Computing Motion from Images

Chapter 9 of S&S plus other work.
General topics

- Low level change detection
- Region tracking or matching over time
- Interpretation of motion
- MPEG compression
- Interpretation of scene changes in video
- Understanding human activities
Motion important to human vision

1. definitely used in human vision
2. object detection and tracking
3. navigation and obstacle avoidance
4. analysis of actions in a scene
5. segmentation and understanding of video
What’s moving: different cases

- still camera, single moving object, constant background
- still camera, several moving objects, constant background
- moving camera, relatively constant scene
- moving camera, several moving objects
Image subtraction

Simple method to remove unchanging background from moving regions.
Change detection for surveillance

- (Left) person appears in an unoccupied workspace
- (Center) Image substraction reveals changed regions where person occludes background and at door and a CRT.
- (Right) change due to person is deemed significant while the other two are expected and hence ignored.
Change detection by image subtraction

Input $I_t[r,c]$ and $I_{t-\delta}[r,c]$: two monochrome input images taken $\delta$ seconds apart.
Input $\tau$ is an intensity threshold.
$I_{out}[r,c]$ is the binary output image; $B$ is a set of bounding boxes.

1. For all pixels $[r,c]$ in the input images,
   set $I_{out}[r,c] = 1$ if $(|I_t[r,c] - I_{t-\delta}[r,c]| > \tau)$
   set $I_{out}[r,c] = 0$ otherwise.

2. Perform connected components extraction on $I_{out}$.
3. Remove small regions assuming they are noise.
4. Perform a closing of $I_{out}$ using a small disk to fuse neighboring regions.
5. Compute the bounding boxes of all remaining regions of changed pixels.
6. Return $I_{out}[r,c]$ and the bounding boxes $B$ of regions of changed pixels.
Closing = dilation + erosion

What to do with regions of change?

- Discard small regions
- Discard regions of non interesting features
- Keep track of regions with interesting features
- Track in future frames from motion plus component features
Some effects of camera motion that can cause problems

- Effects of zooming and panning on imaged features.
- The effect of zoom in is similar to that observed when we move forward in a scene.
- The effect of panning is similar to that observed when we turn.
1 Definition A 2D array of 2D vectors representing the motion of 3D scene points is called the motion field. The motion vectors in the image represent the displacements of the images of moving 3D points. Each motion vector might be formed with its tail at an imaged 3D point at time $t$ and its head at the image of that same 3D point imaged at time $t + \delta$. Alternatively, each motion vector might correspond to an instantaneous velocity estimate at time $t$. 
FOE and FOC

2 Definition The focus of expansion (FOE) is that image point from which all motion field vectors diverge. The FOE is typically the image of a 3D scene point toward which the sensor is moving. The focus of contraction (FOC) is that image point toward which all motion vectors converge, and is typically the image of a 3D scene point from which the sensor is receding.

Will return to use the FOE or FOC or detection of panning to determine what the camera is doing in video tapes.
Global Motion Compensation

Gaming using a camera to recognize the player’s motion

Decathlete game
Decathlete game

The man at the left is making running motions with his arms and hands to control the game of running the hurdles.

The game display is shown at the right.

Running speed and jumping of the avatar is controlled by detected motion of the player’s hands.

from IEEE Computer Graphics, Vol 18, No. 3 (May-June 1998)
Motion detection input device

- Running (hands)
- Jumping (hands)
Motion analysis controls hurdles event (console)

- Top left shows video frame of player
- Middle left shows motion vectors from multiple frames
- Center shows jumping patterns
Related work

- Motion sensed by crude cameras
- Person dances/gestures in space
- Kinect/Leap motion sensors
- System maps movement into music
- Creative environment?
- Good exercise room?
Computing motion vectors from corresponding "points"

High energy neighborhoods are used to define points for matching
Match points between frames

- Find interest points $P_{t,j}$ in frame $t$
- Search for matching points $Q_{t+\delta t,j}$ in frame $t + \delta t$
- Form motion vectors $V_j = [P_{t,j}, Q_{t+\delta t,j}]$

Such large motions are unusual. Most systems track small motions.
Requirements for interest points

- have unique multidirectional energy / texture
- detected and located with high confidence
- edge detector is not good – constraint in only 1 direction
- corner detector is better – constraint in 2 directions
- autocorrelation can be used for matching in second image

Match small neighborhood to small neighborhood. The previous “scene” contains several highly textured neighborhoods.
Interest = minimum directional variance

```
real procedure interest_operator (I, r, c, w )
{
    "w is the halfwidth of operator window"
    "See alternate texture-based interest operator in the exercises."
    v1 := variance of intensity of horizontal pixels I_1[r, c - w]...I_1[r, c + w];
    v2 := variance of intensity of vertical pixels I_1[r - w, c]...I_1[r + w, c];
    v3 := variance of intensity of diagonal pixels I_1[r - w, c - w]...I_1[r + w, c + w];
    v4 := variance of intensity of diagonal pixels I_1[r - w, c + w]...I_1[r + w, c - w];
    return minimum {v1, v2, v3, v4};
}
```

Used by Hans Moravec in his robot stereo vision system.
Interest points were used for stereo matching.
Detecting interest points in I1

```plaintext
procedure detect_corner_points(I, V);
{
    "I[r, c] is an input image of MaxRow rows and MaxCol columns"
    "V is an output set of interesting points from I."
    "\( \tau \) is a threshold on the interest operator output"
    "\( w \) is the halfwidth of the neighborhood for the interest operator"
    for r := 0 to MaxRow - 1
        for c := 0 to MaxCol - 1
            {
                if I[r, c] is a border pixel then break;
                else if ( interest_operator (I, r, c, w) \( \geq \) \( \tau \)) then add \([ (r, c), (r, c) ]\) to set V;
                "The second \( (r, c) \) is a place holder in case vector tip found later."
            }
}
Match points from I1 in I2

$I_1[r, c]$ and $I_2[r, c]$ are input images of MaxRow rows and MaxCol columns. $V$ is the output set of motion vectors $\{[(T_x, T_y), (H_x, H_y)]_i\}$ where $(T_x, T_y)$ is the tail of a motion vector and $(H_x, H_y)$ is its head.

**procedure** extract_motion_field($I_1$, $I_2$, $V$)
{
  “Detect matching corner points and returning motion vectors $V$”
  “$\tau_2$ is a threshold on neighborhood cross-correlation”
  detect_corner_points($I_1$, $V$);
  for all vectors $[(T_x, T_y), (U_x, U_y)]$ in $V$
    match := best_match($I_1$, $I_2$, $T_x$, $T_y$, $H_x$, $H_y$);
    if (match < $\tau_2$) then delete $[(T_x, T_y), (U_x, U_y)]$ from $V$;
    else replace $[(T_x, T_y), (U_x, U_y)]$ with $[(T_x, T_y), (H_x, H_y)]$ in $V$;
}
Search for best match of point P1 in nearby window of I2

real procedure best_match( I_1, I_2, T_x, T_y, H_x, H_y );

“(H_x, H_y) is returned as the center of the neighborhood in I_2 that matches best”
“to the neighborhood centered at (T_x, T_y) in I_1.”

{ 
   “first indicate that a good match has not been found”
   H_x := -1; H_y := -1; best := 0.0;
   for r := T_y - sh to T_y + sh
      for c := T_x - sw to T_x + sw
         { 
            “cross correlate N in I_1 with N in I_2 as in Chapter 5”
            match := cross_correlate(I_1, I_2, T_x, T_y, r, c, h, w );
            if ( match > best ) then
               { 
                  H_y := r; H_x := c; best := match;
               }
         }
}

For both motion and stereo, we have some constraints on where to search for a matching interest point.
Motion vectors clustered to show 3 coherent regions

Motion coherence: points of same object tend to move in the same way

All motion vectors are clustered into 3 groups of similar vectors showing motion of 3 independent objects. (Dina Eldin)
Two frames of aerial imagery

Video frame N and N+1 shows slight movement: most pixels are same, just in different locations.
Can code frame N+d with displacements relative to frame N

- for each 16 x 16 block in the 2nd image
- find a closely matching block in the 1st image
- replace the 16x16 intensities by the location in the 1st image (dX, dY)
- 256 bytes replaced by 2 bytes!
- (If blocks differ too much, encode the differences to be added.)
Frame approximation

Left is original video frame N+1. Right is set of best image blocks taken from frame N. (Work of Dina Eldin)
Best matching blocks between video frames N+1 to N (motion vectors)

The bulk of the vectors show the true motion of the airplane taking the pictures. The long vectors are incorrect motion vectors, *but they do work well for compression of image I2!*

Best matches from 2\textsuperscript{nd} to first image shown as vectors overlaid on the 2\textsuperscript{nd} image. (Work by Dina Eldin.)
Motion coherence provides redundancy for compression

MPEG “motion compensation” represents motion of 16x16 pixels blocks, NOT objects
MPEG represents blocks that move by the motion vector
MPEG has ‘I’, ‘P’, and ‘B’ frames

- Example video sequence has four frames F1, F2, F3, F4
- F1 is coded as an independent (I) frame using JPEG
- F4 is a P frame predicted from F1 using motion vectors together with block differences:
  16 x 16 pixel blocks (b1) are located in frame F1 using a motion vector and a block of differences to be added
- Between frames B1 and B2 are determined entirely by interpolation using motion vectors: 16 x 16 blocks (b2) are reconstructed as an average of blocks (b4) in frame F1 and (b5) in frame F4
- Between frames F2 and F3 can only be decoded after predicted frame F4 has been decoded
- Between frames yield the most compression since each 16 x 16 pixel block is represented by only two motion vectors
- I frames yield the least compression
Computing Image Flow
3 Definition Optical Flow is the apparent flow of intensities across the retina due to motion in the scene or motion of the observer.

We can attempt to compute at each image point \( I[x, y, t] \) the spatio-temporal gradient, which represents the flow.

\[
\begin{array}{c c}
3333333333 & 3333333333 \\
3333333333 & 3333333333 \\
3333333333 & 3373333333 \\
3373333333 & 3397533333 \\
3397533333 & 3399753333 \\
3399753333 & 3399975333 \\
3399975333 & 3333333333 \\
3333333333 & 3333333333 \\
\end{array}
\]

(a) \( t_1 \)  \hspace{1cm} (b) \( t_2 \)

An example of image flow. A brighter triangle moves one pixel upward from time \( t_1 \) to time \( t_2 \). Background intensity is 3 while object intensity is 9.
Motion Field & Optical Flow Field

- Motion Field = Real world 3D motion
- Optical Flow Field = Projection of the motion field onto the 2d image

\[
\bar{u} = (u, v)
\]

Slides from Lihi Zelnik-Manor
Assumptions

• We assume that the object reflectivity and the illumination of the object does not change during the interval \([t_1, t_2]\).

• We assume that the distances of the object from the camera or light sources does not vary significantly over this interval.

• We shall also assume that each small intensity neighborhood \(N_{x,y}\) at time \(t_1\) is observed in some shifted position \(N_{x+\delta x, y+\delta y}\).

• We assume a continuous intensity function \(f(x, y)\) of continuous spatial parameters.
When does it break?

The screen is stationary yet displays motion

Homogeneous objects generate zero optical flow.

Fixed sphere. Changing light source.

Non-rigid texture motion

Slides from Lihi Zelnik-Manor
Image flow equation 1 of 2

\[ f(x + \delta x, y + \delta y, t + \delta t) = f(x, y, t) + \frac{\partial f}{\partial x} \delta x + \frac{\partial f}{\partial y} \delta y + \frac{\partial f}{\partial t} \delta t + h.o.t. \] (1)

The image flow vector \( \mathbf{V} = [\delta x, \delta y] \) carries the intensity neighborhood \( N_1 \) of \((x, y)\) at \( t_1 \) to an identical intensity neighborhood \( N_2 \) of \((x + \delta x, y + \delta y)\) at \( t_2 \). This assumption means that
The image flow vector $\mathbf{V} = [\delta x, \delta y]$ carries the intensity neighborhood $N_1$ of $(x, y)$ at $t_1$ to an identical intensity neighborhood $N_2$ of $(x + \delta x, y + \delta y)$ at $t_2$. This assumption means that

$$f(x + \delta x, y + \delta y, t + \delta t) = f(x, y, t)$$  \hspace{1cm} (2)

We obtain the image flow equation by combining Equations ?? and ?? and ignoring the higher order terms.

$$-\frac{\partial f}{\partial t} \delta t = \frac{\partial f}{\partial x} \delta x + \frac{\partial f}{\partial y} \delta y = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}] \circ [\delta x, \delta y] = \nabla f \circ [\delta x, \delta y]$$  \hspace{1cm} (3)
Estimating Optical Flow

- Assume the image intensity $I$ is constant

$$
I(x, y, t) = I(x + dx, y + dy, t + dt)
$$
Brightness Constancy Equation

\[ I(x, y, t) = I(x + dx, y + dy, t + dt) \]

First order Taylor Expansion

\[ = I(x, y, t) + \frac{\partial I}{\partial x} dx + \frac{\partial I}{\partial y} dy + \frac{\partial I}{\partial t} dt \]

Simplify notations:

\[ I_x dx + I_y dy + I_t dt = 0 \]

Divide by \( dt \) and denote:

\[ u = \frac{dx}{dt}, \quad v = \frac{dy}{dt} \]

Problem I: One equation, two unknowns

\[ I_x u + I_y v = -I_t \]

MSU Fall 2017
Slides from Lihi Zelnik-Manor
Problem II: “The Aperture Problem”

- For points on a line of fixed intensity we can only recover the normal flow

Where did the yellow point move to?

We need additional constraints
Use Local Information

Sometimes enlarging the aperture can help.
Local smoothness
Lucas Kanade (1984)

\[ I_x u + I_y v = -I_t \]

\[ \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -I_t \]

Assume constant \((u,v)\) in small neighborhood

\[ \begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \end{bmatrix} \]

\[ A\tilde{u} = b \]
Lucas Kanade (1984)

Goal: Minimize $\|A\tilde{u} - b\|^2$

Method: Least-Squares

$A\tilde{u} = b$

$A^T A \tilde{u} = A^T b$

$\tilde{u} = (A^T A)^{-1} A^T b$
Lucas-Kanade Solution

\[ \mathbf{\bar{u}} = \left( A^T A \right)^{-1} A^T \mathbf{b} \]

\[ A^T A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \]

We want this matrix to be invertible.

i.e., no zero eigenvalues
Break-downs

- Brightness constancy is **not** satisfied
  - Correlation based methods

- A point does **not** move like its neighbors
  - what is the ideal window size?
    - Regularization based methods

- The motion is **not** small (Taylor expansion doesn’t hold)
  - Use multi-scale estimation
Multi-Scale Flow Estimation

Gaussian pyramid of image $I_t$

image $I_t$

$u=10$ pixels

$u=5$ pixels

$u=2.5$ pixels

$u=1.25$ pixels

Gaussian pyramid of image $I_{t+1}$

image $I_{t+1}$

$u=10$ pixels

$u=5$ pixels

$u=2.5$ pixels

$u=1.25$ pixels
SIFTflow

\[ E(w) = \sum_p \min \left( \| s_1(p) - s_2(p + w(p)) \|_1, t \right) + \]
\[ \sum_p \eta(|u(p)| + |v(p)|) + \]
\[ \sum_{(p,q) \in \varepsilon} \min \left( \alpha |u(p) - u(q)|, d \right) + \min \left( \alpha |v(p) - v(q)|, d \right) \]

http://people.csail.mit.edu/celiu/SIFTflow/

Check out the Obstruction Free Photography work in 2016
Tracking several objects

Use assumptions of physics to compute multiple smooth paths.
(work of Sethi and R. Jain)
Trajectories of three objects, \( \bigcirc, \triangle, \square \) are shown: the location of each object is shown for six instants of time. \( \bigcirc \) and \( \triangle \) are generally moving from left to right while \( \square \) is moving right to left.
Tracking in images over time

- At each instant of time, which object is which?
- What features determine the objects?
- What constraints can we use from physics?
- What constraints can we use from media domain?
General constraints from physics

1. The location of a physical object changes smoothly over time.

2. The velocity of a physical object changes smoothly over time: this includes both its speed and direction.

3. An object can be at only one location in space at a given time.

4. Two objects cannot occupy the same location at the same time.

   The first three assumptions above hold for the 2D projections of 3D space; smooth
Other possible constraints

- Background statistics stable
- Object color/texture/shape might change slowly over frames
- Might have knowledge of objects under surveillance
- Objects appear/disappear at boundary of the frame
Trajectories of two objects, ⭕ and ◊ are shown along with the image flow vector at each of the first five points. A tracker would consider ◊₂ as a likely successor to ⭕₁ and ⭕₅ to be a possible ending of the sequence ⭕₁, ⭕₂, ⭕₃, ⭕₄.
Sethi-Jain algorithm

- first assume $m$ objects in each of $n$ frames
- define the smoothness of a single path
- define the smoothness of $m$ paths
- greedy exchange algorithm optimizes total smoothness
4 Definition *If an object* $i$ *is observed at time instants* $t = 1, 2, \ldots, n$, *then the sequence of image points* $T_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,t}, \ldots, p_{i,n})$ *is called the trajectory of* $i$.

![Diagram showing vectors entering and leaving a trajectory point](image)

Vectors entering and leaving trajectory point $p[i, t]$.

Between any two points of a trajectory, difference vector is

$$V_{i,t} = p_{i,t+1} - p_{i,t}$$  \hspace{1cm} (4)
Total smoothness of m paths

Smoothness value at a trajectory point $p_{i,t}$ defined in terms of entering and leaving vectors

1. Smoothness of direction is measured by their dot product.

2. Smoothness of speed is measured by comparing the geometric mean of their magnitudes to their average magnitude.

$$S_{i,t} = w \left( \frac{V_{i,t-1} \cdot V_{i,t}}{|V_{i,t-1}| |V_{i,t}|} \right) + (1 - w) \left( \frac{2\sqrt{|V_{i,t-1}| |V_{i,t}|}}{|V_{i,t-1}| + |V_{i,t}|} \right)$$  \hspace{1cm} (5)

Total smoothness $T_s = \sum_{i=1}^{m} \sum_{t=2}^{n-1} S_{i,t}$  \hspace{1cm} (6)
Greedy exchange algorithm

1. **initialize**: create $m$ complete paths by linking nearest neighbors

2. **exchange loop**: for time $t = 2 \ldots n - 1$
   
   (a) for all pairs $j \neq k$, compute increase in smoothness if $T[j, t], T[k, t]$ are exchanged
   
   (b) make that exchange, if any, that increases total smoothness most
   
   (c) if exchange made, set exchange flag ON

3. **done?**: if exchange flag ON, set it OFF and repeat exchange loop
### Example data structure

#### Total smoothness for trajectories of Figure 9.14

<table>
<thead>
<tr>
<th></th>
<th>t=1</th>
<th>t=2</th>
<th>t=3</th>
<th>t=4</th>
<th>t=5</th>
<th>smoothness</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bigcirc_1(112\ 262)$</td>
<td>$\square_2(206\ 185)$</td>
<td>$\bigcirc_3(250\ 137)$</td>
<td></td>
<td></td>
<td>0.97</td>
<td></td>
</tr>
<tr>
<td>$\square_1(106\ 175)$</td>
<td>$\bigcirc_2(180\ 188)$</td>
<td></td>
<td>$\square_3(280\ 185)$</td>
<td></td>
<td>0.98</td>
<td></td>
</tr>
<tr>
<td>$\bigcirc_1(112\ 262)$</td>
<td></td>
<td>$\bigcirc_2(180\ 188)$</td>
<td>$\bigcirc_3(250\ 137)$</td>
<td></td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>$\square_1(106\ 175)$</td>
<td>$\square_2(206\ 185)$</td>
<td></td>
<td>$\square_3(280\ 185)$</td>
<td></td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>$\bigcirc_1(112\ 262)$</td>
<td></td>
<td>$\bigcirc_2(180\ 188)$</td>
<td></td>
<td>$\bigcirc_4(360\ 137)$</td>
<td>1.89</td>
<td></td>
</tr>
<tr>
<td>$\square_1(106\ 175)$</td>
<td>$\square_2(206\ 185)$</td>
<td></td>
<td>$\square_3(280\ 185)$</td>
<td>$\square_4(365\ 156)$</td>
<td>1.96</td>
<td></td>
</tr>
<tr>
<td>$\bigcirc_1(112\ 262)$</td>
<td></td>
<td>$\bigcirc_2(180\ 188)$</td>
<td></td>
<td>$\bigcirc_4(360\ 137)$</td>
<td>$\square_5(482\ 80)$</td>
<td>2.84</td>
</tr>
<tr>
<td>$\square_1(106\ 175)$</td>
<td>$\square_2(206\ 185)$</td>
<td></td>
<td>$\square_3(280\ 185)$</td>
<td>$\square_4(365\ 156)$</td>
<td>$\bigcirc_5(478\ 170)$</td>
<td>2.91</td>
</tr>
<tr>
<td>$\bigcirc_1(112\ 262)$</td>
<td></td>
<td>$\bigcirc_2(180\ 188)$</td>
<td></td>
<td>$\bigcirc_4(360\ 137)$</td>
<td>$\bigcirc_5(478\ 170)$</td>
<td>2.89</td>
</tr>
<tr>
<td>$\square_1(106\ 175)$</td>
<td>$\square_2(206\ 185)$</td>
<td></td>
<td>$\square_3(280\ 185)$</td>
<td>$\square_4(365\ 156)$</td>
<td>$\square_5(482\ 80)$</td>
<td>2.94</td>
</tr>
</tbody>
</table>
Example of domain specific tracking (Vera Bakic)

Tracking eyes and nose of PC user. System presents menu (top). User moves face to position cursor to a particular box (choice). System tracks face movement and moves cursor accordingly: user gets into feedback-control loop.
Segmentation of videos/movies

Segment into scenes, shots, specific actions, etc.
Types of changes in videos

- A **scene change** is a change of environment; for example, from a restaurant scene to a street scene. Gross changes in the background are expected.

- A **shot change** is a significant change of camera view of the same scene. Often, this is accomplished by switching cameras.

- A **camera pan** is used to sweep a horizontal view of the scene.

- **Camera zoom** changes the focal length over time to expand the image of some part of the scene (zoom in) or to reduce the image of a scene part and include more adjacent background (zoom out).

- **Camera effects fade, dissolve, and wipe** are used for transitions from one source of imagery to a different source of imagery.
How do we compute the scene change?

Anchor person scene at left

Scene break

Street scene for news story

From Zhang et al 1993
Histograms of frames across the scene change

Histograms at left are from anchor person frames, while histogram at bottom right is from the street frame.