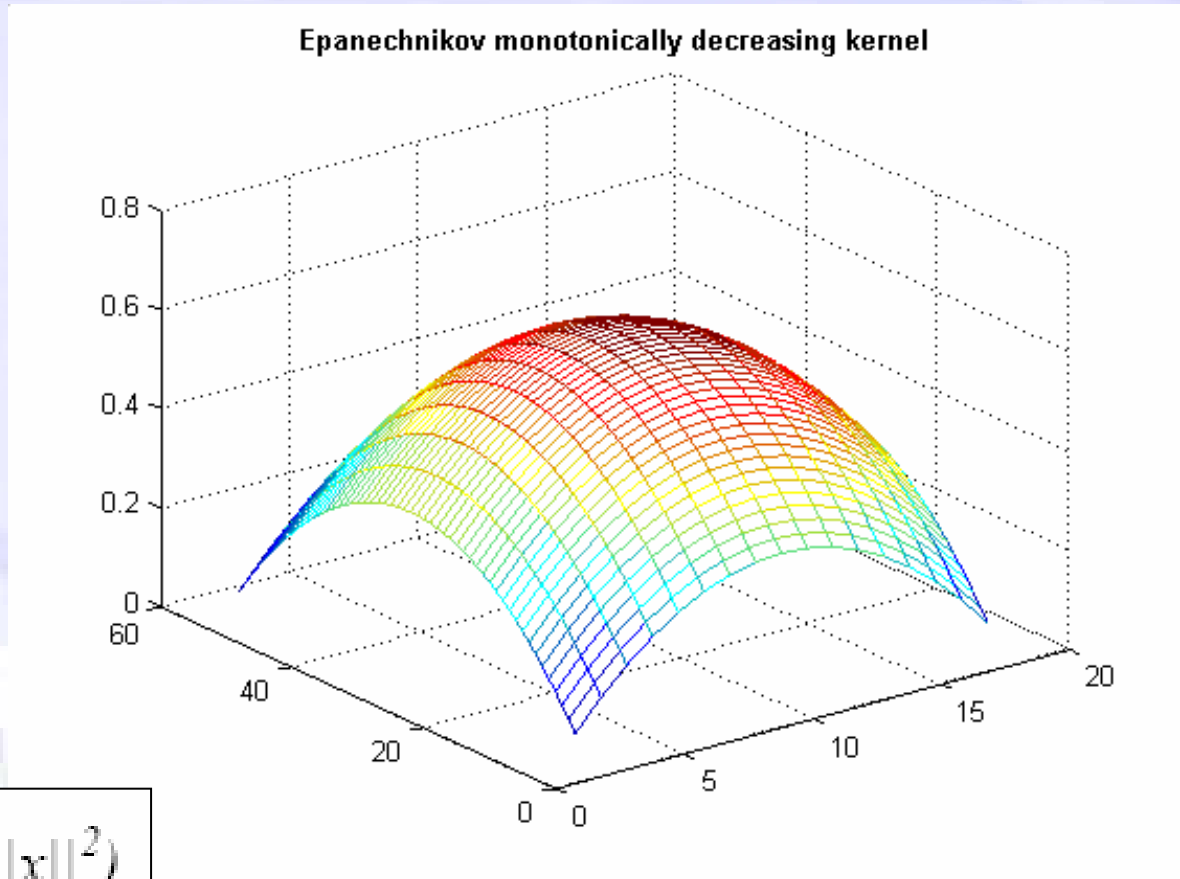


Mean-shift tracker

- Objects of interest characterized by some feature's pdf
 - Features usually color or texture
- Mask the pdf with a monotonically decreasing kernel to give less weight to pixels far from center
 - Those pixels most vulnerable to occlusion and noise
- Mean-shift = non-parametric (i.e. kernel) density estimator
 - Optimize smooth pdf similarity function to obtain the direction of object's movement

Mean-shift tracker (cont'd)

- Kernel:
Epanechnikov,
monotonically
decreasing



$$K_E(x) = \frac{1}{2} c_d^{-1} (d+2) (1 - \|x\|^2)$$

- $cd = \pi$, $d = 2$:

$$K_E(x) = \frac{2}{\pi} (1 - \|x\|^2)$$

Mean-shift tracker (cont'd)

- Similarity function obtained by masking the pdf with a kernel
 - Pdf similarity obtained through Bhattacharyya coefficient:

$$\rho [p(\mathbf{y}), q] = \int \sqrt{p_z(\mathbf{y})q_z} dz$$

- Use function's gradient to determine object's movement
- The algorithm focuses on a small neighborhood instead of performing an exhaustive search

Adaptive feature selection

- Mean-shift uses some color or texture feature, but which one is the best?
- Instead of having a fixed feature, select one out of many “online”
- Evaluate multiple feature spaces while tracking, pick the best one
 - Features that best discriminate object from the background are also the best for tracking

Adaptive feature selection (cont'd)

- Feature selection mechanism based on a two-class (i.e. foreground & background) variance ratio measure
 - Apply measure to a log-likelihood of distributions
 - Augmented variance ratio

$$AVR(F) = \frac{Var(S_F)}{\frac{1}{C} \sum_{i=1..C} \frac{Var_i(S_F)}{\min_{i \neq j} (|mean_i(S_F) - mean_j(S_F)|)}}$$

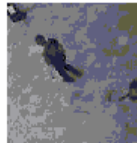
- $C = 2$, $S_F = \text{log-likelihood (L)}$:

$$AVR(L; p, q) \equiv \frac{2 \cdot |\mu_p - \mu_q| \text{var}(L; (p + q))}{[\text{var}(L; p) + \text{var}(L; q)]}$$

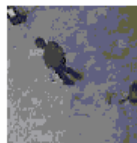
Adaptive feature selection (cont'd)

- 1. calculate features

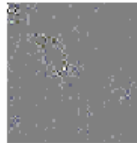
Feat.#1 = R



Feat.#4 = G



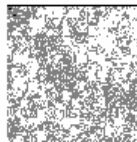
Feat.#7 = wavPackDiag



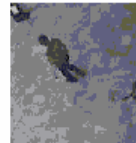
Feat.#10 = r



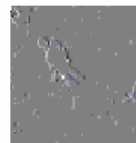
Feat.#13 = cooccurHor45



Feat.#2 = H



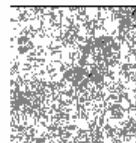
Feat.#5 = wavPackVer



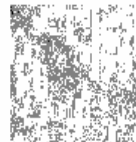
Feat.#8 = V



Feat.#11 = cooccurHor135



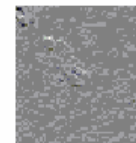
Feat.#14 = cooccurHor0



Feat.#3 = B



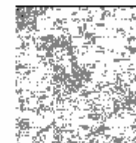
Feat.#6 = wavPackHor



Feat.#9 = Gabor3



Feat.#12 = cooccurHor90

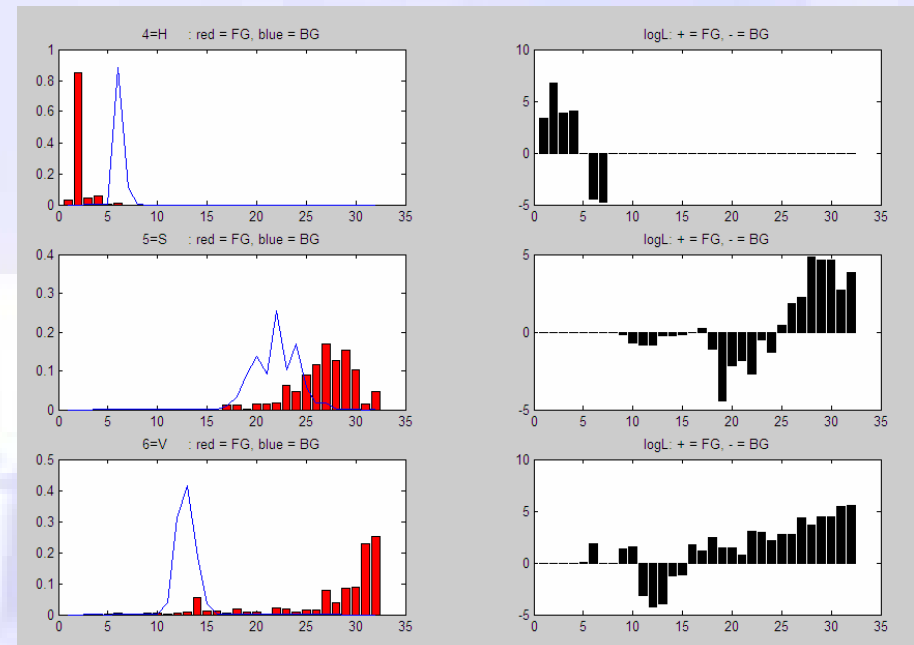
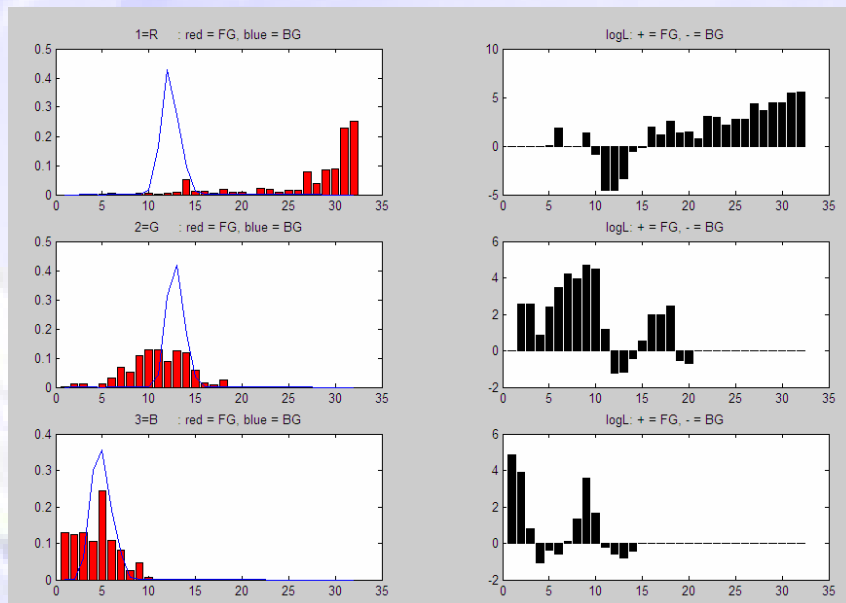


Feat.#15 = Gabor2



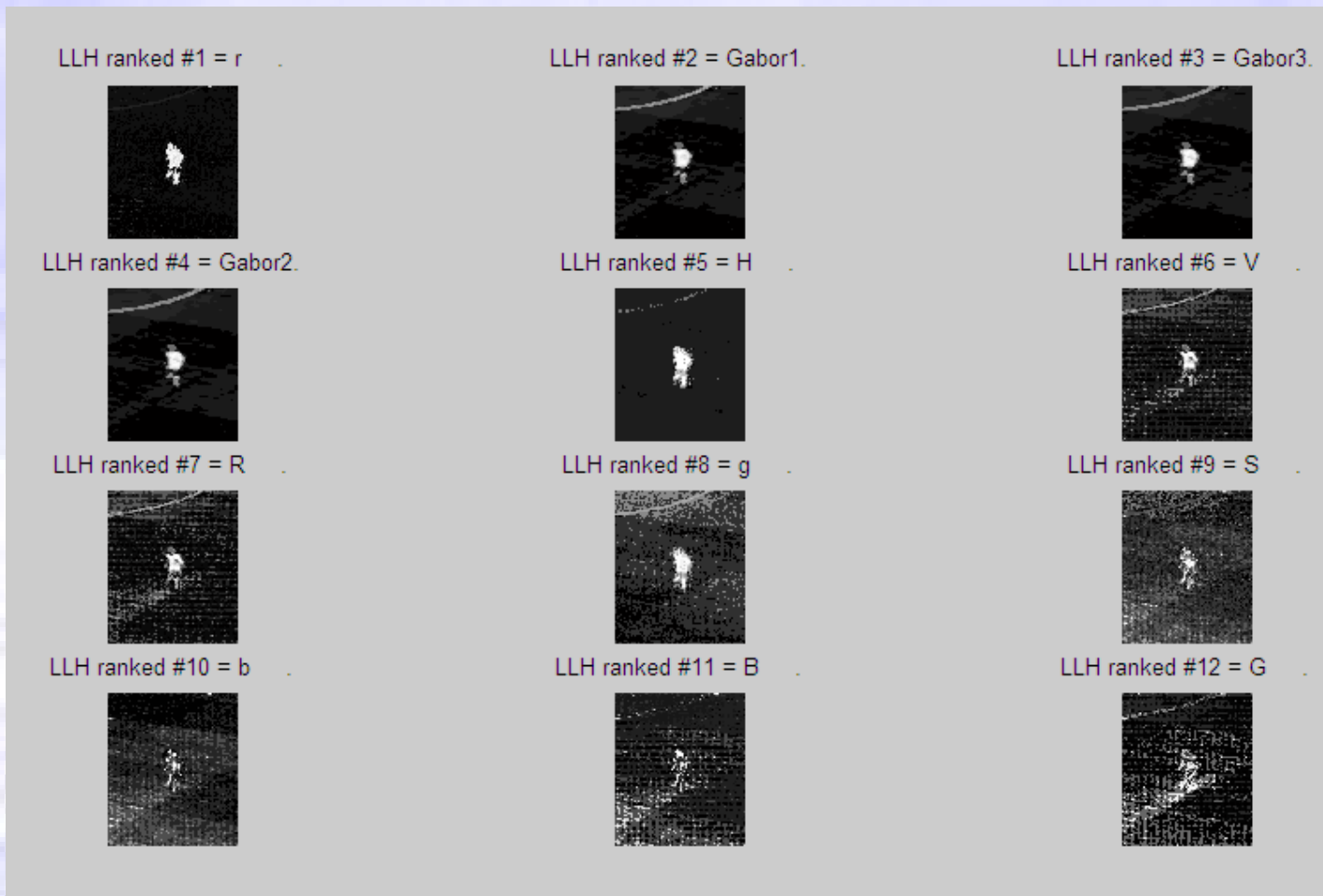
Adaptive feature selection (cont'd)

- 2. calculate likelihood



Adaptive feature selection (cont'd)

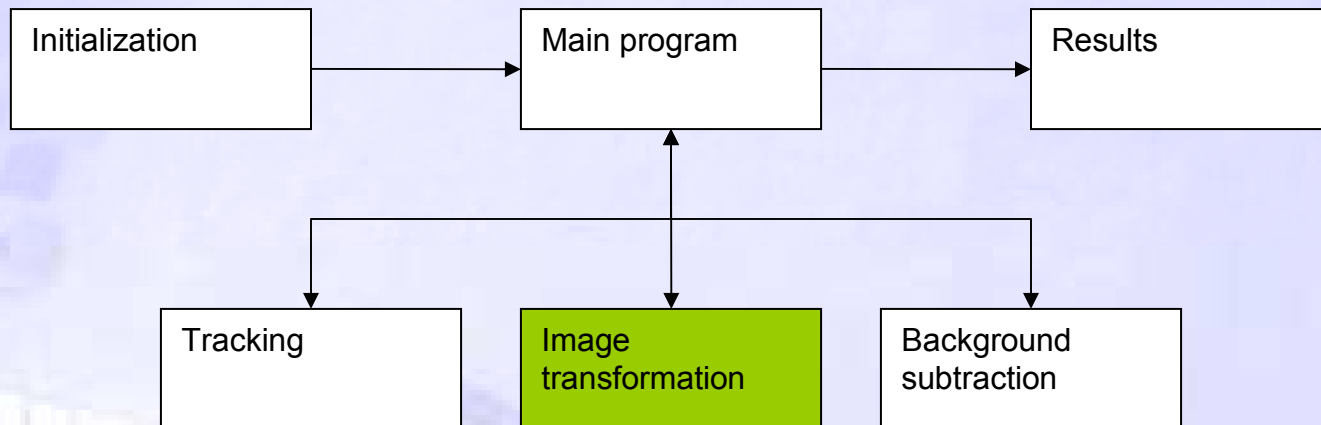
- 3. calculate likelihood images & back-project



Adaptive feature selection (cont'd)

- 4. calculate histograms from likelihood images
- 5. order features based on the AVR of foreground vs. background pdfs
- 6. sort features and pick best one(s)

Fish-eye Lens Correction and Coordinate Transformation



Overview

- Properties of the videoframes
- Removal of fish-eye lens distortion
- Construction of the trajectory model
- Coordinate transformation
- Demo

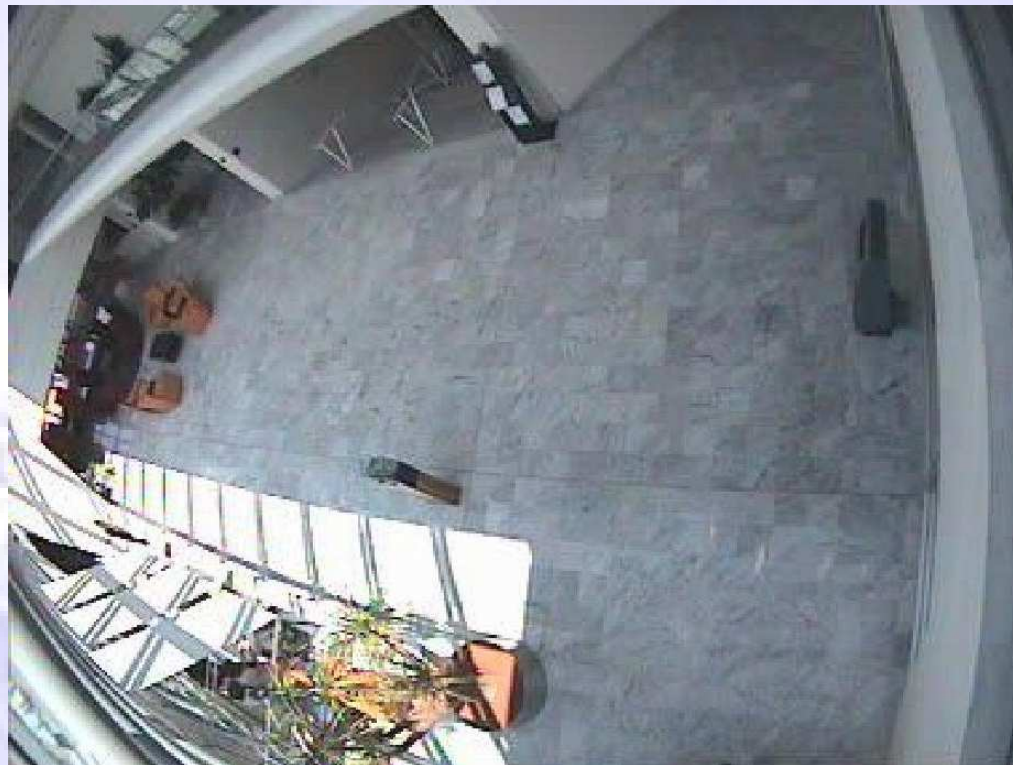
Properties of the videoframes

- Fixed camera with attached Fish-eye lens
 - Advantage: a big view area
 - Disadvantage: distorted image



Properties of the videoframes

- Fixed camera with attached Fish-eye lens
 - Advantage: a big view area
 - Disadvantage: distorted image



Properties of the videoframes

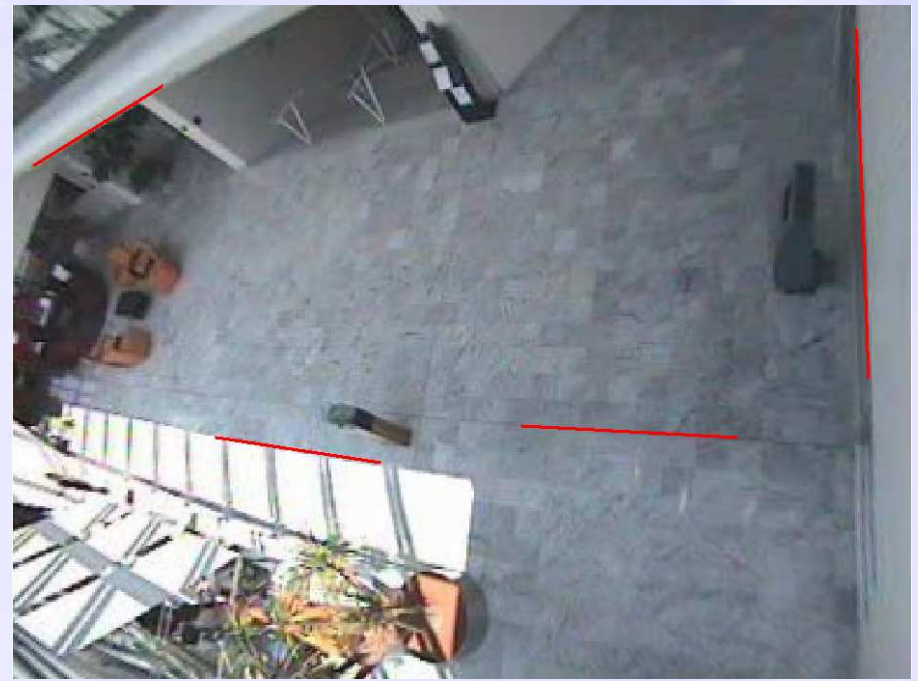
- Other properties of the frames
 - Resolution 384x288
 - Persons in the room simulating actions
 - Slight lighting changes over time

Removal of fish-eye lens distortion

- Radial distortion
- Matlab implementation*
 - `imlenstransform(im, k, cx, cy)`
 - Initially unknown parameter: **k** (radial distortion parameter)

• Peter Kovesei, Matlab function: `imlenstransform`, University of Western Australia, <http://www.csse.uwa.edu.au/~pk>

Removal of fish-eye lens distortion



$K=0.6$

Value for k can be fixed
Computationally expensive!

Construction of the Trajectory model

- Input: undistorted, perspective image
- Given: 3D world coordinates for 4 points in the video frames
 - All points are on the floor ($z=1$)



World coordinates

- (1) 671.5,0
- (2) 670,1116
- (3) 190,1545
- (4) 0,0

Image coordinates

- (1) 152,162
- (2) 323,113
- (3) 503,284
- (4) 112,282

Construction of the Trajectory model

- Transformation performed using:
 - cp2tform
 - imtransform

cp2tform:

Input: world and image coordinate sets, type of transformation: 'projective'

Output: transformation matrix

imtransform:

Input: transformation matrix, image

Output: transformed image

Construction of the Trajectory model

- Result



Coordinate transformation

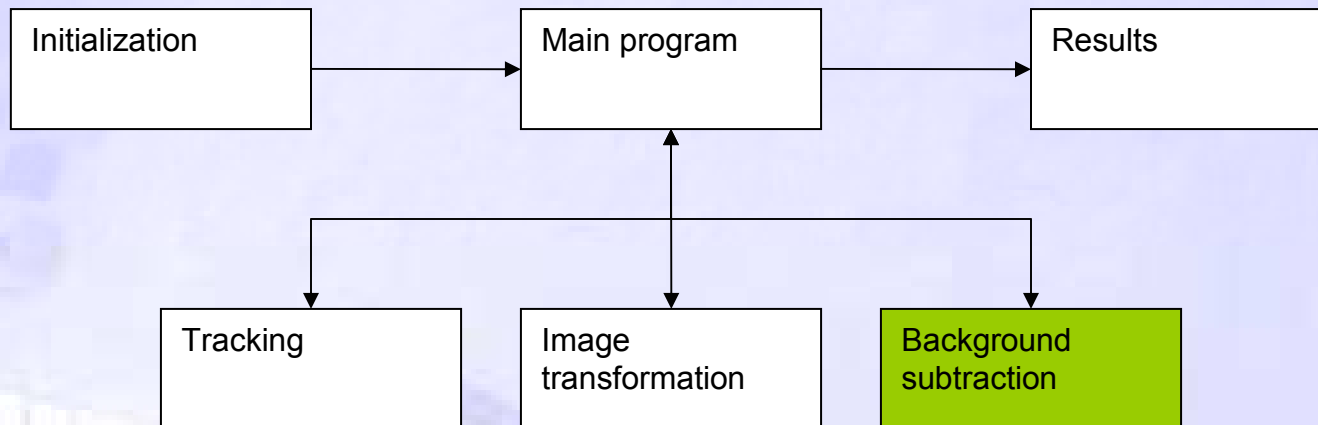
- Transformation of single coordinates
 - Coordinates from the tracker component
- This needs to be fast!
 - Matrix multiplications

Coordinate transformation

- Transformation of single coordinates
 - Coordinates from the tracker component
- This needs to be fast!
 - Matrix multiplications

DEMO

Background Subtraction



Background Subtraction

- The goal is to identify moving objects and notice objects left behind
- When object is left behind, feedback to tracker

Background Subtraction

First Step

- Background subtractor always needs an image of the empty background
- Find empty frames and create background
- Subtract background from image
- Make a mask of the result to get a black/white image. This removes some noise.

Background Subtraction

First Step



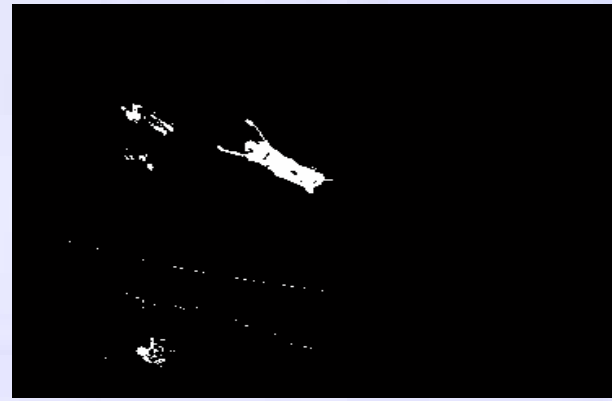
Image



Background



Background subtracted



Masked background

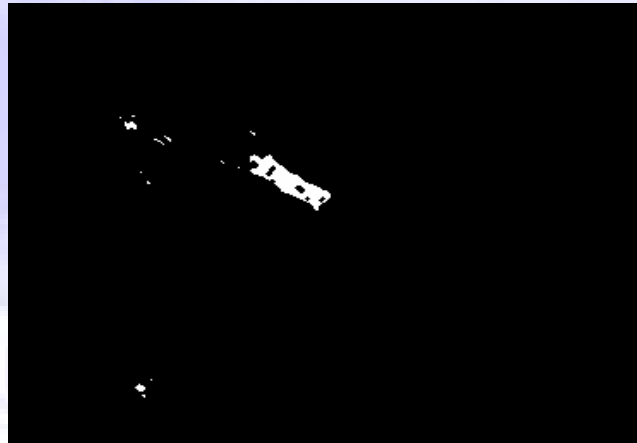
Background Subtraction

Second Step

- Subtraction mask still has some noise which has to be removed
- Use erosion, than dilation to remove the noise from the subtracted image

Background Subtraction

Second Step



Eroded image



Dilated image

Third Step

- Count the number of objects (blobs)
- Compare the number and locations with previous frame to identify merged objects
- Merged objects should be re-identified as separate ones

Third Step

- Get all current centres
- Compare them to centres in the previous frame
- When the distance is smaller than a certain threshold replace centre by previous ones

Fourth Step

- Detection of non moving objects
 - Keep history of object locations in last couple of frames
 - Compare locations of first history frame with current frame
 - When distance under a certain threshold detect object as non moving

Fourth Step

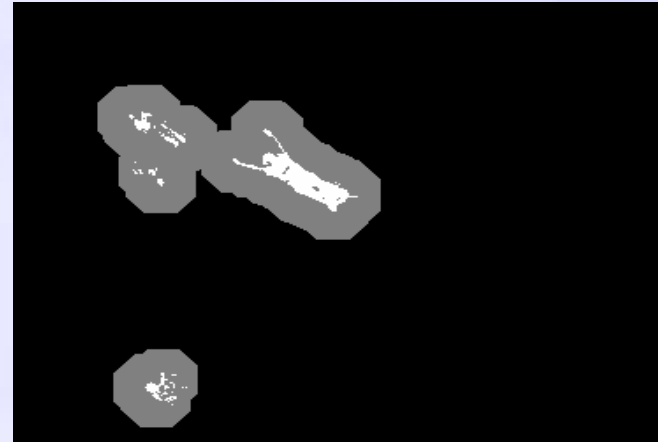
- When such an object has been detected, check with tracker of object is being tracked
 - When it is being tracked move on
 - If not, ring all emergency bells. Something unknown appeared in the area. It's probably a bomb!

Background Subtraction

Display of objects

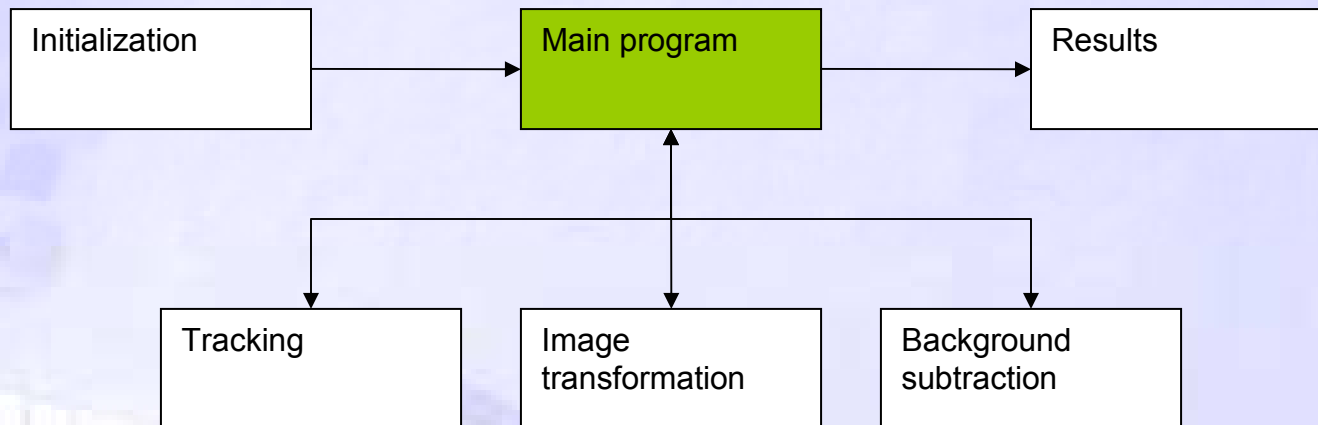


Dilated image multiplied by the subtracted image to get rid of all noise and still keep object contour



Dilated image and contour image added up for nice display

Main Program



Integration

- At last everything has to be combined
- Using a main loop all separate parts are called
- Then the final result is displayed

Integration

- Main loop
 - creates transformations
 - Does background subtraction
 - Does tracking
 - Handles feedback between tracker and background subtractor
 - For each frame

Conclusion

- The system is working
 - Robust against change in appearance
 - Robustness against occlusion not tested
 - Not real-time, but it is feasible
- Future work



Questions