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%

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>15</td>
<td>19</td>
<td>55</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>21</td>
<td>32</td>
</tr>
<tr>
<td>19</td>
<td>17</td>
<td>24</td>
<td>76</td>
</tr>
<tr>
<td>43</td>
<td>54</td>
<td>27</td>
<td>80</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>11</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>39</td>
<td>39</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>39</td>
<td>39</td>
<td>24</td>
<td>24</td>
</tr>
</tbody>
</table>

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>15</td>
<td>19</td>
<td>55</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>21</td>
<td>32</td>
</tr>
<tr>
<td>39</td>
<td>17</td>
<td>24</td>
<td>76</td>
</tr>
<tr>
<td>43</td>
<td>34</td>
<td>27</td>
<td>80</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>11</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>33</td>
<td>33</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td>33</td>
<td>33</td>
<td>51</td>
<td>51</td>
</tr>
</tbody>
</table>

Subsampling

- A better way - averaging
- Compression ratio 75%

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>15</td>
<td>19</td>
<td>55</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>21</td>
<td>32</td>
</tr>
<tr>
<td>39</td>
<td>17</td>
<td>24</td>
<td>76</td>
</tr>
<tr>
<td>43</td>
<td>34</td>
<td>27</td>
<td>80</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>11</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>33</td>
<td>33</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td>33</td>
<td>33</td>
<td>51</td>
<td>51</td>
</tr>
</tbody>
</table>

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 quantization, but the error can be recovered
### Truncation

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

#### Example

<table>
<thead>
<tr>
<th>9</th>
<th>11</th>
<th>17</th>
<th>21</th>
<th>0</th>
<th>10</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>51</td>
<td>33</td>
<td>14</td>
<td>10</td>
<td>50</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>19</td>
<td>23</td>
<td>18</td>
<td>15</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>53</td>
<td>47</td>
<td>12</td>
<td>43</td>
<td>50</td>
<td>40</td>
<td>10</td>
<td>40</td>
</tr>
</tbody>
</table>

### Rounding

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

#### Example

<table>
<thead>
<tr>
<th>13</th>
<th>19</th>
<th>9</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>10</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>17</td>
<td>8</td>
<td>15</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>52</td>
<td>49</td>
<td>53</td>
<td>47</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>50</td>
<td>58</td>
<td>51</td>
<td>42</td>
<td>50</td>
<td>60</td>
<td>50</td>
<td>40</td>
</tr>
</tbody>
</table>

### 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 finely 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

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

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

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

DCT Bases Functions

Example

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

image

Example

\[
F_{i,j} = \begin{bmatrix}
1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1 \\
-23 & -17 & -6 & -3 & -3 & 0 & 0 & 1 \\
-11 & -9 & -2 & 2 & 0 & -1 & -1 & 0 \\
-7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\
-1 & -1 & 1 & 1 & 0 & -1 & 1 & 1 \\
2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\
-1 & 0 & 0 & -1 & 0 & 2 & 1 & -1 \\
-3 & 2 & -4 & -2 & 2 & 1 & -1 & 0
\end{bmatrix}
\]

DCT
JPEG 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.

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)

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

Example (Encoding)

Example (Decoding)
Comparison

<table>
<thead>
<tr>
<th>Original Image</th>
<th>Decoded Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>52 55 61 66 70 61 64 73</td>
<td>58 64 67 64 59 62 70 78</td>
</tr>
<tr>
<td>63 59 66 90 109 85 69 72</td>
<td>56 55 67 89 98 88 74 69</td>
</tr>
<tr>
<td>62 59 68 113 144 104 66 73</td>
<td>69 50 70 119 141 116 80 64</td>
</tr>
<tr>
<td>63 58 71 122 154 106 70 69</td>
<td>69 51 71 128 149 115 77 68</td>
</tr>
<tr>
<td>67 61 68 104 126 88 68 70</td>
<td>74 53 64 105 115 84 65 72</td>
</tr>
<tr>
<td>79 65 60 70 77 68 58 75</td>
<td>76 57 56 74 75 57 57 74</td>
</tr>
<tr>
<td>85 71 64 59 55 61 65 83</td>
<td>83 69 59 60 61 61 67 78</td>
</tr>
<tr>
<td>87 79 69 68 65 76 78 84</td>
<td>93 81 67 62 69 80 84 84</td>
</tr>
</tbody>
</table>

JPEG

Original 64K  13K  5K

Difference

MPEG-1 Encoder

<table>
<thead>
<tr>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>−6 −9 −6 2 11 −1 −6 −5</td>
</tr>
<tr>
<td>7 4 −1 1 11 −3 −5 3</td>
</tr>
<tr>
<td>2 9 −2 −6 −3 −12 −14 9</td>
</tr>
<tr>
<td>−6 7 0 −4 −5 −9 −7 1</td>
</tr>
<tr>
<td>−7 8 4 −1 11 4 3 2</td>
</tr>
<tr>
<td>3 8 4 −4 2 11 1 1</td>
</tr>
<tr>
<td>2 2 5 −1 −6 0 −2 5</td>
</tr>
<tr>
<td>−6 −2 2 6 −4 −4 −6 10</td>
</tr>
</tbody>
</table>

Rate control

Input Image

Prediction Error

DCT

Q

Q−1

TO VLC

Predicted Image

Motion Estimation

Motion Compensation

Frame Memory-1

Frame Memory-2

Reconstructed Image

DCT−1

Motion Vectors

0
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

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

- 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

- 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.

Face Model

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

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 \( \frac{\sqrt{d^2 + e^2}}{f} \) depth.
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

Video Phones
Motion Estimation

Perspective Projection (optical flow)

\begin{align*}
    u &= f \left( \frac{V_1}{Z} + \Omega_2 \right) - \frac{V_3}{Z} x - \Omega_3 y - \frac{\Omega_1}{f} x y + \frac{\Omega_2}{f} x^2 \\
    v &= f \left( \frac{V_2}{Z} - \Omega_1 \right) + \Omega_3 x - \frac{V_3}{Z} y + \frac{\Omega_2}{f} x y - \frac{\Omega_1}{f} y^2
\end{align*}

Optical Flow Constraint Eq

\[ f_x u + f_y v + f_t = 0 \]
\[ f_x (\frac{V}{Z} + \Omega_2) - \frac{V}{Z} x - \Omega_1 y - \Omega \frac{y}{f} x^2 + \frac{\Omega}{f} x^2 + f_x \]

\[ f_y (\frac{V}{Z} - \Omega_2) + \Omega_2 x - \frac{V}{Z} y + \Omega \frac{x y}{f} - \frac{\Omega}{f} y^2 + f_y = 0 \]

\[ (f_x, \frac{f_y}{f}) \Omega_z + (f_y, \frac{f_x}{f}) \Omega_z + (\frac{f_y}{f} (f_x - f_y)) \Omega_z + \]

\[ (-f_y, \frac{f_x}{f} - f, f) \Omega_x + (f_y, f + f, \frac{f_x}{f}) \Omega_x + 
\]

\[ (f_x, y + f, x) \Omega_z = -f_x \]

\[ \mathbf{A} \mathbf{x} = \mathbf{b} \quad \text{Solve by Least Squares} \]

\[ \mathbf{x} = (V_x, V_z, \Omega_x, \Omega_z, \Omega_2) \]

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

\[
X' = RX + T
\]

\[
\begin{bmatrix}
X' \\
Y' \\
Z'
\end{bmatrix} =
\begin{bmatrix}
1 & -\alpha & \beta \\
\alpha & 1 & -\gamma \\
-\beta & \gamma & 1
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} +
\begin{bmatrix}
T_x \\
T_y \\
T_z
\end{bmatrix}
\]

3-D Rigid+Non-rigid Motion

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

Facial expressions

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

\[
X' = RX + T + \Phi
\]
3-D Rigid+Non-rigid Motion

\[
\begin{bmatrix}
X' \\
Y' \\
Z'
\end{bmatrix} = 
\begin{bmatrix}
1 - \alpha & \beta & 0 \\
\alpha & 1 & -\gamma \\
-\beta & \gamma & 1
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} + T_x + \sum_{i=1}^n E_i(X)\phi_i
\]

\[
\begin{bmatrix}
X' \\
Y' \\
Z'
\end{bmatrix} = 
\begin{bmatrix}
1 - \alpha & \beta & 0 \\
\alpha & 1 & -\gamma \\
-\beta & \gamma & 1
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} + T_x + \sum_{i=1}^n E_i(X)\phi_i
\]

\[
\begin{bmatrix}
X' - X \\
Y' - Y \\
Z' - Z
\end{bmatrix} = 
\begin{bmatrix}
0 & -\alpha & \beta \\
\alpha & 0 & -\gamma \\
-\beta & \gamma & 0
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} + T_x + \sum_{i=1}^n E_i(X)\phi_i
\]

Perspective Projection (arbitrary flow)

\[
x = \frac{fX}{Z}
\]
\[
y = \frac{fY}{Z}
\]
\[
u = \dot{z} = \frac{f(Z\dot{X} - fX\dot{Z})}{Z^2} = f\frac{\dot{X}}{Z} - \frac{\dot{Z}}{Z}
\]
\[
v = \dot{y} = \frac{f(Z\dot{Y} - fY\dot{Z})}{Z^2} = f\frac{\dot{Y}}{Z} - \frac{\dot{Z}}{Z}
\]
Perspective Projection (arbitrary flow)

\[
\begin{align*}
\hat{x} &= \omega y_{x} + \omega z_{x} + \sum_{\omega} e_{\omega} \\
\hat{y} &= \omega y_{y} + \omega z_{y} + \sum_{\omega} e_{\omega} \\
\hat{z} &= \omega y_{z} + \omega z_{z} + \sum_{\omega} e_{\omega}
\end{align*}
\]

\[
\begin{align*}
u &= \frac{f \hat{x} X - f \hat{X} \hat{Z}}{Z} = f \frac{\hat{x}}{Z} - \frac{\hat{X}}{Z} \\
v &= \frac{f \hat{y} Y - f \hat{Y} \hat{Z}}{Z} = f \frac{\hat{y}}{Z} - \frac{\hat{Y}}{Z}
\end{align*}
\]

Optical Flow Constraint Eq

\[
f_{x} u + f_{y} v + f_{t} = 0
\]

Ax = b

\[
x = (V_1, V_2, V_3, \Omega_1, \Omega_2, \Omega_3, \phi_1, \phi_2, \ldots, \phi_m)
\]
Estimation Using Flexible Wireframe Model

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

Generalized Optical Flow Constraint

\[ f(x, y, t) = \rho N(t) \cdot L \]
\[ \frac{df(x, y, t)}{dt} = \rho L \cdot \frac{dN}{dt} \]
\[ f_x u + f_y v + f_t = \rho L \cdot \frac{dN}{dt} \]

Lambertian Model

\[ f(x, y) = n \cdot L = (n_x, n_y, n_z) \cdot (l_x, l_y, l_z) \]
\[ f(x, y) = n \cdot L = \frac{1}{\sqrt{p^2 + q^2 + 1}} \cdot (-p, -q, 1) \cdot (l_x, l_y, l_z) \]
Sphere

\[ z = \sqrt{R^2 - x^2 - y^2} \]

\[ p = \frac{\partial z}{\partial x} = -\frac{x}{z} \]

\[ q = \frac{\partial z}{\partial y} = -\frac{y}{z} \]

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

Vase

\[ X = \Omega \times X + V \]

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

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

\[ \dot{Z} = \Omega_1 Y - \Omega_2 X + V_1 \]

Orthographic Projection

\((u,v)\) is optical flow

\[ X = \Omega \times X + V \]

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

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

\[ \dot{Z} = \Omega_1 Y - \Omega_2 X + V_1 \]
Optical flow equation

\[ f_i(\Omega_2 Z - \Omega_2 y + V_i) + f_i(\Omega_2 x - \Omega_2 Z + V_2) + f_i = \rho L \frac{dN}{dt} \]

\[ f_i(\Omega_2 Z - \Omega_2 y + V_i) + f_i(\Omega_2 x - \Omega_2 Z + V_2) + f_i = \rho L \left[ \frac{(-p' - q' L)^T}{\sqrt{p'^2 + q'^2 + 1}} - \frac{(-p - q L)^T}{\sqrt{p^2 + q^2 + 1}} \right] \]

Error Function

\[ f_i(\Omega_2 Z - \Omega_2 y + V_i) + f_i(\Omega_2 x - \Omega_2 Z + V_2) + f_i = \rho L \left[ \frac{(-p' - q' L)^T}{\sqrt{p'^2 + q'^2 + 1}} - \frac{(-p - q L)^T}{\sqrt{p^2 + q^2 + 1}} \right] \]

\[ e_i(x, y) = f_i(\Omega_2 y - \Omega_2 (p x + q y + c)) + V_i + f_i(\Omega_2 x + \Omega_2 (p x + q y + c)) + V_i + f_i \left( \frac{-\Omega_2 x + p \Omega_2 z - \Omega_2 x + q \Omega_2 y}{1 + \Omega_2 p^2 - 1 - \Omega_2 q^2} \right) \]

\[ \left( \frac{-\Omega_2 x + q \Omega_2 y}{1 + \Omega_2 p^2 - 1 - \Omega_2 q^2} \right)^{1/2} \]

\[ \frac{(-p' - q' L)^T}{\sqrt{p'^2 + q'^2 + 1}} \]

Error Function

\[ E = \sum_i \sum_{(x, y)} e_i^2 \]

\[ p_i x_i^{(i)} + q_i x_i^{(i)} + c_i = p_i x_i^{(i)} + q_i x_i^{(i)} + c_i \]

Constraint: Neighboring patches
Should intersect at straight line

Equation of a Planar Patch

\[ P_1^{(i)} = (X_1^{(i)}, Y_1^{(i)}, Z_1^{(i)}) \]

\[ P_2^{(i)} = (X_2^{(i)}, Y_2^{(i)}, Z_2^{(i)}) \]

\[ P_3^{(i)} = (X_3^{(i)}, Y_3^{(i)}, Z_3^{(i)}) \]

\[ P^{(i)} = (X^{(i)}, Y^{(i)}, Z^{(i)}) \]

\[ P_1^{(i)} P_2^{(i)} P_3^{(i)} (P_1^{(i)} P_2^{(i)} P_3^{(i)}) = 0 \]
Equation of a Planar Patch

\[ Z^{(i)} = p_i X^{(i)} + q_i Y^{(i)} + c_i \]

Homework 4.2
Show this.

\[
\begin{align*}
p_i &= - \frac{(Z^{(j)}_2 - Y^{(j)}_2)(Z^{(j)}_3 - Y^{(j)}_3) - (Z^{(i)}_2 - Y^{(i)}_2)(Z^{(i)}_3 - Y^{(i)}_3)}{(X^{(j)}_1 - X^{(j)}_3)(Y^{(j)}_1 - Y^{(j)}_3) - (X^{(j)}_2 - X^{(j)}_3)(Y^{(j)}_2 - Y^{(j)}_3)} \\
q_i &= - \frac{(Z^{(j)}_2 - Y^{(j)}_2)(X^{(j)}_1 - X^{(j)}_3) - (Z^{(i)}_2 - Y^{(i)}_2)(X^{(i)}_1 - X^{(i)}_3)}{(X^{(j)}_1 - X^{(j)}_3)(Y^{(j)}_1 - Y^{(j)}_3) - (X^{(j)}_2 - X^{(j)}_3)(Y^{(j)}_2 - Y^{(j)}_3)} \\
c_i &= Z^{(i)}_1 + X^{(i)}_1 \frac{(Z^{(j)}_2 - Y^{(j)}_2)(Z^{(j)}_3 - Y^{(j)}_3) - (Z^{(i)}_2 - Y^{(i)}_2)(Z^{(i)}_3 - Y^{(i)}_3)}{(X^{(j)}_1 - X^{(j)}_3)(Y^{(j)}_1 - Y^{(j)}_3) - (X^{(j)}_2 - X^{(j)}_3)(Y^{(j)}_2 - Y^{(j)}_3)} \\
&\quad + (X^{(i)}_2 - X^{(i)}_3)(Z^{(i)}_3 - Z^{(i)}_2)(X^{(i)}_1 - X^{(i)}_3) \frac{(Z^{(j)}_2 - Y^{(j)}_2)(X^{(j)}_1 - X^{(j)}_3) - (Z^{(i)}_2 - Y^{(i)}_2)(X^{(i)}_1 - X^{(i)}_3)}{(X^{(j)}_1 - X^{(j)}_3)(Y^{(j)}_1 - Y^{(j)}_3) - (X^{(j)}_2 - X^{(j)}_3)(Y^{(j)}_2 - Y^{(j)}_3)} \end{align*}
\]

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.

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 \( j \) have been visited yet, then \( p_j, q_j, c_i \) are all independently perturbed.
  - If, only one of the neighbor, \( i \), has been visited, then two parameters, say \( p_j, q_j \), are independent and perturbed. The dependent variable \( c_i \) is calculated from the equation:

\[
c_i = p_j x^{(j)} + q_j y^{(j)} + c_j - p_j x^{(j)} - q_j y^{(j)}
\]
Main Points of Algorithm

- If two of the neighboring patches, say \( j \) and \( k \), have already been visited, i.e., the variables \( p_x, q_x, c_x \) and \( p_y, q_y, c_y \) have been updated, then only one variable \( p_i, q_i, c_i \) can be evaluated as

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

\[
q_i = p_x x^{(ik)} + q_y y^{(ik)} + c_i - p_i x^{(ik)} - c_i
\]

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 \( n \).

\[
\begin{align*}
p_i x^{(in)} + q_i y^{(in)} + c_i &= p_j x^{(in)} + q_j y^{(in)} + c_j \\
p_i x^{(in)} + q_i y^{(in)} + c_i &= p_k x^{(in)} + q_k y^{(in)} + c_k \\
X^{(i)} &= \begin{bmatrix} p_i - p_j & q_i - q_j & c_i - c_j \\
& & & \\
p_i - p_k & q_i - q_k & -c_i + c_k \\
& & & \\
Z^{(i)} &= p_i X^{(i)} + q_i Y^{(i)} + c_i
\end{bmatrix}
\end{align*}
\]

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

For patch 2 to n
- If the count=1
  - Perturb p and q
  - Compute c using equation for ci
  - Increment the count
  \[ c_i = p_i x_i^{(i)} + q_i y_i^{(i)} + c_i - p_j x_j^{(i)} - q_j y_j^{(i)} \]

- If count=2
  - Perturb pi
  - Compute ci and qi using equations
  \[ c_i = p_i x_i^{(i)} + q_i y_i^{(i)} + c_i - p_j x_j^{(i)} - q_j y_j^{(i)} \]
  \[ q_i = p_i x_i^{(i)} + q_i y_i^{(i)} + c_i - p_j x_j^{(i)} - c_i \]
  - 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.

\[
\begin{bmatrix}
X^{(i)} \\
Y^{(i)}
\end{bmatrix} = \begin{bmatrix}
p_i & p_j - p_i & p_j \\
q_i & q_j - q_i & q_j
\end{bmatrix} \begin{bmatrix}
c_i - c_j \\
q_j - q_i
\end{bmatrix} + \begin{bmatrix}
c_i + 2c_i \\
-q_i
\end{bmatrix} = p_i X^{(i)} + q_i Y^{(i)} + c_i
\]

Go to step (A)