When we cannot apply BCE

- The relationship between $x$, $y$, and $t$ derivatives has to hold
  \[ f_x u + f_y v + f_t = 0 \]
- When motion happens, all derivatives may be affected
- If motion is very large, derivative window might be too small to capture it

For large motions...

- A larger derivative mask may be used
- OR, smaller images may be used and operations done incrementally
- PYRAMIDS... related to wavelet transform

Pyramids

- Very useful image representation
- Pyramid representation has multiple copies of the image
- Each "level" is $\frac{1}{4}$ the size of the previous level
- Lowest level is highest resolution
- Highest level is lowest resolution

Pyramid – Reduce Operation

\[ g_l(i, j) = \sum_{m=0}^{2} \sum_{n=0}^{2} w(m, n) g_{l-1}(2i+m, 2j+n) \]

\[ g_l = REDUCE[g_{l-1}] \]
Pyramid – Expand Operation

\[ g_{l,0}(i, j) = \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} w(p, q) g_{l-1}(i/p, j/q) \]

\[ g_{l,a} = \text{EXPAND}[g_{l-1,a}] \]

Reduce – 1D

\[ g_i(i) = \sum_{m=-\infty}^{\infty} w(m) g_{i/2+1}(2i+m) \]

\[ g_i(2) = \hat{w}(-2) g_{i/2+1}(4-2) + \hat{w}(-1) g_{i/2+1}(4-1) + \hat{w}(0) g_{i/2+1}(4) + \hat{w}(1) g_{i/2+1}(4+1) + \hat{w}(2) g_{i/2+1}(4+2) \]

Pyramids... implementation

Reduce Operation

Gaussian Pyramids

- Where weights are distributed according to a Gaussian Distribution
- Special property that Gaussian masks are separable

\[ w(n, m) = \hat{w}(n) \otimes \hat{w}(m) \]

\[ \{0.05 0.25 0.4 0.25 0.05\} \]

Generating 2D Pyramid

- Since Gaussian is separable,
- Apply mask to alternate pixels in row direction
- Apply mask to alternate columns of resultant image

Pyramids
Pyramids

Reduce followed by Expand
Why not just simply sub-sample?

- Aliasing will occur on simple sub-sampling
- Follows from Nyquist Theorem (for those who have taken a DSP class)
- Example from MATLAB
- 'Expand' can be substituted by Bilinear interpolation – filtering step would be discarded

Lucas & Kanade (Least Squares)

- Optical flow eq
  \[ f_x u + f_y v = -f_t \]
- Consider 3 by 3 window
  \[
  \begin{bmatrix}
  f_{xx} & f_{xy} & -f_x \\
  f_{yx} & f_{yy} & -f_y \\
  f_{ex} & f_{ey} & -f_e
  \end{bmatrix}
  \]
  \[ Au = f_t \]

Problem with Standard LK

- Motions that may be captured are limited by derivative window size
  - which is generally 3x3 or 5x5

Lucas & Kanade

\[
\begin{bmatrix}
  u \\
  v
\end{bmatrix} = \frac{1}{\sum f_{x,i}^2 - \sum f_{x,i} f_{y,i} - \sum f_{x,i} f_{y,i}}
\begin{bmatrix}
  \sum f_{x,i} f_{x,i} & \sum f_{x,i} f_{y,i} \\
  \sum f_{y,i} f_{x,i} & \sum f_{y,i} f_{y,i}
\end{bmatrix}^{-1}
\begin{bmatrix}
  -\sum f_{x,i} f_t \\
  -\sum f_{y,i} f_t
\end{bmatrix}
\]
Lucas-Kanade with Pyramids

- Compute 'simple' LK at highest level
- At level $i$
  - Take flow $u_{i-1}, v_{i-1}$ from level $i-1$
  - Bilinear interpolate (or expand) it, multiply by 2 to create $u^*, v^*$ matrices of twice resolution
  - Compute $f_t$ from a block displaced by $u^*(x,y), v^*(x,y)$
  - Apply LK to get $u'(x, y), v'(x, y)$ (the correction in flow)
  - Add correction to $u^*, v^*$ to get $u_i, v_i$