Parallelizing The Adaptive Vector Median Filter.....progress.....

Rajkiran Natarajan

Review

- Last presentation, the Vector Median Filter (VMF) was introduced. I extended my work to parallelize the Adaptive Vector Median Filter (AVMF).
- The VMF applies a filtering operation to each pixel in the input image indiscriminately. Signal pixels (true pixels) will also get filtered. The AVMF attempts to discriminate signal pixels from noise and only filters the pixel if it is suspected to be noise.
- The VMF degrades sharpness of the image because all pixels are filtered, while the AVMF only filters noisy pixels and leaves signal pixels untouched
- Edge preserving ability measured by Mean Average Error:

$$MAE = \frac{1}{3 \cdot M \cdot N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left[|R(i,j) - \hat{R}(i,j)| + |G(i,j) - \hat{G}(i,j)| + |B(i,j) - \hat{B}(i,j)| \right]$$

Review – The VMF vs the AVMF
Original Image

Review – The VMF vs the AVMF
AVMF(Noisy Image, size 3 nhood)

MAE = 0.0053

Review – The VMF vs the AVMF
VMF(Noisy Image, size 5 nhood)

MAE = 0.0093

Review – The VMF vs the AVMF
VMF(Noisy Image, size 7 nhood)

MAE = 0.0117

Review – The VMF vs the AVMF
VMF(Noisy Image, size 9 nhood)

MAE = 0.0140

Review – The VMF vs the AVMF
With the AVMF

Size 3 nhood : MAE = 0.000533 => ~10x
Size 5 nhood : MAE = 0.0010 => ~9x
Size 7 nhood : MAE = 0.0015 => ~8x
Size 9 nhood : MAE = ~9x

VMF with size 9 neighbourhood

AVMF with size 11 neighbourhood

AVMF Algorithm

- For each pixel:
  - Apply the VMF and get ranking of aggregate vector distances
  - Is this pixel noise?
    - If noise, replace pixel with result of VMF filter
    - Else, if its signal, leave it alone

$$\text{IF } Val \geqslant Tol \quad \text{THEN } \boldsymbol{x}_{(N+1)/2} \text{ is impulse}$$
$$\text{ELSE } \boldsymbol{x}_{(N+1)/2} \text{ is noise-free}$$

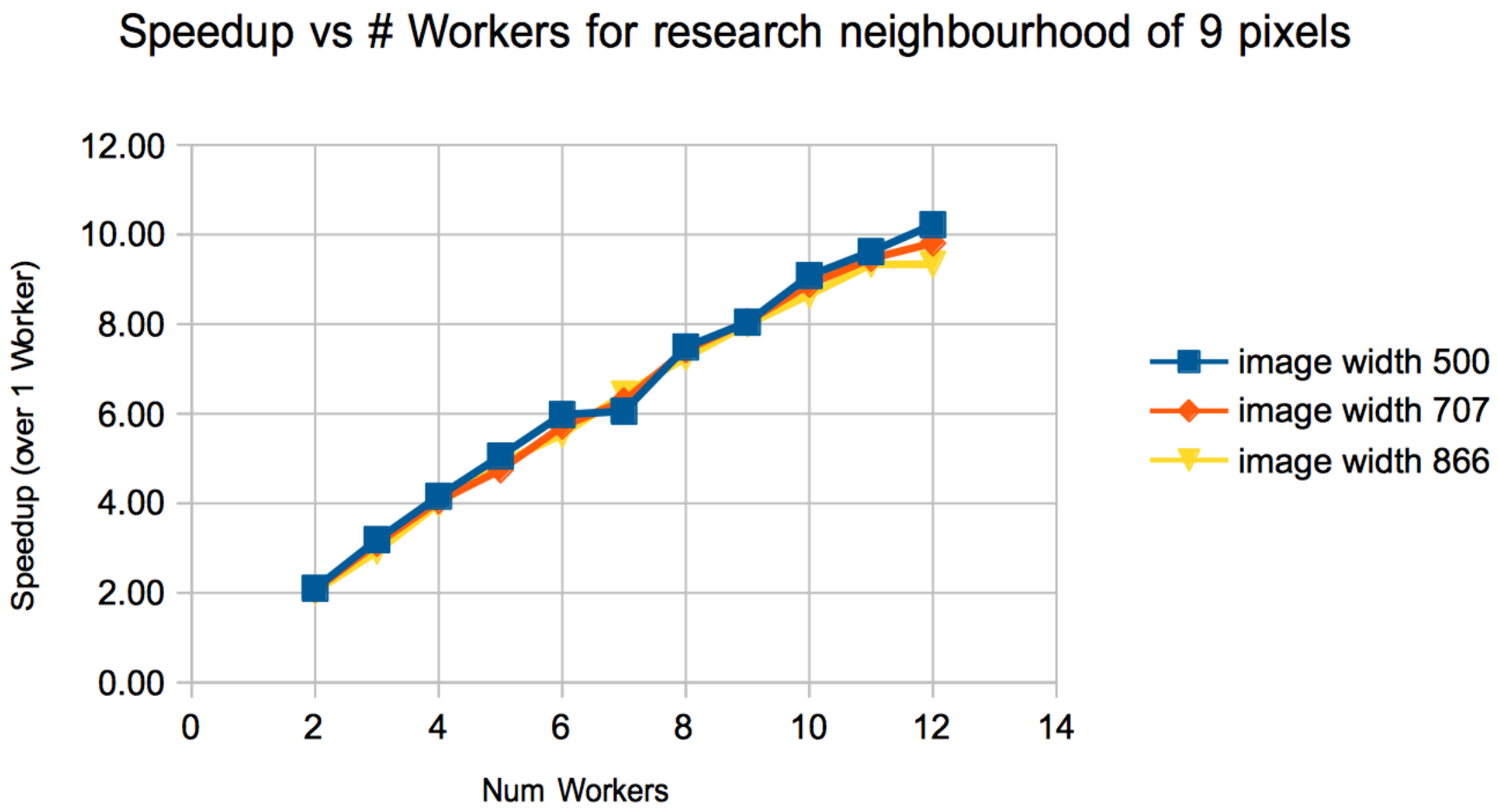
$$Val = \left\| \boldsymbol{x}_{(N+1)/2} - \frac{1}{r} \sum_{i=1}^{r} \boldsymbol{x}^{(i)} \right\|_{\gamma}$$

Progress since last time.....

- The AVMF was implemented and tested
  - The VMF kernel was vectorized (not yet enhanced with GPU accelerated functions)
  - The outer loop was parallelized with parfor
  - Tested on Zeno last Friday
  - Excellent results thus far in terms of speedup and efficiency for image sizes generally tested in literature

Progress since last time.....preliminary results

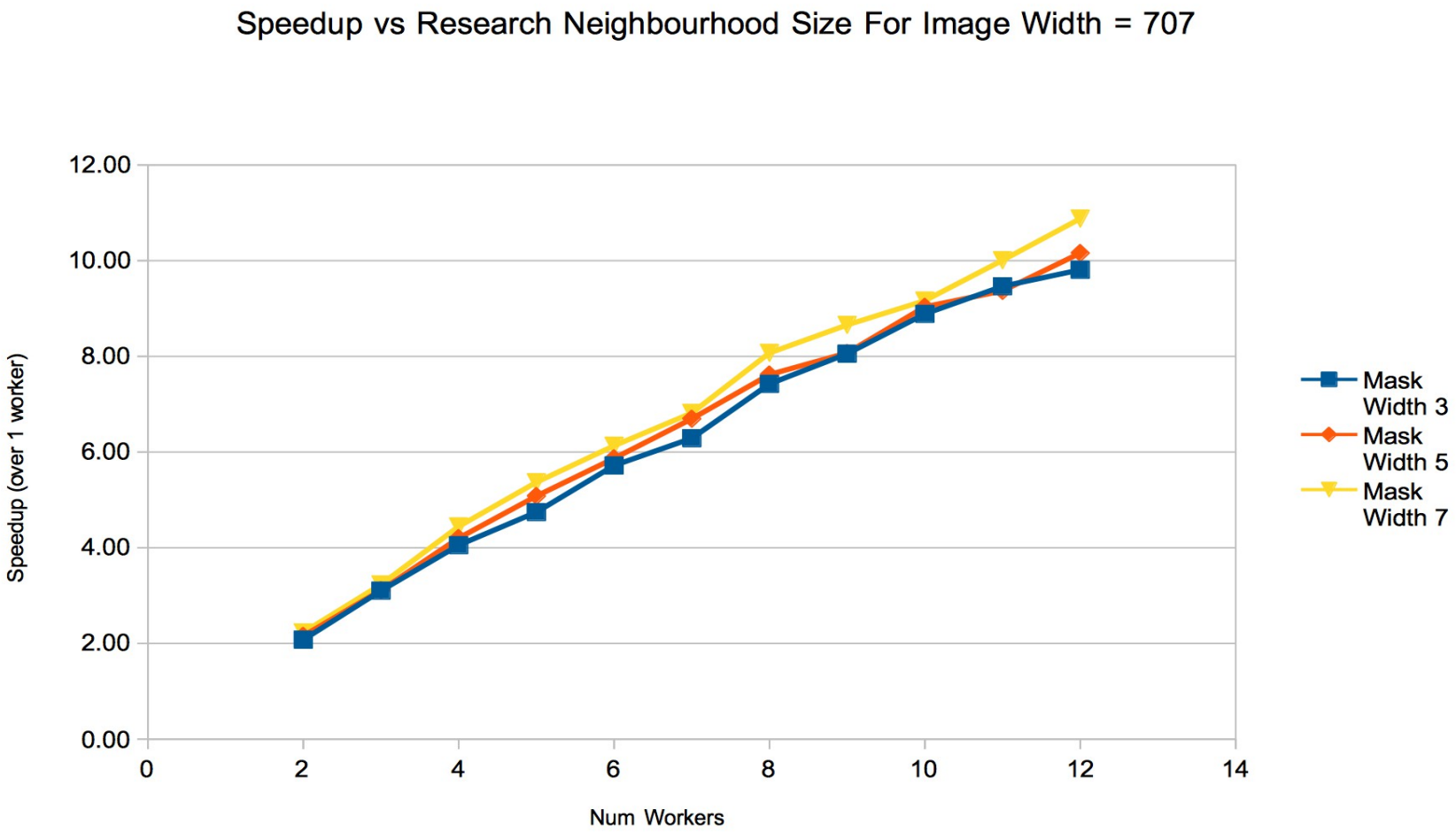
Speedup as image size increases and neighbourhood size fixed



Similar behaviour for other mask sizes (25, 49, 81 pixel neighbourhoods)

Progress since last time.....preliminary results

Speedup as mask size increases and image size is fixed



Similar behaviour for other image sizes (500, 866 wide images)

Pending work...

- Enhance the VMF kernel with GPU enhanced functions (reshape, repmat, permute), integrate into parfor loop – Still waiting for HPC group to finish configuring on Zeno
- Collapse the nested for loops that index over the input image pixels into a single loop (MATLAB linear slicing)
  - Parfor has restrictions on indexing with parfor loop indices. I currently store individual pixel data in a cell array as fancy indexing is incompatible with parfor (ex. B = <a 3d array>; C = B(i-4:i+4,:,:)
  - I expect better efficiency
- Parallelize the Adapted Centre Weighted Vector Median Filter, if I have time

From the review...

- Why haven’t very many non-linear filters related to the VMF been parallelized?