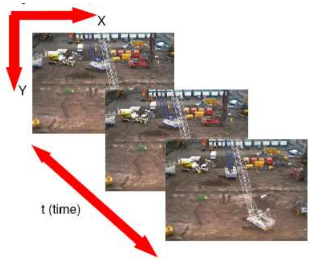
THE TRACKING PROBLEM

Given a sequence of N images, is it possible to:



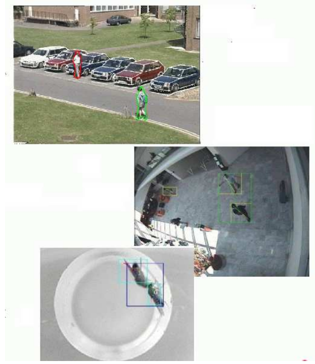
- Identify moving objects
- Predict their position in the next image

Goal: a sequence of tracked positions (r, c) for each target as it moves across the image

Data: a sequence of images (ie. a video)

MOTIVATION

- Objects: sign language recognition, vehicle monitoring
- People: overcrowding, sports, exclusion zones
- Animals: behaviour, health monitoring



TARGET TRACKING WITH NOISE AND BOUNCING

PROBLEM: track a ball falling and bouncing

SEE: `homepages.inf.ed.ac.uk/rbf/...`
`...AVAUDIO/AUDIO8/demo.html`

THE TARGET

PLAN:

1. Removal of irrelevant background + **detection of changes**
2. Tracking noisy motion with **Kalman filter**
3. Coping with events and noise with **condensation tracking**

Issues & Constraints

+ Constant background
+ Color difference with background: Realistic for controlled environments, less realistic for public places: plazas, streets, shopping areas
+ Newtonian motion model

Problems: Motion blur & the bounce

Why a ball?

- Ball bounce (direction, magnitude) is hard to model without precise knowledge of mass, forces, elasticity
- Prediction of $n + 1$ position using first n frames
- Simple shape allows us to concentrate on tracking issues without 3D shape problems

BALL DETECTION CODE

```
% sub background & select pixels with a big difference
fore = (abs(Imwork(:,:,1)-Imback(:,:,1)) > 10) ...
   | (abs(Imwork(:,:,2) - Imback(:,:,2)) > 10) ...
   | (abs(Imwork(:,:,3) - Imback(:,:,3)) > 10);

% erode to remove small noise
foremm = bwmorph(fore,'erode',2);

% select largest object
labeled = bwlabel(foremm,4);
stats = regionprops(labeled,['basic']);
[N,W] = size(stats);

% do bubble sort (large to small) on regions in case
% there are more than 1
```



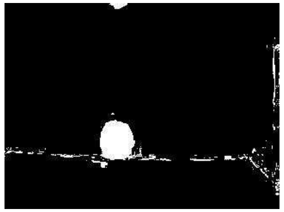
```
for i = 1 : N
  id(i) = i;
end
for i = 1 : N-1
  for j = i+1 : N
    if stats(i).Area < stats(j).Area
      tmp = stats(i);
      stats(i) = stats(j);
      stats(j) = tmp;
      tmp = id(i);
      id(i) = id(j);
      id(j) = tmp;

% get center of mass and radius of largest
centroid = stats(1).Centroid;
radius = sqrt(stats(1).Area/pi);
```

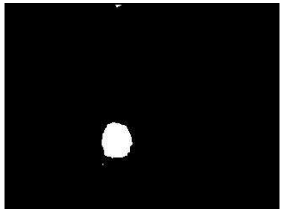
DETECTION RESULTS

INPUT

BACKGROUND REMOVED



CLEANED



DETECTED

What's wrong?

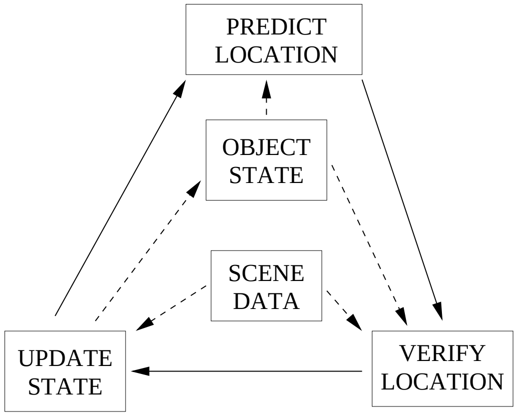
- Moving ball blurred
- Noisy observations
- Potentially poor contrast

We done have:

- Track of positions for ball in frames $0 \ldots N$
- Ability to predict position in frame $N + 1$

So: incorporate motion model in tracker

Architecture of Model-Based Tracker



Incorporates a motion model, eg. ball accelerating due to gravity

Model based Tracking: Kalman filter

Why? Model can be used to

1. Predict likely position, thus reducing search
2. Integrate noisy observations, thus giving improved estimates

What's in model (here called **state**): position, velocity, shape, ...

KALMAN FILTER INTRODUCTION

" A set of mathematical equations that provides an efficient computational (recursive) solution to the least-squares method." [Welch & Bishop]

Most commonly used position estimator used in tracking problems

School of Informatics, University of Edinburgh

KALMAN FILTER THEORY

Assumes:

1. A changing **state** (situation) vector: $\vec{x}_t$ and its associated covariance matrix $\mathbf{A}_t$
2. A **process model** that updates the state over time:

$$\vec{x}_t = \mathbf{A}\vec{x}_{t-1} + \mathbf{B}\vec{u}_{t-1} + \vec{w}_{t-1}$$

where:

- $\mathbf{A}$ - updates the state
- $\mathbf{B}\vec{u}$ - some external control of the state
- $\vec{w}$ - process noise: multi-variate normal distribution, mean $\vec{0}$ and covariance $\mathbf{Q}$

School of Informatics, University of Edinburgh

3. An **observation model** that relates measured data $\vec{z}_t$ to the current state:

$$\vec{z}_t = \mathbf{H}\vec{x}_t + \vec{v}_t$$

where:

- $\mathbf{H}$ - extracts observations
- $\vec{v}$ - observation noise: multi-variate normal distribution, mean $\vec{0}$ and covariance $\mathbf{R}$

School of Informatics, University of Edinburgh

KALMAN FILTER ALGORITHM

1. Predict likely state given what we already know: $\vec{y}_t = \mathbf{A}\vec{x}_{t-1} + \mathbf{B}\vec{u}_{t-1}$
2. Estimate error of predicted state: $\mathbf{E}_t = \mathbf{A}\mathbf{P}_{t-1}\mathbf{A}' + \mathbf{Q}$
3. Estimate correction gain between actual and predicted observations: $\mathbf{K}_t = \mathbf{E}_t\mathbf{H}'(\mathbf{H}\mathbf{E}_t\mathbf{H}' + \mathbf{R})^{-1}$

4. Estimate new state given prediction and correction from observations:
   $\vec{x}_t = \vec{y}_t + \mathbf{K}_t(\vec{z}_t - \mathbf{H}\vec{y}_t)$
5. Estimate error of new state:
   $\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\mathbf{E}_t$

BALL TRACKING WITH THE KALMAN FILTER

Ball physical model:

Position: $\vec{p}_t = (col_t, row_t)$
Velocity: $\vec{v}_t = (velcol_t, velrow_t)$
Position update: $\vec{p}_t = \vec{p}_{t-1} + \vec{v}_{t-1}\Delta t$
Velocity update: $\vec{v}_t = \vec{v}_{t-1} + \vec{a}_{t-1}\Delta t$
Acceleration (gravity down): $\vec{a}_t = (0, g)'$

State vector: $\vec{x}_t = (col_t, row_t, velcol_t, velrow_t)'$
Initial state vector: random

Ball physics update

Prediction: $\vec{y}_t = \mathbf{A}\vec{x}_{t-1} + \mathbf{B}\vec{u}_t$

$$\mathrm{A} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathrm{B}\vec{u}_t = \begin{bmatrix} 0 \\ 0 \\ 0 \\ g\Delta t \end{bmatrix}$$

Use $\Delta t = 1$

Rest of model

Observation process:

$$\mathrm{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Measurement noise:

$$\mathrm{R} = \begin{bmatrix} 0.285 & 0.005 \\ 0.005 & 0.046 \end{bmatrix}$$

System noise: $\mathrm{Q} = 0.01 \times \mathrm{I}$

KALMAN FILTER SUCCESSES

SEE: `homepages.inf.ed.ac.uk/rbf/...`
`...AVAUDIO/AUDIO8/demo.html`

8: 

21: 

35: 

42: 

KALMAN FILTER FAILURES

14: BOUNCE OVERSHOOT 16: SLOW CATCH UP

59: GRAVITY PULLS DOWN AT REST

Kalman filter analysis

- Smooths noisy observations (not so noisy here) to give better estimates
- Could also estimate ball radius
- Could also plot boundary of 95% likelihood of ball position - grows when fit is bad
- Dynamic model doesn't work at bounce & stop

MID-LECTURE QUESTION

HOW MIGHT YOU CORRECT THE TRACKING FAILURE AT THE BOUNCE?

CONDENSATION TRACKING

Conditional **Den**sity Propog**ation**
AKA Particle Filtering

- Keeps multiple hypotheses
- Updates using new data
- Selects hypotheses probabilistically
- Copes with: very noisy data & process state changes
- Tunable computation load

CONDENSATION TRACKING: THEORY

- Maintains set of multiple hypotheses (eg. state vectors, including different models) with estimated probabilities
- Probabilistically generates new hypotheses from the set
- Update hypotheses with observed data (Kalman filter)
- Update hypothesis probabilities

CONDENSATION TRACKING THEORY

Given set of N hypotheses at time $t-1$
$\mathcal{H}_{t-1} = \{\vec{x}_{1,t-1}, \vec{x}_{2,t-1}, \ldots \vec{x}_{N,t-1}\}$ with associated probabilities $\{p(\vec{x}_{1,t-1}), p(\vec{x}_{2,t-1}), \ldots p(\vec{x}_{N,t-1})\}$

Repeat N times to generate $\mathcal{H}_t$:

1. Randomly select a hypothesis $\vec{x}_{k,t-1}$ from $\mathcal{H}_{t-1}$ with probability $p(\vec{x}_{k,t-1})$
2. Generate a new state vector $\vec{s}_{t-1}$ from a distribution centered at $\vec{x}_{k,t-1}$
3. Get new state vector using dynamic model $\vec{x}_t = f(\vec{s}_{t-1})$ and Kalman filter

4. Evaluate probability $p(\vec{z}_t \mid \vec{x}_t)$ of observed data $\vec{z}_t$ given state $\vec{x}_t$
5. Use Bayes rule to get $p(\vec{x}_t \mid \vec{z}_t)$

WHY DOES CONDENSATION TRACKING WORK?

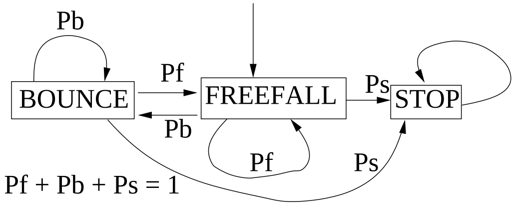
- Many slightly different hypotheses: maybe get one that fits better
- Dynamic model can introduce different effects (eg. state transitions)
- Sampling by probability weeds out bad hypotheses
- Generating by probability introduces corrections

CONDENSATION TRACKING OF BOUNCING BALL

1) Select (N=100 samples) of a ball motion vector by probability of vector

2) Use estimated covariance P() to create state samples $\vec{s}_{t-1}$

3) Situation switching model. $P_b = 0.3$,$P_s = 0.05$



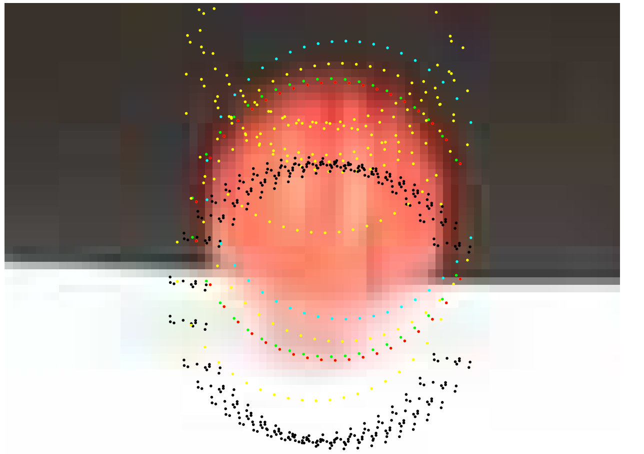
If in STOP situation: zero vertical speed

If in BOUNCE situation: $v_{row} = -0.7 * v_{row}$
Also don't know when bounce was so add some random vertical motion

Then use Kalman filter

4) Estimate hypothesis goodness by
$1/ \mid\mid \mathrm{H}\vec{x}_t - \vec{z}_t \mid\mid^2$
Normalize to estimate hypothesis probability

EXAMPLE OF SAMPLING EFFECTS



Red:final estimate Green:data
Yellow:BOUNCE Blue:STOP Black:FALL

CONDENSATION TRACKING CORE CODE

```
ident: an array of IDCOUNT sample ids. Each id
appears with the same probability as in H_(t-1)
P(): estimated state covariance
x(): state vectors

% generate NCON new samples
for j = 1 : NCON

  k = ident(ceil(IDCOUNT*rand(1))); % get sample
  xc(:) = x(k,time-1,:);            % get state
```



```
  % generate a new SAMPLE at this state
  xc = xc + 5*sqrt(P(j,time-1,:,:))*randn(4);
  if tracksituation(k,time-1)==1 % if in stop sit.
    A,B = ...     % replace A,B for stop model
    xc(4) = 0;    % zero vertical velocity
    tracksituation(j,time)=1;
  else
    r=rand(1);% random number for sit. selection

    if r < pstop    % gone to stop situation
      A,B = ...     % replace A,B for state model
      xc(4) = 0; % zero vertical velocity
      tracksituation(j,time)=1;
```



```
    elseif r < (pbounce + pstop) % bounce sit.
      % add random vertical motion due to
      % imprecision about time of bounce
      xc(2) = xc(2) + 3*abs(xc(4))*(rand(1)-0.5);
      % invert velocity with some loss
      xc(4) = -loss*xc(4);
      tracksituation(j,time)=2;

    else % normal motion
      tracksituation(j,time)=3;

  % update new hypotheses via Kalman filter
  x(j,time,:) = f(xc)
  P(j,time,:,:) = ...
```

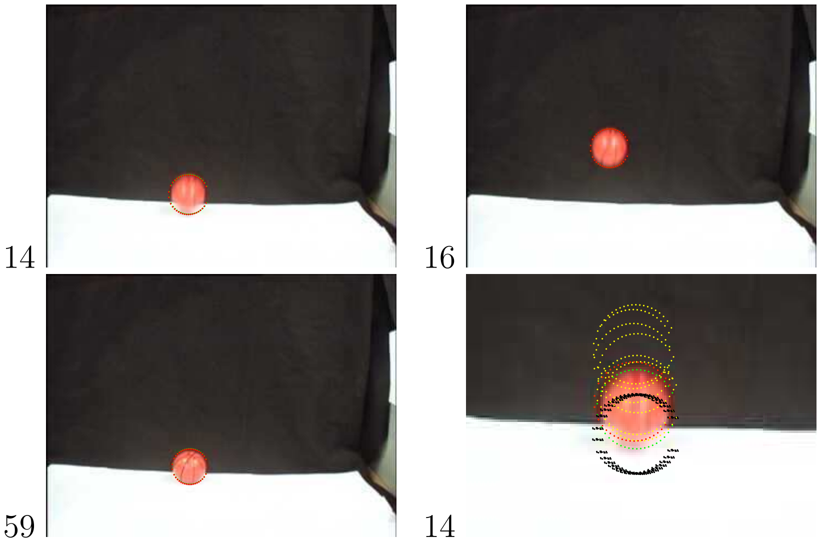


```
  % weight hypothesis by distance from data
  dvec = [cc(time),cr(time)]
            - [x(j,time,1),x(j,time,2)];
  weights(j,time) = 1/(dvec*dvec');

% rescale new hypothesis weights to give sum=1
totalw=sum(weights(:,time)');
weights(:,time)=weights(:,time)/totalw;

% select top hypothesis to draw
subset=weights(:,time);
top = find(subset == max(subset));
```

KALMAN FILTER FAILURES FIXED



N=20 HYPS Y:BOUNCE B:FALL R:FINAL G:DATA

TRACKING IN GENERAL

Can track { people, vehicles, animals } using Kalman filter or condensation tracking

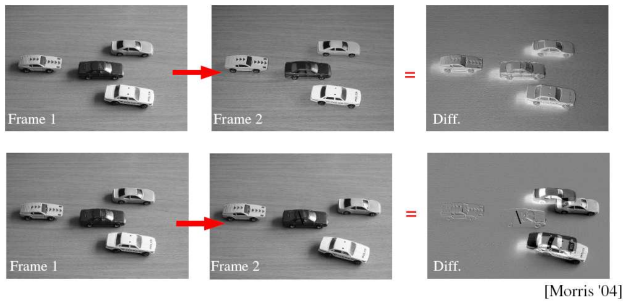
- Need a motion model
- Can learn model, or from calibrated parametric model

Newton's Laws of Motion often used:
$\vec{x}(t) = \vec{s}_0 + t\vec{v}_0 + \frac{1}{2}t^2\vec{a}$

BUT ....

- Still need to know what is being tracked in image
- Easy for bouncing ball scene: contrasting object, plain background
- Hard in real scenes: objects come and go, lighting changes, shadows, moving scene structure (eg. leaves)

TARGET DETECTION BY IMAGE DIFFERENCING



Problems: Illumination changes, overlapping changes, scene vibrations
Solutions: Compare images to pre-learned background image model

ADAPTIVE CHANGE DETECTION

Naive method

$$| \, current - background \, | > threshold$$

doesn't work well in uncontrolled situations

Fix by using:

- Color spaces & shadows
- Kernel density modelling
- Kernel parameter estimation

CHANGE DETECTION ISSUES

If we have a single background, then what about:

- Gradual illumination changes: sun movement
- Rapid illumination changes: lights on
- Background object shadow movement
- Camera jitter
- Halting objects: cars parked

Problem: model out of date
Solution: adapt background model over time

CHROMATICITY COORDINATES

Image: (red,green,blue)=(R,G,B)

Shadows have same color, but are darker

Use chromaticity coordinates

$(r, g, b) = (\frac{R}{R+G+B}, \frac{G}{R+G+B}, \frac{B}{R+G+B})$

Normalizes for lightness

$r + g + b = 1$ so just use (r,g)

SIMILAR FOREGROUND COLORS

In chromaticity space, grey=white=black

Want to detect lightness changes

Lightness: $s = (R + G + B)/3$

Model pixel at time t as (r_t, g_t, s_t)
Model background as (r_B, g_B, s_B)

If $\frac{s_t}{s_B} < \alpha$ or $\frac{s_t}{s_B} > \beta$ or chromaticity different then foreground else background

(Eg. $\alpha = 0.8, \beta = 1.2$)

CHROMATICITY MODELLING

Using average color has problems with scene and camera jitter: no single pixel value

Instead use non-parametric distribution:

$$Pr(x|\text{ BACKGROUND}) = \frac{1}{N}\sum_{i=1}^{N} K_\sigma(x - b_i)$$

b_i : samples from background

Gauss kernel function $K_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{x^2}{2\sigma^2}}$

ADDING COLOR INTO MODEL

Chromaticity coordinates have 2 values: (r, g)

Use $\vec{x} = (r, g)$

$$Pr(\vec{x}|\text{BACKGROUND}) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j\in\{r,g\}} K_\sigma(x_j - b_{ij})$$

ROBUSTLY ESTIMATING KERNEL PARAMETER σ

Different σ for each pixel. use robust estimator:

Assumption: consecutive pixel values usually in same distribution

Use robust estimator for σ, based on $m = median(\{|\ x_t - x_{t+1}\ |\})$

Median gets typical difference due to noise, rather than abrupt scene changes, like due to jitter

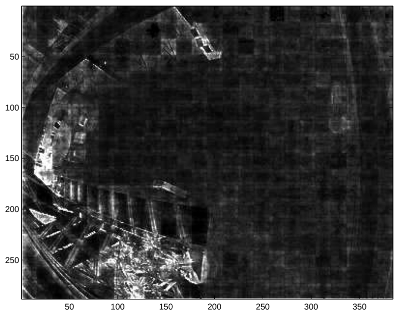
$$\sigma = \frac{m}{0.68\sqrt{2}}$$

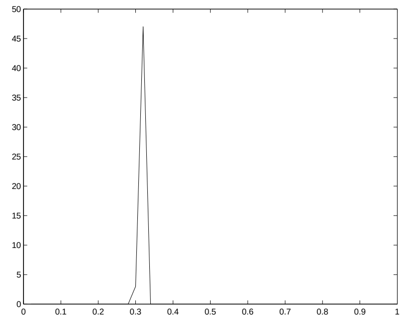
TEST SCENE

BACKGROUND IMAGE TYPICAL IMAGE

DISTRIBUTION RESULTS

σ — 50 SAMPLES Xi AT PIXEL (100,100)





DETECTING CHANGES I

Maintain background history H $= \{\vec{v}_i\} = \{(r_i, g_i, s_i)\}$ for each pixel
H is the last N pixel values classified as background for this pixel
A different set H for each pixel

At time t for a new pixel value $\vec{x}_t = (r_t, g_t, s_t)$, for each $\vec{b}_i = (r_i, g_i, s_i)$ in the background history H for this pixel

If $\alpha \leq \frac{s_t}{s_i} \leq \beta$ record sample in M ($\alpha = 0.8, \beta = 1.2$)

If $\mid M \mid = 0$
then FOREGROUND
else estimate probability of $\vec{x}_t = (r_t, g_t, s_t)$ being background

DETECTING CHANGES II

Want to estimate $Pr(\text{BACKGROUND}|\vec{x}_t)$

$$Pr(\vec{x}_t|\text{BACKGROUND}) = \frac{1}{\mid M \mid} \sum_{i \in M} \prod_{j \in \{r,g\}} K_\sigma(x_j - b_{ij})$$

$$Pr(\text{BG}|\vec{x}_t) = \frac{Pr(\vec{x}_t|\text{BG}) \times Pr(\text{BG})}{Pr(\vec{x}_t|\text{BG}) \times Pr(\text{BG}) + Pr(\vec{x}_t|\text{FG}) \times (1 - Pr(\text{BG}))}$$

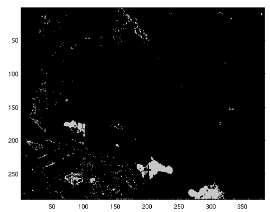
$Pr(\text{BACKGROUND}) = 0.99$ (estimated *a priori* likelihood)
$Pr(\vec{x}_t|\text{FOREGROUND}) = 0.001$ (estimated - all values likely)

If $Pr(\text{BACKGROUND}|\vec{x}_t) < \tau$ then FOREGROUND ($\tau = 0.05$)

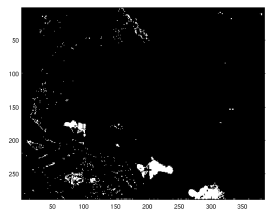
BACKGROUND

TYPICAL IMAGE

PROBABILITY



THRESHOLDED



UPDATING THE MODEL

At each pixel i, keep N most recent (r_t, g_t, s_t) background pixel values

Allows slow drift in illumination
Set allows multiple backgrounds due to jitter

(Discard non-background pixels)

$N = 50$ in examples

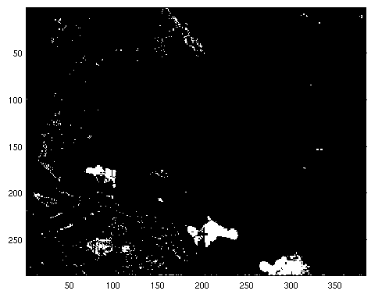
NOISE CLEANING

Final stage: remove noise in thresholded foreground image:

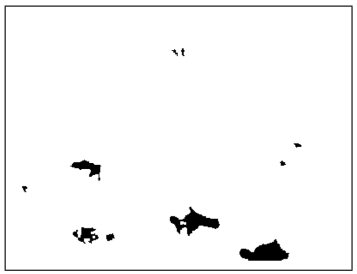
1. Collect into regions by 4-connectedness
2. Remove groups with less than 5 pixels
3. "Close" (dilate and then erode) to fill in gaps
4. Remove resulting groups still with less than 20 pixels

Future: remove groups whose bounding boxes do not overlap another in previous & next frame
Future: Track boxes thru time using Kalman filter

THRESHOLDED



CLEANED



RESULTS URL

SEE: `homepages.inf.ed.ac.uk/rbf/...`
`...AVAUDIO/AUDIO8/demo2.html`

OBSERVATIONS & EXTENSIONS

1. Big model arrays (σ and kernel samples per pixel): 100+ Mb history for 50 observations
2. Rapid illumination changes, eg. lights on: chromaticity ok, lightness not
3. Image compression introduces noise: eg. JPEG artifacts
4. Future: suppress moving groups (eg. moving tree branches)
5. Future: foreground statistical models

SUMMARY

Techniques good for:

1. Change detection by modelling the background statistically
2. Kalman filtering - tracking & hypothesis noise reduction
3. Condensation tracking - multiple undecided hypotheses, situation change

Visual Ethics

Time to ask yourself questions:

1. **Video surveillance:** around prisons? Lothian Road? Corner shops?
2. **Autonomous navigation:** goods delivery in factories? Predator AUVs?
3. **Factory Automation:** cheaper, more reliable goods? unemployment?
4. **Biometrics:** Spot the terrorist? Secure banking?
5. **Car registration plate reading:** Speed control? Police database?