Parallel Video Processing

Neal Wadhwa

December 10, 2012

Parallel Video Processing

- Processing is on uncompressed video

Parallel Video Processing

- Processing is on uncompressed video
- Uncompressed videos uses huge amounts of space.

Parallel Video Processing

- Processing is on uncompressed video
- Uncompressed videos uses huge amounts of space.
- 1080p at 30 FPS is one gigabyte per second

Parallel Video Processing

- Processing is on uncompressed video
- Uncompressed videos uses huge amounts of space.
- 1080p at 30 FPS is one gigabyte per second
- Lots of algorithms are easy to parallelize due to independence of processing in space or time.

Example: Motion Magnification

- DSP based method to magnify subtle motions

Example: Motion Magnification

- DSP based method to magnify subtle motions
- Here are some cool examples of motion magnification.

Example: Motion Magnification

- DSP based method to magnify subtle motions
- Here are some cool examples of motion magnification.
- Switch to video.

Example: Motion Magnification

- DSP based method to magnify subtle motions
- Here are some cool examples of motion magnification.
- Switch to video.
- FFT-based algorithm lends itself to being parallelized.

Example: Motion Magnification

- DSP based method to magnify subtle motions
- Here are some cool examples of motion magnification.
- Switch to video.
- FFT-based algorithm lends itself to being parallelized.
- Try to parallelize and see how far we can get

Outline of algorithm - three stages

- 1. Spatially decompose each frame.

Outline of algorithm - three stages

- 1. Spatially decompose each frame.
- 2. Temporally process each pixel in each decomposition level

Outline of algorithm - three stages

- 1. Spatially decompose each frame.
- 2. Temporally process each pixel in each decomposition level
- 3. Reconstruct each frame

Outline of algorithm - three stages

- 1. Spatially decompose each frame.
- 2. Temporally process each pixel in each decomposition level
- 3. Reconstruct each frame
- Every stage is easy to parallelize individually

Outline of algorithm - three stages

- 1. Spatially decompose each frame.
- 2. Temporally process each pixel in each decomposition level
- 3. Reconstruct each frame
- Every stage is easy to parallelize individually
- Serial algorithm takes several hours on high resolution videos.

Outline of algorithm - Spatial Decomposition

- Say you have frames $F_1, \ldots, F_n$

Outline of algorithm - Spatial Decomposition

- Say you have frames $F_1, \ldots, F_n$
- Decompose each frame into different spatial bands

$$F_i \to (D_{i,1}, \ldots, D_{i,k})$$

uses k times as much space as the original frame.

Outline of algorithm - Spatial Decomposition

- Say you have frames $F_1, \ldots, F_n$
- Decompose each frame into different spatial bands

$$F_i \rightarrow (D_{i,1}, \ldots, D_{i,k})$$

uses k times as much space as the original frame.

- Decomposition is performing by FFT, multiplying by filters and applying IFFT

$$D_{i,j} = \mathcal{F}^{-1}\{T_j \times \mathcal{F}\{F_i\}\}$$

Outline of algorithm - Spatial Decomposition

- Say you have frames $F_1, \ldots, F_n$
- Decompose each frame into different spatial bands

  $$F_i \to (D_{i,1}, \ldots, D_{i,k})$$

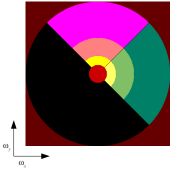
  uses k times as much space as the original frame.
- Decomposition is performing by FFT, multiplying by filters and applying IFFT

  $$D_{i,j} = \mathcal{F}^{-1}\{T_j \times \mathcal{F}\{F_i\}\}$$

- Transform is invertible.

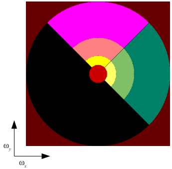
Outline of algorithm - Spatial Decomposition

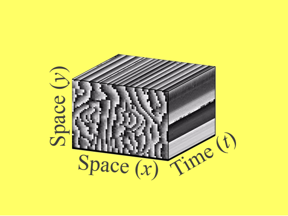
- Create decomposition for every frame



Outline of algorithm - Spatial Decomposition

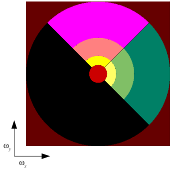
- Create decomposition for every frame

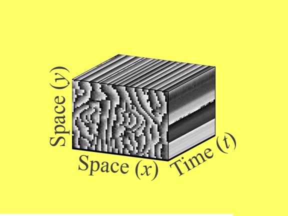


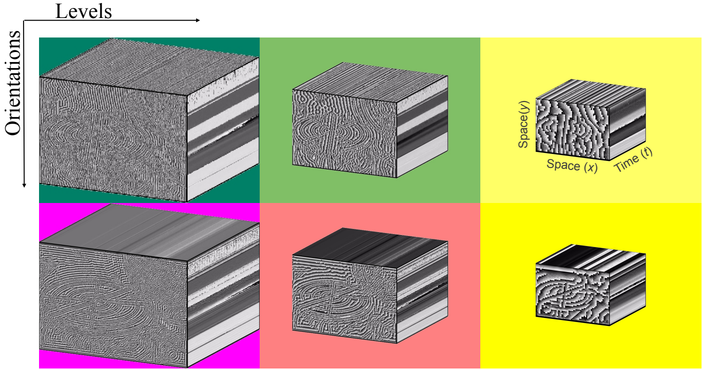


Outline of algorithm - Spatial Decomposition

- Create decomposition for every frame

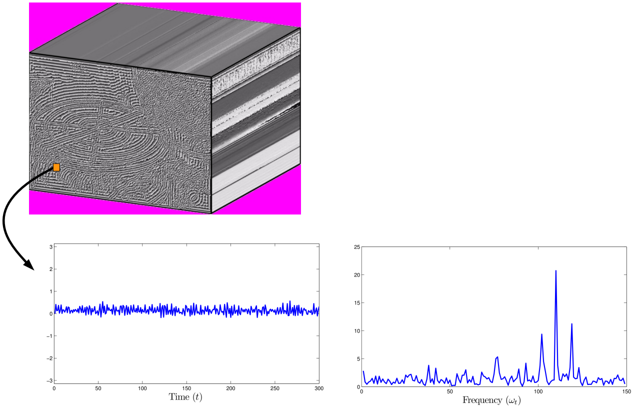






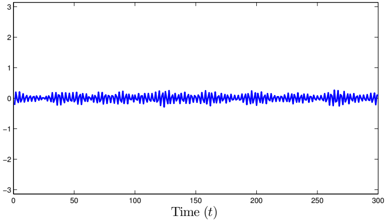
Outline of Algorithm

- For every pixel in every level, values contain motion signal

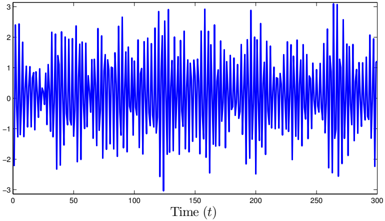


Outline of Algorithm

- Bandpass from 100 Hz to 120Hz
- Add bandpassed signal to original signal



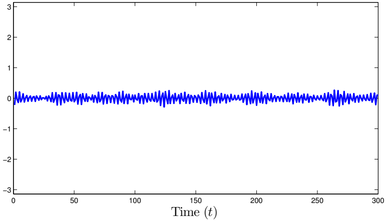
Bandpass 100-120 Hz



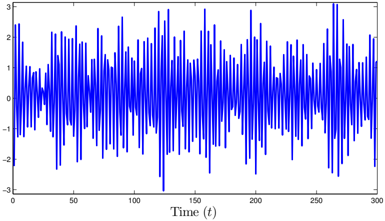
Magnified)

Outline of Algorithm

- Bandpass from 100 Hz to 120Hz
- Add bandpassed signal to original signal



Bandpass 100-120 Hz



Magnified)

- Amplifies only selected frequency

Easy to Parallelize

- Parallelize spatial decomposition over frames

Easy to Parallelize

- Parallelize spatial decomposition over frames
- Parallelize temporal filtering over pixels.

Easy to Parallelize

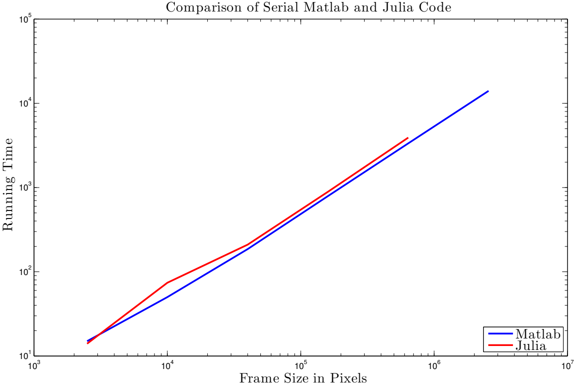
- Parallelize spatial decomposition over frames
- Parallelize temporal filtering over pixels.
- Difficulty lies in how to store data over cores.

Matlab vs. Julia

- Relatively easy to port code to Julia

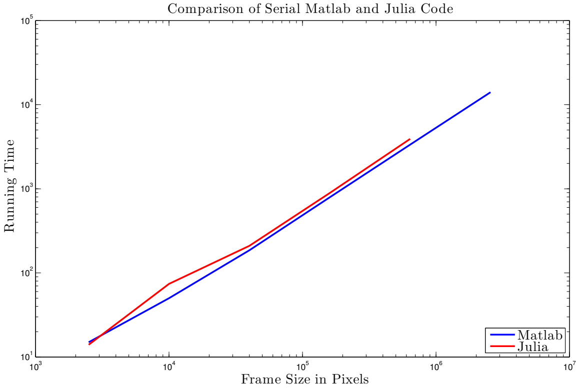
Matlab vs. Julia

- Relatively easy to port code to Julia
- Compare serial performance of matlab vs. julia at different image sizes.



Matlab vs. Julia

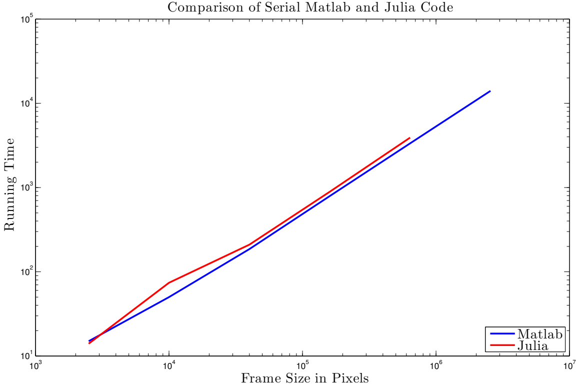
- Relatively easy to port code to Julia
- Compare serial performance of matlab vs. julia at different image sizes.



- Julia is slightly slower, but comparable.

Matlab vs. Julia

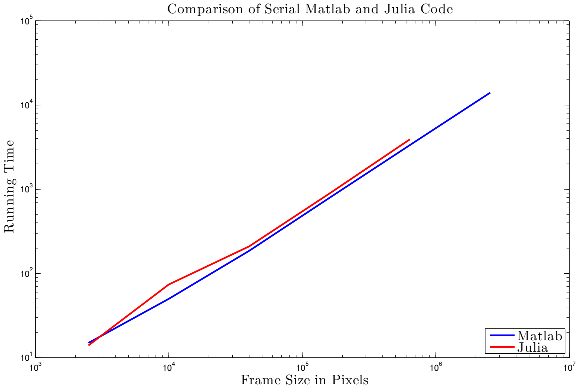
- Relatively easy to port code to Julia
- Compare serial performance of matlab vs. julia at different image sizes.



- Julia is slightly slower, but comparable.
- Not surprising since main processing occurs in ffts (in libfftw).

Matlab vs. Julia

- Relatively easy to port code to Julia
- Compare serial performance of matlab vs. julia at different image sizes.



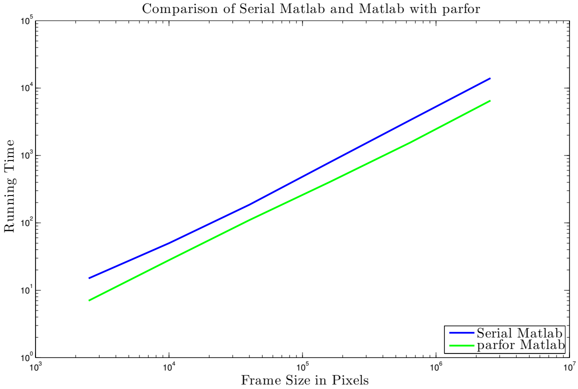
- Julia is slightly slower, but comparable.
- Not surprising since main processing occurs in ffts (in libfftw).
- Uses 400 GB at largest problem size, 1600x1600x300.

Matlab Parfor

- Parfor gives factor of two improvement when used with 12 cores.

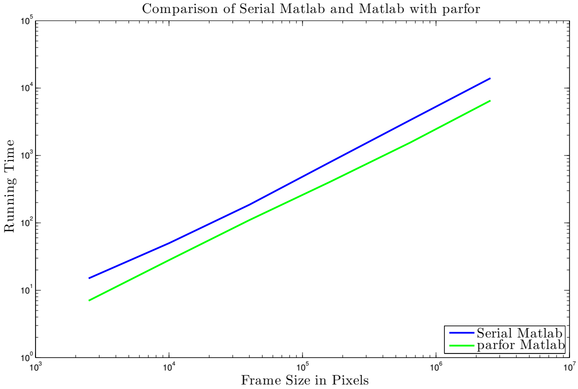
Matlab Parfor

- Parfor gives factor of two improvement when used with 12 cores.
- Parfor processing on frames and on temporal processing



Matlab Parfor

- Parfor gives factor of two improvement when used with 12 cores.
- Parfor processing on frames and on temporal processing



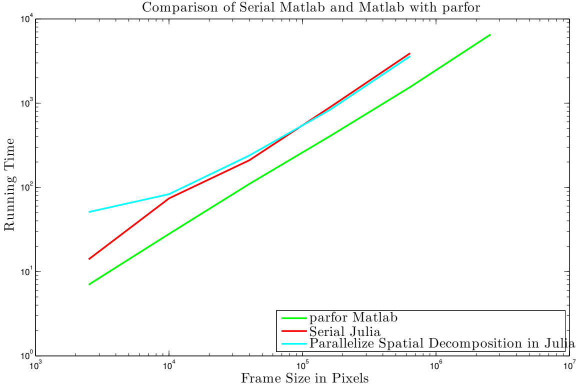
- Only 2x improvement

Julia spawnat vs. Matlab parfor

- Parllelize the spatial decomposition and reconstruction in Julia

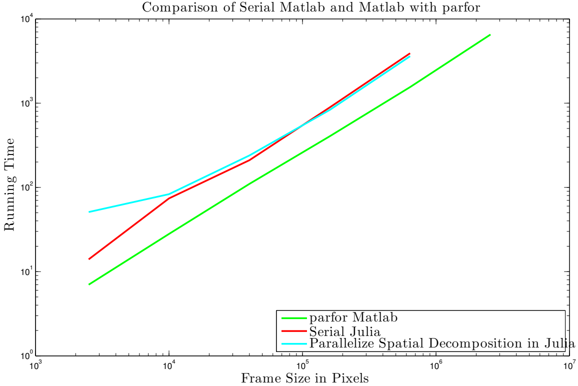
Julia spawnat vs. Matlab parfor

- Parllelize the spatial decomposition and reconstruction in Julia
- Faster than serial Julia for large problem size.



Julia spawnat vs. Matlab parfor

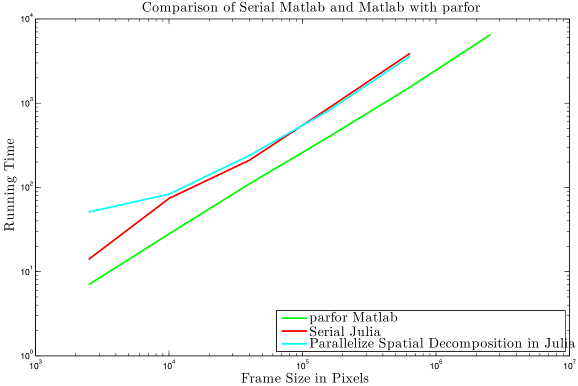
- Parllelize the spatial decomposition and reconstruction in Julia
- Faster than serial Julia for large problem size.



- The spatial decomposition is extremely fast, but reordered data for temporal filtering is very, very slow.

Julia spawnat vs. Matlab parfor

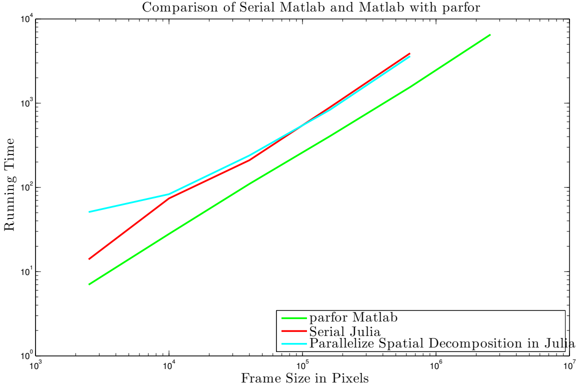
- Parllelize the spatial decomposition and reconstruction in Julia
- Faster than serial Julia for large problem size.



- The spatial decomposition is extremely fast, but reordered data for temporal filtering is very, very slow.
- In serial code, temporal processing uses 14% of time.

Julia spawnat vs. Matlab parfor

- Parllelize the spatial decomposition and reconstruction in Julia
- Faster than serial Julia for large problem size.



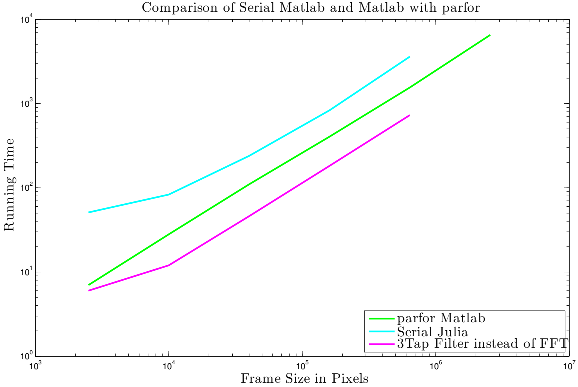
- The spatial decomposition is extremely fast, but reordered data for temporal filtering is very, very slow.
- In serial code, temporal processing uses 14% of time.
- In parallel code, temporal processing uses 50% of time.

Change processing to use 3-tap primal domain temporal filter

- Makes temporal processing more local to avoid communication overhead.

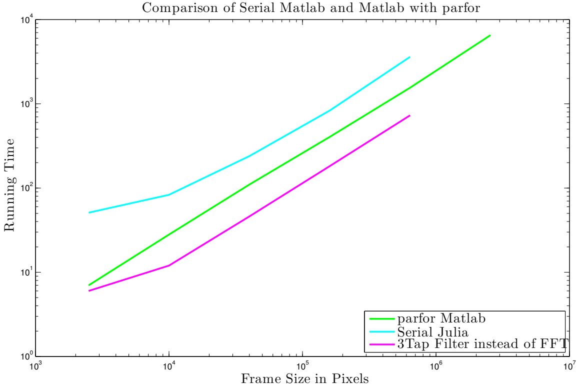
Change processing to use 3-tap primal domain temporal filter

- Makes temporal processing more local to avoid communication overhead.
- Store temporally close pixels on same processors



Change processing to use 3-tap primal domain temporal filter

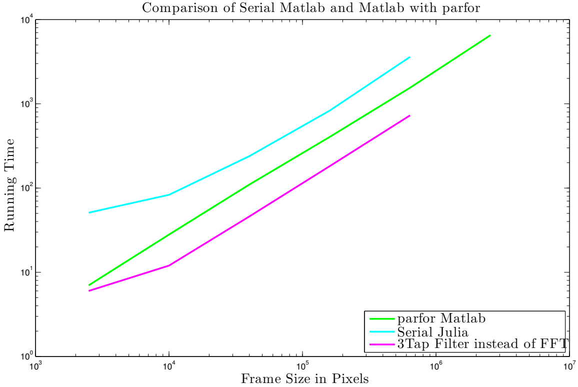
- Makes temporal processing more local to avoid communication overhead.
- Store temporally close pixels on same processors



- 2.5x faster than Matlab, 5x faster than serial Julia

Change processing to use 3-tap primal domain temporal filter

- Makes temporal processing more local to avoid communication overhead.
- Store temporally close pixels on same processors



- 2.5x faster than Matlab, 5x faster than serial Julia
- Matlab parfor fails to capitalize on this