Lecture-12

Model-based Video Compression
Li, Teklap

Video Compression

What is Compression?

- Compression is a process of converting data into a form requiring less space to store or less time to transmit, which permits the original data to be reconstructed with acceptable precision at a later time.

Orange Juice Analogy!

- Freshly squeezed orange juice (uncompressed)
- Remove water (redundancy), convert it to concentrate (encoding)
- Shipped, stored, and sold.
- Add water to concentrate (decoding), tastes like freshly squeezed!!!

Why is compression necessary?

- Storage space limitations
- Transmission bandwidth limitations.

Why is compression acceptable?

- Limitations of visual perception
  - Number of shades (colors, gray levels) we can perceive
  - Reduced sensitivity to noise in high-frequencies (e.g. edges of objects)
  - Reduced sensitivity to noise in brighter areas
- Ability of visual perception
  - Ability of the eye to integrate spatially
  - Ability of the mind to interpolate temporally

Why is compression acceptable?

- Some type of visual information is less important than others
- Goal is to throw away bits in psycho-visually lossless manner
- We have been conditioned to accept imperfect reproduction
- Limitations of intended output devices

Why is compression possible?

- Some sample values (gray levels, colors) are more likely to occur at a particular pixel than others.
  - Remove spatial and temporal redundancy that exist in natural video
    - Correlation itself can be removed in a lossless fashion
    - Important to medical applications
    - Only realizes about 2:1 compression

Why is compression possible?

- No single algorithm can compress all possible data
- Random data cannot be compressed

Lossless Compression

- Needed when loss is unacceptable or highly undesirable
- Fixed compression ratio is hard to achieve
- Compression/decompression time varies with image

Lossy Compression

- Used when loss is acceptable or inevitable
- Permits fixed compression ratios
- Better suited for fixed time decompression

Compression Techniques

- Subsampling
- Quantization
- Delta Coding
- Prediction
- Color space conversion
- Huffman coding
- Run-length encoding
- De-correlation
- Motion Compensation
- Model-based compression

Subsampling

- Selecting one single value to represent several values in a part of the image.
  - For example, use top left corner of 2X2 block to represent the block
  - Compression ratio 75%

11	15	19	55
13	14	21	32
39	17	24	76
43	34	27	80

→

11	11	19	19
11	11	19	19
39	39	24	24
39	39	24	24

Subsampling

- A better way- averaging
- Compression ratio 75%

11	15	19	55
13	14	21	32
39	17	24	76
43	34	27	80

→

13	13	32	32
13	13	32	32
33	33	51	51
33	33	51	51

Quantization

- Mapping of a large range of possible sample values into a smaller range of values or codes.
- Fewer bits are required to encode the quantized sample.
- Examples
  - -Letter grades (A, B, C, D, F)
  - Rounding of person’s age, height, or weight

Quantization

- Truncation and Rounding
- Quantized levels need not be evenly spaced
- Can be used for relative as well as absolute information
- Information is lost in quantiztion, but the error can be recovered

Truncation

- Discard lower-order bits
  - average error 1/2 LSB of target resolution
- Example

9	11	17	21		0	10	10	20
19	51	33	14		10	50	30	10
19	23	18	15	→	10	20	10	10
53	47	12	43		50	40	10	40

Rounding

- Add 5 and then truncate the result.
  - One more LSB participate than in truncation
  - average error 1/4 LSB

13	19	9	5		10	20	10	10
14	17	8	15		10	20	10	20
52	49	53	47	→	50	50	50	50
50	58	51	42		50	60	50	40

Delta Coding

- Code the difference between adjacent pixels.
- Since adjacent pixels are similar, the difference is normally small, and requires fewer bits to code.
- A typical pixel value requires 8 bits.
- The difference between any 8 bit pixels is in the range [-255,255], which needs 9 bits!

Delta Coding

- But most deltas will be small.
  - Smaller deltas can be assigned shorter codes
  - Smaller deltas can be ignored completely
  - smaller deltas can be quantized more finally for better quality
- Complementary delta values can share a code; e.g., +1 and -255 yield same result in 8 bit positive value.
- 9 bits are not required!

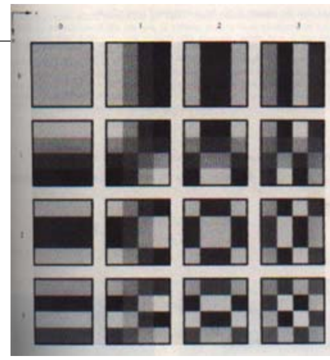
Discrete Cosine Transform

$$C(u,v)=\alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

$$f(x,y)=\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\alpha(u)\alpha(v)C(u,v)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

$$\alpha(u)=\begin{cases}\sqrt{\frac{1}{N}} & u=0\\ \sqrt{\frac{2}{N}} & u=1,2,\dots N-1\end{cases}$$

DCT Bases Functions



Example

$$I_{i,j}=\begin{bmatrix}139 & 144 & 149 & 153 & 155 & 155 & 155 & 155\\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156\\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156\\ 159 & 161 & 162 & 160 & 160 & 159 & 159 & 159\\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155\\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157\\ 162 & 162 & 161 & 161 & 162 & 157 & 157 & 157\\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158\end{bmatrix}$$

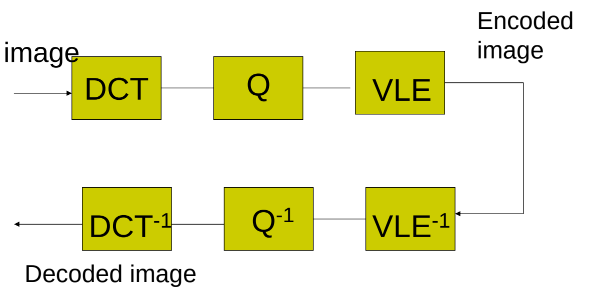
image

Example

$$F_{u,v}=\begin{bmatrix}1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1\\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & 1\\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0\\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0\\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1\\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1\\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1\\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0\end{bmatrix}$$

DCT

JPEG BLOCK DIAGRAM



JPEG Baseline Coding

- Divide image into blocks of size 8X8.
- Level shift all 64 pixels values in each block by subtracting 2^{n-1}, (where 2^n is the maximum number of gray levels).
- Compute 2D DCT of a block.
- Quantize DCT coefficients using quantization table.

JPEG Baseline Coding

- Zig-zag scan the quantized DCT coefficients to form 1-D sequence.
- Code 1-D sequence (AC and DC) using JPEG Huffman variable length codes.

JPEG Quantization Table (Luma)

$$Q_{u,v} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 51 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

JPEG Quantization Table (Chroma)

$$
\begin{bmatrix}
17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\
18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\
24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\
47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\
99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\
99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\
99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\
99 & 99 & 99 & 99 & 99 & 99 & 99 & 99
\end{bmatrix}
$$

JPEG ZIG-ZAG SCAN

$$
\begin{bmatrix}
1 & 2 & 6 & 7 & 15 & 16 & 28 & 29 \\
3 & 5 & 8 & 14 & 17 & 27 & 30 & 43 \\
4 & 9 & 13 & 18 & 26 & 31 & 42 & 44 \\
10 & 12 & 19 & 25 & 32 & 41 & 45 & 54 \\
11 & 20 & 24 & 33 & 40 & 46 & 53 & 55 \\
21 & 23 & 34 & 39 & 47 & 52 & 56 & 61 \\
22 & 35 & 38 & 48 & 51 & 57 & 60 & 62 \\
36 & 37 & 49 & 50 & 58 & 59 & 63 & 64
\end{bmatrix}
$$

Example (Encoding)

$$
I = \begin{bmatrix}
52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\
63 & 59 & 66 & 90 & 109 & 85 & 69 & 72 \\
62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\
63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\
67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\
79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\
85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\
87 & 79 & 69 & 68 & 65 & 76 & 78 & 94
\end{bmatrix}
\quad
I' = \begin{bmatrix}
-76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\
-65 & -69 & -62 & -38 & -19 & -43 & -59 & -56 \\
-66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\
-65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\
-61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\
-49 & -63 & -68 & -58 & -51 & -65 & -70 & -53 \\
-43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\
-41 & -49 & -59 & -60 & -63 & -52 & -50 & -34
\end{bmatrix}
$$

$$
DCT = \begin{bmatrix}
-415 & -29 & -62 & 25 & 55 & -20 & -1 & 3 \\
7 & -21 & -62 & 9 & 11 & -7 & -6 & 6 \\
-46 & 8 & 77 & -25 & -30 & 10 & 7 & -5 \\
-50 & 13 & 35 & -15 & -9 & 6 & 0 & 3 \\
-11 & -8 & -13 & -2 & -1 & 1 & -4 & 1 \\
-10 & 1 & 3 & -3 & -1 & 0 & 2 & -1 \\
-4 & -1 & 2 & -1 & 2 & -3 & 1 & -2 \\
-1 & -1 & -1 & -2 & -1 & -1 & 0 & -1
\end{bmatrix}
\quad
Q' = \begin{bmatrix}
-26 & -3 & -6 & 2 & 2 & 0 & 0 & 0 \\
1 & -2 & -4 & 0 & 0 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Example (Decoding)

$$
P = \begin{bmatrix}
-26 & -3 & -6 & 2 & 2 & 0 & 0 & 0 \\
1 & -2 & -4 & 0 & 0 & 0 & 0 & 0 \\
-3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\
-4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\quad
P' = \begin{bmatrix}
-416 & -33 & -60 & 32 & 48 & 0 & 0 & 0 \\
12 & -24 & -56 & 0 & 0 & 0 & 0 & 0 \\
-42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\
-56 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\
18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$
P'' = \begin{bmatrix}
-70 & -64 & -61 & -64 & -69 & -66 & -58 & -50 \\
-72 & -73 & -61 & -39 & -30 & -40 & -54 & -59 \\
-68 & -78 & -58 & -9 & 13 & -12 & -48 & -64 \\
-59 & -77 & -57 & 0 & 22 & -13 & -51 & -60 \\
-52 & -71 & -72 & -54 & -54 & -71 & -71 & -54 \\
-42 & -50 & -70 & -68 & -67 & -67 & -61 & -50 \\
-45 & -59 & -70 & -68 & -67 & -67 & -61 & -50 \\
-35 & 47 & -61 & -66 & -60 & -48 & -44 & -44
\end{bmatrix}
\quad
P''' = \begin{bmatrix}
58 & 64 & 67 & 64 & 59 & 62 & 70 & 78 \\
56 & 55 & 67 & 89 & 98 & 88 & 74 & 69 \\
60 & 50 & 70 & 119 & 141 & 116 & 80 & 64 \\
69 & 51 & 71 & 128 & 149 & 115 & 77 & 68 \\
74 & 53 & 64 & 105 & 115 & 84 & 65 & 72 \\
76 & 57 & 56 & 74 & 75 & 57 & 57 & 74 \\
83 & 69 & 59 & 60 & 61 & 61 & 67 & 78 \\
93 & 81 & 67 & 62 & 69 & 80 & 84 & 84
\end{bmatrix}
$$

Comparison

$$
I = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 66 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}
\qquad
P''' = \begin{bmatrix} 58 & 64 & 67 & 64 & 59 & 62 & 70 & 78 \\ 56 & 55 & 67 & 89 & 98 & 88 & 74 & 69 \\ 60 & 50 & 70 & 119 & 141 & 116 & 80 & 64 \\ 69 & 51 & 71 & 128 & 149 & 115 & 77 & 68 \\ 74 & 53 & 64 & 105 & 115 & 84 & 65 & 72 \\ 76 & 57 & 56 & 74 & 75 & 57 & 57 & 74 \\ 83 & 69 & 59 & 60 & 61 & 61 & 67 & 78 \\ 93 & 81 & 67 & 62 & 69 & 80 & 84 & 84 \end{bmatrix}
$$

Original Image   Decoded Image

Difference

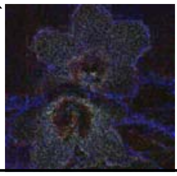
$$
Diff = \begin{bmatrix} -6 & -9 & -6 & 2 & 11 & -1 & -6 & -5 \\ 7 & 4 & -1 & 1 & 11 & -3 & -5 & 3 \\ 2 & 9 & -2 & -6 & -3 & -12 & -14 & 9 \\ -6 & 7 & 0 & -4 & -5 & -9 & -7 & 1 \\ -7 & 8 & 4 & -1 & 11 & 4 & 3 & 2 \\ 3 & 8 & 4 & -4 & 2 & 11 & 1 & 1 \\ 2 & 2 & 5 & -1 & -6 & 0 & -2 & 5 \\ -6 & -2 & 2 & 6 & -4 & -4 & -6 & 10 \end{bmatrix}
$$

JPEG

Original 64K

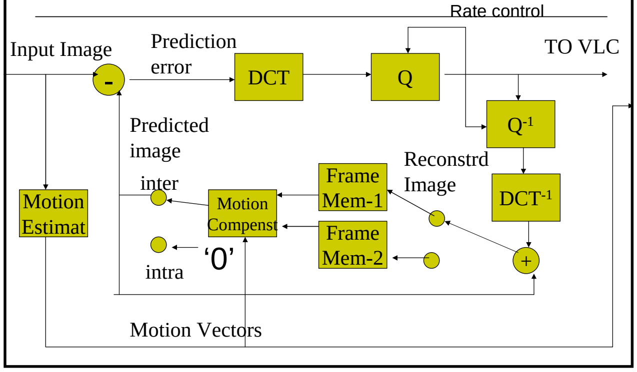
13K

5K



Difference

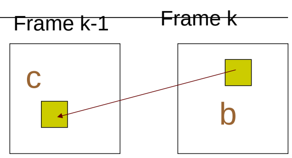
MPEG-1 Encoder



Motion Prediction

$$b' = c'$$

$$Error = b - b'$$



MPEG-1 & MPEG -2 Artifacts

- Blockiness
  - poor motion estimation
  - seen during dissolves and fades
- Mosquito Noises
  - edges of objects (high frequency DCT terms)
- Dirty Window
  - streaks or noise remain stationary while objects move

MPEG-1 & MPEG -2 Artifacts

- Wavy Noise
  - seen during pans across crowds
  - coarsely quantized high frequency terms cause errors

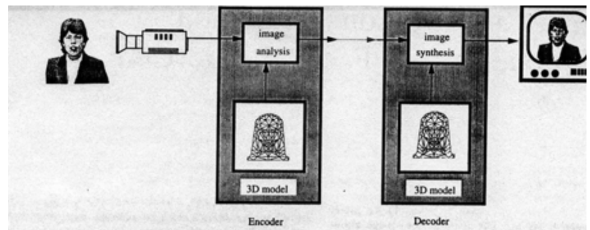
Where MPEG-2 will fail?

- Motions which are not translation
  - zooms
  - rotations
  - non-rigid (smoke)
  - dissolves
- Others
  - shadows
  - scene cuts
  - changes in brightness

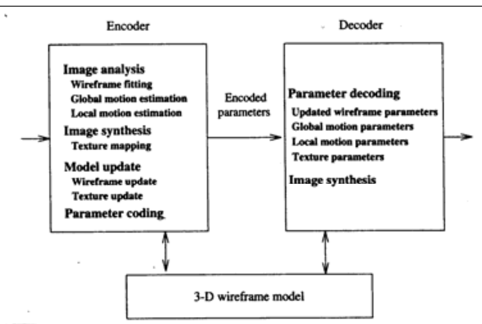
Video Compression At Low Bitrate

- The quality of block-based coding video (MPEG-1 & MPEG-2) at low bitrate, e.g., 10 kbps is very poor.
  - Decompressed images suffer from blockiness artifacts
  - Block matching does not account for rotation, scaling and shear

Model-Based Image Coding



Model-Based Image Coding



Model-Based Image Coding

- The transmitter and receiver both posses the same 3D face model and texture images.
- During the session, at the transmitter the facial motion parameters: global and local, are extracted.
- At the receiver the image is synthesized using estimated motion parameters.
- The difference between synthesized and actual image can be transmitted as residuals.

Candide Model

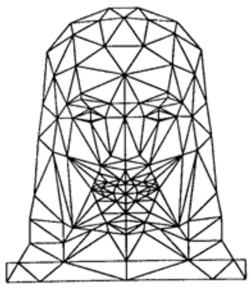


Fig. 2. Wire-frame model of the face.

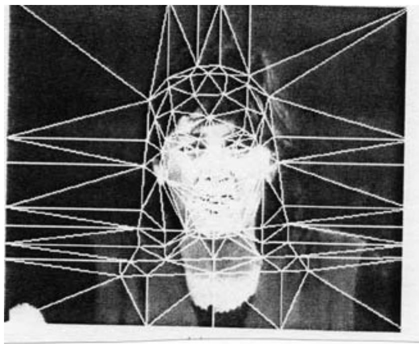
[..\..\CANDIDE.HTM](..\..\CANDIDE.HTM)

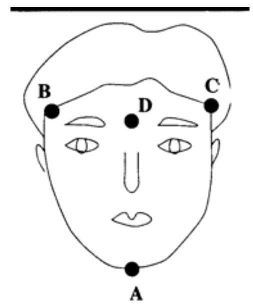
Face Model

- Candide model has 108 nodes, 184 polygons.
- Candide is a generic head and shoulder model. It needs to be conformed to a particular person’s face.
- Cyberware scan gives head model consisting of 460,000 polygons.

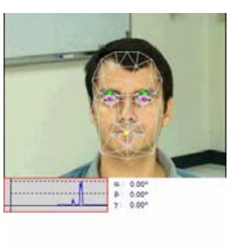
Wireframe Model Fitting

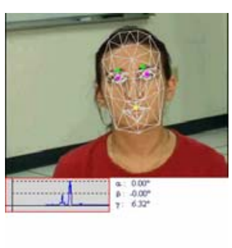
- Fit orthographic projection of wireframe to the frontal view of speaker using Affine transformation.
  - Locate three to four features in the image and the projection of a model.
  - Find parameters of Affine transformation using least squares fit.
  - Apply Affine to all vertices, and scale $\sqrt{(a_1^2 + a_4^2)^2 / 2}$ depth.





Alper Yilmaz and Mubarak Shah, Automatic Feature Detection and Pose Recovery for Faces, Asian Conference on Computer Vision, Australia, Jan 2002.

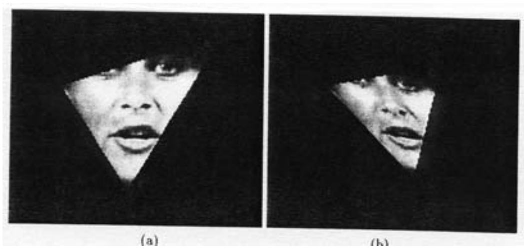




Synthesis

- Collapse initial wire frame onto the image to obtain a collection of triangles.
- Map observed texture in the first frame into respective triangles.
- Rotate and translate the initial wire frame according to global and local motion, and collapse onto the next frame.
- Map texture within each triangle from first frame to the next frame by interpolation.

Texture Mapping


(a) (b)

Video Phones

Motion Estimation

Perspective Projection (optical flow)

$$u = f(\frac{V_1}{Z} + \Omega_2) - \frac{V_3}{Z}x - \Omega_3 y - \frac{\Omega_1}{f}xy + \frac{\Omega_2}{f}x^2$$

$$v = f(\frac{V_2}{Z} - \Omega_1) + \Omega_3 x - \frac{V_3}{Z}y + \frac{\Omega_2}{f}xy - \frac{\Omega_1}{f}y^2$$

Optical Flow Constraint Eq

$$f_x u + f_y v + f_t = 0$$

$$f_x(f(\frac{V_1}{Z}+\Omega_2)-\frac{V_3}{Z}x-\Omega_3 y-\frac{\Omega_1}{f}xy+\frac{\Omega_2}{f}x^2)+f_y$$

$$(f(\frac{V_2}{Z}-\Omega_1)+\Omega_3 x-\frac{V_3}{Z}y+\frac{\Omega_2}{f}xy-\frac{\Omega_1}{f}y^2)+f_t=0$$

$$(f_x\frac{f}{Z})V_1+(f_y\frac{f}{Z})V_2+(\frac{f}{Z}(f_x x-f_y y)V_3+$$

$$(-f_x\frac{xy}{f}+f_y\frac{y^2}{f}-f_y f)\Omega_1+(f_x f+f_x\frac{x^2}{f}+f_y\frac{xy}{f})\Omega_2+$$

$$(f_x y+f_y x)\Omega_3=-f_t$$

$$(f_x\frac{f}{Z})V_1+(f_y\frac{f}{Z})V_2+(\frac{f}{Z}(f_x x-f_y y)V_3+$$

$$(-f_x\frac{xy}{f}+f_y\frac{y^2}{f}-f_y f)\Omega_1+(f_x f+f_x\frac{x^2}{f}+f_y\frac{xy}{f})\Omega_2+$$

$$(f_x y+f_y x)\Omega_3=-f_t$$

$$\mathbf{Ax}=\mathbf{b}$$

Solve by Least Squares

$$\mathbf{x}=(V_1,\quad V_2,\quad V_3,\quad \Omega_1,\quad \Omega_2,\quad \Omega_3)$$

$$\mathbf{Ax}=\mathbf{b}$$

$$\begin{bmatrix} & & & \vdots & & & \\ (f_x\frac{f}{Z}) & (f_y\frac{f}{Z}) & (\frac{f}{Z}(f_x x-f_y y) & (-f_x\frac{xy}{f}+f_y\frac{y^2}{f}-f_y f) & (f_x f+f_x\frac{x^2}{f}+f_y\frac{xy}{f}) & (f_x y+f_y x) \\ & & & \vdots & & & \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix}=\begin{bmatrix} \vdots \\ f_t \\ \vdots \end{bmatrix}$$

Comments

- This is a simpler (linear) problem than sfm because depth is assumed to be known.
- Since no optical flow is computed, this is called “direct method”.
- Only spatiotemporal derivatives are computed from the images.

Problem

- We have used 3D rigid motion, but face is not purely rigid!
- Facial expressions produce non-rigid motion.
- Use global rigid motion and non-rigid deformations.

3-D Rigid Motion

$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T}$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & -\gamma \\ -\beta & \gamma & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \left(\begin{bmatrix} 0 & -\alpha & \beta \\ \alpha & 0 & -\gamma \\ -\beta & \gamma & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

3-D Rigid Motion

$$\begin{bmatrix} X'-X \\ Y'-Y \\ Z'-Z \end{bmatrix} = \begin{bmatrix} 0 & -\alpha & \beta \\ \alpha & 0 & -\gamma \\ -\beta & \gamma & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\dot{\mathbf{X}} = \Omega \times \mathbf{X} + \mathbf{V}$$

3-D Rigid+Non-rigid Motion

$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T} + \mathbf{E}(\mathbf{X})\Phi$$

Facial expressions

Action Units:
-opening of a mouth
-closing of eyes
-raising of eyebrows

$$\Phi = (\phi_1, \phi_2, \ldots, \phi_m)^T$$

3-D Rigid+Non-rigid Motion

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & -\alpha & \beta \\ \alpha & 1 & -\gamma \\ -\beta & \gamma & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X + \sum_{i=1}^{m} E_{1i}(X)\phi_i \\ T_Y + \sum_{i=1}^{m} E_{2i}(X)\phi_i \\ T_Z + \sum_{i=1}^{m} E_{3i}(X)\phi_i \end{bmatrix}$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 0 & -\alpha & \beta \\ \alpha & 0 & -\gamma \\ -\beta & \gamma & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X + \sum_{i=1}^{m} E_{1i}(X)\phi_i \\ T_Y + \sum_{i=1}^{m} E_{2i}(X)\phi_i \\ T_Z + \sum_{i=1}^{m} E_{3i}(X)\phi_i \end{bmatrix}$$

3-D Rigid+Non-rigid Motion

$$\begin{bmatrix} X' - X \\ Y' - Y \\ Z' - Z \end{bmatrix} = \begin{bmatrix} 0 & -\alpha & \beta \\ \alpha & 0 & -\gamma \\ -\beta & \gamma & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X + \sum_{i=1}^{m} E_{1i}(X)\phi_i \\ T_Y + \sum_{i=1}^{m} E_{2i}(X)\phi_i \\ T_Z + \sum_{i=1}^{m} E_{3i}(X)\phi_i \end{bmatrix}$$

3-D Rigid+Non-rigid Motion

$$\dot{X} = -\Omega_3 Y + \Omega_2 Z + V_1 + \sum_{i=1}^{m} E_{1i}\phi_i$$

$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2 + \sum_{i=1}^{m} E_{2i}\phi_i$$

$$\dot{Z} = -\Omega_2 X + \Omega_1 Y + V_3 + \sum_{i=1}^{m} E_{3i}\phi_i$$

Perspective Projection (arbitrary flow)

$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

$$u = \dot{x} = \frac{fZ\dot{X} - fX\dot{Z}}{Z^2} = f\frac{\dot{X}}{Z} - x\frac{\dot{Z}}{Z}$$

$$v = \dot{y} = \frac{fZ\dot{Y} - fY\dot{Z}}{Z^2} = f\frac{\dot{Y}}{Z} - y\frac{\dot{Z}}{Z}$$

Perspective Projection (arbitrary flow)

$$\dot{X} = -\Omega_3 Y + \Omega_2 Z + V_1 + \sum_{i=1}^{m} E_{1i}\phi_i$$

$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2 + \sum_{i=1}^{m} E_{2i}\phi_i$$

$$\dot{Z} = -\Omega_2 X + \Omega_1 Y + V_3 + \sum_{i=1}^{m} E_{3i}\phi_i$$

$$u = \dot{x} = \frac{fZ\dot{X} - fX\dot{Z}}{Z^2} = f\frac{\dot{X}}{Z} - x\frac{\dot{Z}}{Z}$$

$$v = \dot{y} = \frac{fZ\dot{Y} - fY\dot{Z}}{Z^2} = f\frac{\dot{Y}}{Z} - y\frac{\dot{Z}}{Z}$$

$$u = f\left(\frac{V_1 + \sum_{i=1}^{m} E_{1i}\phi_i}{Z} + \Omega_2\right) - \frac{V_3 + \sum_{i=1}^{m} E_{3i}\phi_i}{Z}x - \Omega_3 y - \frac{\Omega_1}{f}xy + \frac{\Omega_2}{f}x^2$$

$$v = f\left(\frac{V_2 + \sum_{i=1}^{m} E_{2i}\phi_i}{Z} - \Omega_1\right) + \Omega_3 x - \frac{V_3 + \sum_{i=1}^{m} E_{3i}\phi_i}{Z}y + \frac{\Omega_1}{f}y^2 - \frac{\Omega_2}{f}xy$$

Optical Flow Constraint Eq

$$f_x u + f_y v + f_t = 0$$

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{x} = (V_1, V_2, V_3, \Omega_1, \Omega_2, \Omega_3, \phi_1, \phi_2, \ldots, \phi_m)$$

3-D motion estimation in model-based facial image coding
Li, H.; Roivainen, P.; Forchheimer, R.
Pattern Analysis and Machine Intelligence, IEEE Transactions on
Volume 15, Issue 6, Jun 1993 Page(s):545 - 555

Estimation Using Flexible Wireframe Model

Main Points

- Model photometric effects
- Simultaneously compute 3D motion and adapt the wireframe model.

Generalized Optical Flow Constraint

Lambertian Model

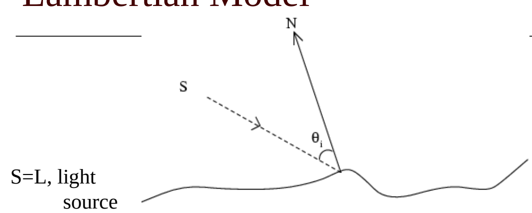
$$f(x,y,t) = \rho N(t).L$$

Albedo
Surface Normal
(-p,-q,1)

$$\frac{df(x,y,t)}{dt} = \rho L.\frac{dN}{dt}$$

$$f_x u + f_y v + f_t = \rho L.\frac{dN}{dt}$$

Lambertian Model



$$f(x,y) = n.L = (n_x, n_y, n_z).(l_x, l_y, l_z)$$

$$f(x,y) = n.L = (\frac{1}{\sqrt{p^2+q^2+1}}(-p,-q,1)).(l_x, l_y, l_z)$$

Sphere

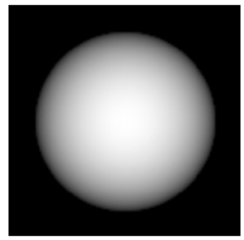
$$z = \sqrt{(R^2 - x^2 - y^2)}$$

$$p = \frac{\partial z}{\partial x} = -\frac{x}{z}$$

$$q = \frac{\partial z}{\partial y} = -\frac{y}{z}$$

$$(n_x, n_y, n_z) = \frac{1}{R}(x, y, z)$$

Sphere



Vase

(1, 0, 1)

(-1, 1, 1)

(-1,-1, 1)

Orthographic Projection

$$\dot{\mathbf{X}} = \Omega \times \mathbf{X} + \mathbf{V}$$

$$\dot{X} = \Omega_2 Z - \Omega_3 Y + V_1$$

$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2$$

$$\dot{Z} = \Omega_1 Y - \Omega_2 X + V_3$$

(u,v) is optical flow

$$u = \dot{x} = \Omega_2 Z - \Omega_3 y + V_1$$

$$v = \dot{y} = \Omega_3 x - \Omega_1 Z + V_2$$

Optical flow equation

$$f_x(\Omega_2 Z - \Omega_3 y + V_1) + f_y(\Omega_3 x - \Omega_1 Z + V_2) + f_t = \rho L.\frac{dN}{dt}$$

$$f_x(\Omega_2 Z - \Omega_3 y + V_1) + f_y(\Omega_3 x - \Omega_1 Z + V_2) + f_t =$$
$$\rho L.\left[\frac{(-p',-q',1)^T}{\sqrt{p'^2+q'^2+1}} - \frac{(-p,-q,1)^T}{\sqrt{p^2+q^2+1}}\right]$$

Homework 4.1
Show this.

Error Function

$$f_x(\Omega_2 Z - \Omega_3 y + V_1) + f_y(\Omega_3 x - \Omega_1 Z + V_2) + f_t =$$
$$\rho L.\left[\frac{(-p',-q',1)^T}{\sqrt{p'^2+q'^2+1}} - \frac{(-p,-q,1)^T}{\sqrt{p^2+q^2+1}}\right]$$

$$e_i(x,y) = f_x(\Omega_3 y - \Omega_2(p_i x + q_1 y + c_i) + V_1)$$
$$+ f_y(-\Omega_3 x + \Omega_1(p_i x + q_1 y + c_i) + V_2) + f_t$$
$$-\rho(L_1,L_2,L_3).\left(\frac{(-\frac{-\Omega_2+p_i}{1+\Omega_2 p_i}, \frac{-\Omega_1+q_i}{1-\Omega_1 q_i})}{((\frac{-\Omega_2+p_i}{1+\Omega_2 p_i})^2 + (\frac{-\Omega_1+q_i}{1-\Omega_1 q_i})^2 + 1)^{1/2}} - \frac{(-p_i,-q_i,1)}{(p_i^2+q_i^2+1)^{1/2}}\right.$$

Homework 4.3
Show this.

Error Function

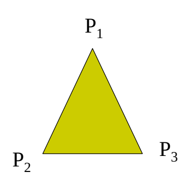
$$E = \sum_i \sum_{(x,y)\varepsilon ith patch} e_i^2$$

$$p_i x_1^{(ij)} + q_i x_2^{(ij)} + c_i = p_j x_1^{(ij)} + q_j x_2^{(ij)} + c_j$$

Constraint: Neighboring patches
Should intersect at straight line

Equation of a Planar Patch

$$P_1^{(i)} = (X_1^{(i)}, Y_1^{(i)}, Z_1^{(i)})$$
$$P_2^{(i)} = (X_2^{(i)}, Y_2^{(i)}, Z_2^{(i)})$$
$$P_3^{(i)} = (X_3^{(i)}, Y_3^{(i)}, Z_3^{(i)})$$



$$P^{(i)} = (X^{(i)}, Y^{(i)}, Z^{(i)})$$

$$\overline{P^{(i)} P_1^{(i)}}.(\overline{P_2^{(i)} P_1^{(i)}} \times \overline{P_3^{(i)} P_1^{(i)}}) = 0$$

Equation of a Planar Patch

Homework 4.2
Show this.

$$Z^{(i)} = p_i X^{(i)} + q_i Y^{(i)} + c_i$$

$$p_i = -\frac{(Y_2^{(i)} - Y_1^{(i)})(Z_3^{(i)} - Z_1^{(i)}) - (Z_2^{(i)} - Z_1^{(i)})(Y_3^{(i)} - Y_1^{(i)})}{(X_2^{(i)} - X_1^{(i)})(Y_3^{(i)} - Y_1^{(i)}) - (Y_2^{(i)} - Y_1^{(i)})(X_3^{(i)} - X_1^{(i)})}$$

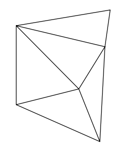
$$q_i = -\frac{(Z_2^{(i)} - Z_1^{(i)})(X_3^{(i)} - X_1^{(i)}) - (X_2^{(i)} - X_1^{(i)})(Z_3^{(i)} - Z_1^{(i)})}{(X_2^{(i)} - X_1^{(i)})(Y_3^{(i)} - Y_1^{(i)}) - (Y_2^{(i)} - Y_1^{(i)})(X_3^{(i)} - X_1^{(i)})}$$

$$c_i = Z_1^{(i)} + X_1^{(i)} \frac{(Y_2^{(i)} - Y_1^{(i)})(Z_3^{(i)} - Z_1^{(i)}) - (Z_2^{(i)} - Z_1^{(i)})(Y_3^{(i)} - Y_1^{(i)})}{(X_2^{(i)} - X_1^{(i)})(Y_3^{(i)} - Y_1^{(i)}) - (Y_2^{(i)} - Y_1^{(i)})(X_3^{(i)} - X_1^{(i)})} +$$

$$Y_1^{(i)} \frac{(Z_2^{(i)} - Z_1^{(i)})(X_3^{(i)} - X_1^{(i)}) - (X_2^{(i)} - X_1^{(i)})(Z_3^{(i)} - Z_1^{(i)})}{(X_2^{(i)} - X_1^{(i)})(Y_3^{(i)} - Y_1^{(i)}) - (Y_2^{(i)} - Y_1^{(i)})(X_3^{(i)} - X_1^{(i)})}$$

Structure of Wireframe Model

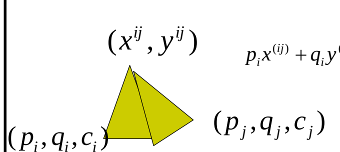
- Each triangular patch is either surrounded by two (if it is on the boundary of the wireframe) or three other triangles.



Neighboring patches must intersect at a straight line.

$$Z^{(i)} = p_i X^{(i)} + q_i Y^{(i)} + c_i$$

$$p_i x^{(ij)} + q_i y^{(ij)} + c_i = p_j x^{(ij)} + q_j y^{(ij)} + c_j$$



Main Points of Algorithm

- Stochastic relaxation.
- In each iteration visit all patches in a sequential order.
  - If, at present iteration none of neighboring patches of i have been visited yet, then p_i, q_i, c_i are all independently perturbed.
  - If, only one of the neighbor, j, has been visited, then two parameters, say p_i, q_i, are independent and perturbed. The dependent variable c_i is calculated from the equation:

$$c_i = p_j x^{(ij)} + q_j y^{(ij)} + c_j - p_i x^{(ij)} - q_i y^{(ij)}$$

Main Points of Algorithm

- If two of the neighboring patches, say j and k, have already been visited, i.e., the variables p_k, q_k, c_{ik} and p_j, q_j, c_j have been updated, then only one variable p_i is independent, and is perturbed. q_i, c_i can be evaluated as

$$c_i = p_j x^{(ij)} + q_j y^{(ij)} + c_j - p_i x^{(ij)} - q_i y^{(ij)}$$

$$q_i = \frac{p_k x^{(ik)} + q_k y^{(ik)} + c_k - p_i x^{(ik)} - c_i}{y^{(ik)}}$$

Main Points of Algorithm

- The perturbation of structure parameters (surface normal) for each patch, results in the change of coordinates (X,Y,Z) of the nodes of wireframe.

Updating of (X,Y,Z):

Patches i, j, k intersect at node n.

$$p_i X^{(n)} + q_i Y^{(n)} + c_i = p_j X^{(n)} + q_j Y^{(n)} + c_j$$

$$p_i X^{(n)} + q_i Y^{(n)} + c_i = p_k X^{(n)} + q_k Y^{(n)} + c_k$$

$$\begin{bmatrix} X^{(n)} \\ Y^{(n)} \end{bmatrix} = \begin{bmatrix} p_i - p_j & q_i - q_j \\ p_i - p_k & q_i - q_k \end{bmatrix}^{-1} \begin{bmatrix} c_j - c_i \\ -c_i + c_k \end{bmatrix}$$

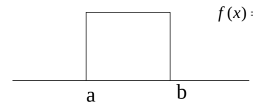
$$Z^{(i)} = p_i X^{(i)} + q_i Y^{(i)} + c_i$$

Algorithm

- Estimate light source direction
- Initialize coordinates of all nodes using approximately scaled wireframe model. Determine initial values of surface normals for each triangle.
- Determine initial motion parameters based on selected feature correspondences and their depth values from wireframe model. (Assume motion parameters.)
- (A) Compute the value of error function E.

- ~~If error E is less than some threshold, then stop~~
- Else
  - Perturb motion parameters (3 rotations and 2 translations) by random amount (zero mean Gaussian, s.d. equal to error E) (you can use uniform distribution)
  - Perturb structure parameters (p,q,c):
    - Perturb p, q, and c of first patch by adding random amount (zero mean Gaussian, s.d. equal to error E)
    - Increment count for all neighbors of patch-1 by 1

Uniform Distribution



$$f(x) = \frac{1}{(b-a)}$$

$$\bar{X} = mean = \frac{(a+b)}{2}$$

$$\sigma^2 = \text{variance} = \frac{(a-b)^2}{12}$$

Use rand() in C to generate random number between a range.

- For patch 2 to n
  - If the count==1
    - Perturb p and q
    - Compute c using equation for c_i
    - Increment the count

$$c_i = p_j x^{(ij)} + q_j y^{(ij)} + c_j - p_i x^{(ij)} - q_i y^{(ij)}$$

- If count==2
  - Perturb p_i
  - Compute c_i and q_i using equations

$$c_i = p_j x^{(ij)} + q_j y^{(ij)} + c_j - p_i x^{(ij)} - q_i y^{(ij)}$$

$$q_i = \frac{p_k x^{(ik)} + q_k y^{(ik)} + c_k - p_i x^{(ik)} - c_i}{y^{(ik)}}$$

  - Increment the count
- If p, q and c for at least three patches intersecting at node are updated, then update the coordinates of the node using equation.

$$\begin{bmatrix} X^{(n)} \\ Y^{(n)} \end{bmatrix} = \begin{bmatrix} p_i - p_j & p_j - p_k \\ q_i - q_j & q_j - q_k \end{bmatrix}^{-1} \begin{bmatrix} c_j - c_i \\ -c_j + c_k \end{bmatrix} \quad Z^{(i)} = p_i X^{(i)} + q_i Y^{(i)} + c_i$$

- Go to step (A)