Video

- **Video**
    - Aim is to reliably track moving objects of interest in a scene
        - Motion Detection
        - Moving object detection and location
        - Derivation of 3D object properties
    - **What is an object of interest?**
        - Size - can only track reasonably large objects
        - max/min velocity and acceleration - can't go to fast, can't accelerate too quickly
        - Assumptions:
            - Mutual correspondence of object appearance
            - Common motion - e.g. walking, arms and legs move periodically
    - **Common Problems**
        - Illumination and appearance changes
            - Gradual (time of day), sudden (clouds/lights), shadows, weather (rain/snow)
        - Background changes
            - Objects becoming part of the background
            - Objects leaving the background
            - Background objects oscillating slightly
        - Setup
            - Camera motion
            - Frame rate
            - Field of view
            - Distance to objects
            - Location of camera
    - **Difference images**
        - Image subtraction
            - $$d(i,j) = |f_k(i,j) - b(i,j)|$$ $$d(i,j) = \begin{cases} 0 & \text{if } |f_k(i,j) - b(i,j)| < T \\ 1 & \text{otherwise} \end{cases}$$
                - First formula retains greyscale, second is binary
        - Difference between current frame and background model can highlight objects of interest
            - Colour images - Per channel difference? Just process hue?
        - Won't always be right
            - False positives - incorrect object of interest identification
            - False negatives - incorrect background identification
            - Thresholding
                - Too high = not enough movement detected
                - Too low = too much accepted as objects of interest
            - **Dependent on high contrast between bg model and o.o.i**

- **Background Models**
  - **Static background**
    - Simplest approach
    - Sensitive to threshold T
    - How to get a background model?
      - First frame? Naive
  - **Running average**
    - $$b_{n+1}(i,j) = \alpha . f_n(i,j) + (1-\alpha) . b_n(i,j)$$
      - Where α is the learning rate
    - Adapts to a changing scene
    - **But…**
      - incorporates moving objects
      - Can try to avoid this by using great number of frames, or small learning rate, but makes slow to adapt
  - **Median**
    - Middle value from an ordered list
    - $$h_n(i,j,p) = \sum_{k=(n-m+1)..n} \begin{cases} 1 & \text{... if } (f_k(i,j) = p) \\ 0 & \text{... otherwise} \end{cases}$$
      - This creates a histogram
      - The median value for each pixel over m frames, assuming current frame is n
    - Median value is computed using this histogram
    - Every new frame that we see, the oldest value is discarded and the new one is added
      - **Very expensive** - store each frame being used to compute, as well as a histogram for each pixel
    - Determine a value for m and limit the histogram quantisation
    - **Update the histogram using aging (computationally inexpensive)**
      - $$h_n(i,j,p) = \sum_{k=1..n} \begin{cases} w_k & \text{... if } (f_k(i,j) = p) \\ 0 & \text{... otherwise} \end{cases}$$
        $$\text{where } w_1 = 1 \text{ and } w_k = w_{k-1} * 1.001$$
    - Could do
      - Selective update
      - Mode
        - $b_n(i,j) = p \text{ where } h_n(i,j,p) \geq h_n(i,j,q) \text{ for all } q \neq p$
  - **Gaussian Mixture Model**
      - Approach to dealing with multi-modal background pixels
        - E.g. trees in wind, water etc
      - Model multiple values (3-5) at each point
      - Unsupervised learning
      - **Most popular methods for background modelling**

- **Approach**
    -
        - Model each pixel $f_n(i,j)$ using k Gaussian distributions each wi
            - $\pi_n(i,j,m)$
            - $\mu_n(i,j,m)$
            - $\sigma^2_n(i,j,m)$
        - Weighting
        - Mean
        - Std. Deviation
    - Set a learning constant α (where $0.01 \leq \alpha \leq 0.1$)
    - For each new sample $f_n(i,j)$
    -
        - Select **the best close Gaussian distribution**
            - **close** = within $2.5\ \sigma_n(i,j,m)$ of $\mu_n(i,j,m)$
        - If there is a best close Gaussian l

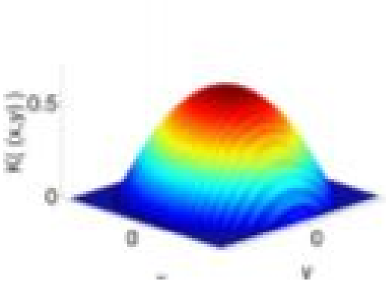
          $\pi_{n+1}(i,j,m) = \alpha * O_n(i,j,m) + (1-\alpha) * \pi_n(i,j,m)$

          where $O_n(i,j,m) = 1$ for **the close Gaussian distribution** and 0 otherwise

          $\mu_{n+1}(i,j,m) = \mu_n(i,j,m) + O_n(i,j,m) * (\alpha / \pi_{n+1}(i,j,m)) * (f_n(i,j)-\mu_n(i,j,m))$

          $\sigma^2_{n+1}(i,j,m) = \sigma^2_{n+1}(i,j,m) + O_n(i,j,m) * (\alpha / \pi_{n+1}(i,j,m)) * ((f_n(i,j)-\mu_n(i,j,m))^2 - \sigma^2_n(i,j,m))$
        - If there is no close Gaussian (replace one...)
            - $x = argmin_m(\pi_n(i,j,m))$
            - $\mu_{n+1}(i,j,x) = f_n(i,j)$
            - $\sigma^2_{n+1}(i,j,x) = 2.max_m\ \sigma^2_n(i,j,m)$
        - When there isn't a similar enough Gaussian, replace the smallest one
    - **Identifying background distributions**
        - Define T, a proportion of frames in which the background pixels should be visible
        - Order the Gaussians by $\pi_{n+1}(i,j,m)/\sigma_{n+1}(i,j,m)$
        - Gaussians 1..B are considered background where
        - $B = argmin_b((\Sigma_{b=1..m}\ \pi_{n+1}(i,j,m)) > T)$
    - Check if best close Gaussian (or the new Gaussian) is a background distribution
    - **Use dilations and erosions to remove small regions and fill holes**
- **Issues with GMM**
    - Fails under fast variations
    - Low sensitivity to Gaussian tails
    - Less frequent events produce low probability & high variance
    - Needs to compute floating point probabilities

- ~~**Codebook**~~
    - ~~Designed to combat shortcomings of GMM~~
        - ~~Models each pixel independently~~
        - ~~For each pixel a codebook is maintained of RGB values (R_i,G_i,B_i)~~
- **Shadow detection**
    - Compare current frame to background image
        - **Intensity/luminance decreases**
            - **Not by too much**
        - **Saturation doesnt increase too much**
        - **Neither does hue**
    - Hue unpredictable and change in luminance can be small
    - $$SP_k(i,j) = \begin{cases} 1 \ldots \text{if } \left(\alpha < \frac{f_k^V(i,j)}{B_k^V(i,j)} < \beta\right) \textit{and} \left(\left(f_k^S(i,j) - B_k^S(i,j)\right) < \tau_S\right) \textit{and} \left(\left|f_k^H(i,j) - B_k^H(i,j)\right| < \tau_H \right. \\ 0 \ldots \text{otherwise} \end{cases}$$
    - $$SP_k(i,j) = \begin{cases} 1 \ldots \text{if } \left(\lambda < \frac{f_k^V(i,j)}{B_k^V(i,j)} < 1.0\right) \textit{and} \left(\left|f_k^S(i,j) - B_k^S(i,j)\right| < \tau_S\right) \\ 0 \ldots \text{otherwise} \end{cases}$$
- **Tracking**
    - Used in surveillance, sports video analysis, car driving systems etc
    - **Difficulties arise as objects:**
        - May be undergoing complex motion
        - May change shape
        - May be occluded
        - May change appearance due to lighting/weather
        - May physically change appearance
    - **Exhaustive Search**
        - Extract object to be tracked from frame
        - Compare in all possible positions in future frame(s)
            - **Using a similarity metric e.g. normalised cross correlation**
        - Requires four degrees of freedom
            - Two for position
                - Change in x and y coordinates
            - One for scale
                - Movement towards camera (increase size), away from camera (decrease size)
            - One for rotation
        - **Can restrict our local maxima search using assumptions about the amount of change allowed between frames**
        - Will fail if motion is too complex
    - **Mean Shift**
        - Track objects by
            - Searching locally for the most similar region
            - Using a histogram to represent the object
            - Using iterative gradient ascent/hill climbing to locate best match

- Histogram of colours, oriented gradients, textures etc.
- Weighting of centres is far greater than the edge colours
- **Modelling the object**
    - Probability density function (histogram) of colours
        - $$\hat{q}_u \triangleq C \sum_{i=1}^{n} k\left(\left\|\frac{\mathbf{x}_i}{h_q}\right\|^2\right) \delta\left[b\left(\mathbf{x}_i\right) - u\right]$$
        - Limiting the number of bins
    - Typically use an elliptical region
    - Weight the values relative to their location - **epanechnikov kernel**
        - 

- **Model candidate regions**
    - $$\hat{p}_u(\mathbf{y}) \triangleq C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h_p}\right\|^2\right) \delta\left[b\left(\mathbf{x}_i\right) - u\right]$$
    - Matching to find the best new location
        - Compare histograms directly
        - Move to the mode in matching space
        - NCC observes the best results - clear spike
- **Mean Shift Approach**
    - Consider the gradient of the similarity function (the Bhattacharyya coefficient)
    - Gradient of superposition of kernels, centred at each data point is equivalent to convolving the data points with the gradient of the kernel
    - $$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\mathbf{y}_0)}} \delta\left[b\left(\mathbf{x}_i\right) - u\right] \qquad \mathbf{y}_1 = \frac{\sum_{i=1}^{n_h} \mathbf{x}_i w_i}{\sum_{i=1}^{n_h} w_i}$$
        - Derived from the Bhattacharyya similarity measure
        - Assumes Epanechnikov’s kernels
        - Moves in direction of highest gradient
    - **Iterate until convergence - separation between y_0 and y_1 < a convergence threshold (**ε**)**
- **Parameters**
    - Number of bits pooer channel, convergence parameter (ε), kernel type
- **Background exclusion**

        - If a background model is available…
            - Favour images which are similar to the object model **&&** dissimilar to the corresponding background region
    - **Multipart model**
        - Histograms lack spatial structure - can use a multipart model
        - Performance improves - **16 is optimal**
- **Dense Optical Flow**
    - Compute a motion field (known as an optical flow) for the entire image
        - **Direction and magnitude**
    - Based on the **brightness constancy constancy constraint**
        - **Object points will have the same brightness over a short period of time**
        - $$f_t(i,j) = f_{t+\Delta t}(i+\Delta i, j+\Delta j)$$
    - Need to find the displacement (Δi,Δj) which will minimise the residual error
        - $$\varepsilon(\Delta i, \Delta j) = \sum_{i=i_{current}-w}^{i_{current}+w} \sum_{j=j_{current}-w}^{j_{current}+w} f_t(i,j) - f_{t+\Delta t}(i+\Delta i, j+\Delta j)$$
    - To compute the optical flow $\left(\frac{\Delta i}{\Delta t}, \frac{\Delta j}{\Delta t}\right)$, assuming that displacement is small…
        - $$f_{t+\Delta t}(i+\Delta i, j+\Delta j) = f_t(i,j) + \frac{\partial f}{\partial i}\Delta i + \frac{\partial f}{\partial j}\Delta j + \frac{\partial f}{\partial t}\Delta t$$
        - **Hence…**
        - $$\frac{\partial f}{\partial i}\Delta i + \frac{\partial f}{\partial j}\Delta j + \frac{\partial f}{\partial t}\Delta t = 0$$

          So

          $$\frac{\partial f}{\partial i}\frac{\Delta i}{\Delta t} + \frac{\partial f}{\partial j}\frac{\Delta j}{\Delta t} + \frac{\partial f}{\partial t} = 0$$

          And reorganising

          $$\begin{bmatrix} \frac{\partial f}{\partial i} & \frac{\partial f}{\partial j} \end{bmatrix} \begin{bmatrix} \frac{\Delta i}{\Delta t} \\ \frac{\Delta j}{\Delta t} \end{bmatrix} = -\frac{\partial f}{\partial t}$$
    - **Problems**
        - Just shows the apparent motion of points, in a scene, based on brightness patterns
        - What happens when the brightness changes (brightness constancy does not hold)
            - **Perhaps look at optical flow in the gradient space**
        - What is a point moves differently to its neighbours
            - **Use regions based optical flow**

        - What happens to a rotating sphere? Or a barber pole?
            - **It fails**
        - What happens if the motion is too large?
            - **Use iterative refinement**
- **Feature Based Tracking**
    - **Feature based optical flow**
        - We cannot compute optical flow for constant regions or along edges
        - **Better to compute just for features**