A SIMPLE INPAINTING METHOD AND ITS GPU IMPLEMENTATION

Hannes Fassold - JOANNEUM RESEARCH, DIGITAL
hannes.fassold@joanneum.at

ICVSS 2011
Sicily ~ 11-16 July
International Computer Vision Summer School

Abstract

A simple image inpainting method is proposed, and its efficient GPU implementation for NVIDIA GPUs is described. A speedup factor of 7 - 11 is observed for the GPU implementation, compared with an optimized CPU implementation.

Algorithm

- Determine set of hole border pixels
- For each hole border pixel, propagate its intensity into the hole along a fixed set of directions (e.g. 16)
  - Uses accumulator image A and weight image W
  - All pixels P visited during line tracing (Bresenham) are updated via

$$A(p) = A(p) + g_b/d_{curr}$$

$$W(p) = W(p) + 1/d_{curr}$$

    where g_b is the border pixel intensity and d_{curr} is the distance to the respective border pixel
  - Inpainted image $I = A/W$
- Distance-adaptive blurring of inpainted regions
  - Reduces some star-like artefacts which may occur
  - Calculate distance map and blur hole pixels adaptively (less blurring for hole pixels near to border)

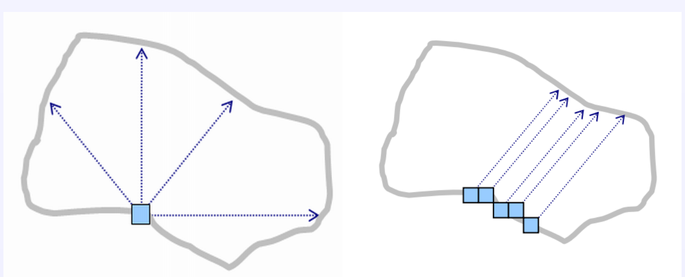


Illustration of border pixel propagation

Goal

- Simple and fast inpainting algorithm
- Suitable for thin, crack-like holes (e.g. appearing in warped images in frame interpolation)
- Maps well onto massively parallel devices like GPUs

GPU implementation

Determine hole border pixel set

- 3 x 3 Dilate and Subtraction from original image delivers mask where only border pixels are set
- Use compaction function from NVIDIA CUDPP library to retrieve list of border pixel positions

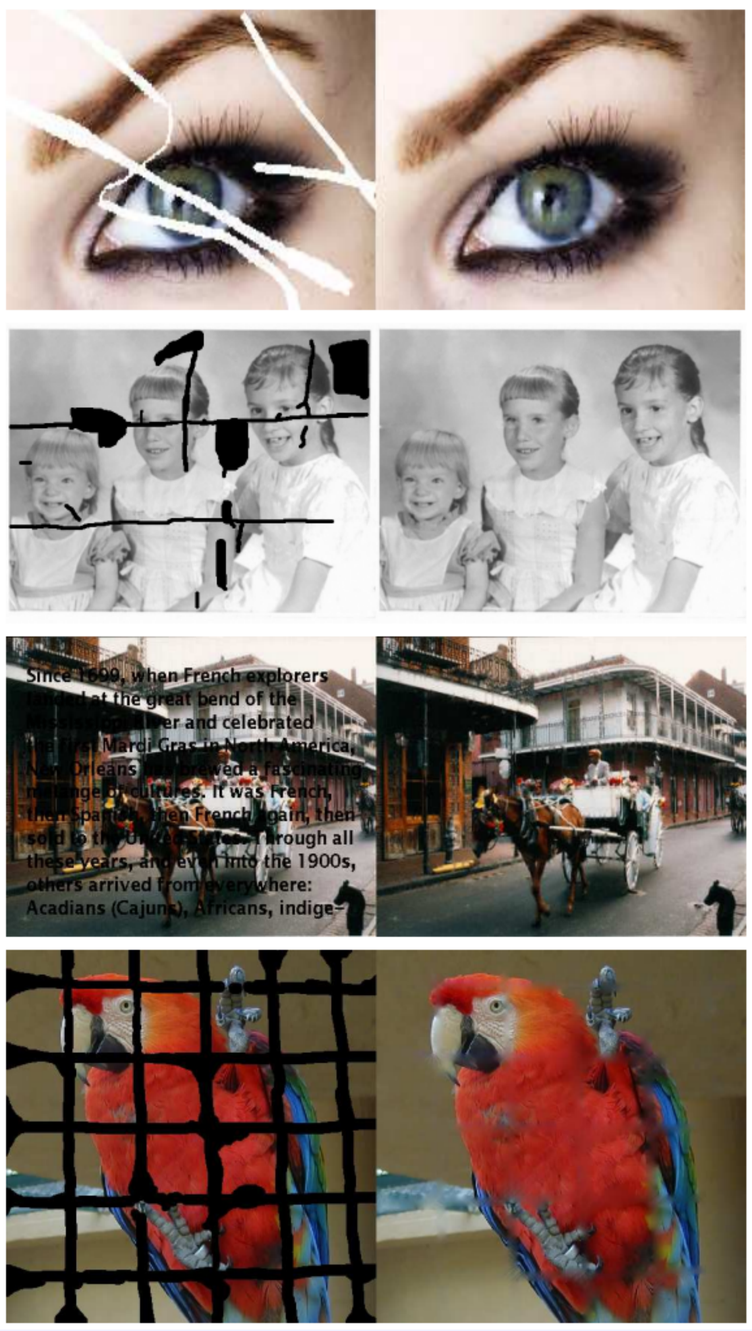
Border pixel propagation

- One thread = one border pixel
- One kernel call per direction (all threads trace in the same direction)
- Atomic operations *not* used
  - Generally slow on pre-Fermi GPUs
  - Not available for *float* type for pre-Fermi GPU
- Performance penalty due to warp divergence
  - Due to different paths the threads are tracing

Distance-adaptive blurring

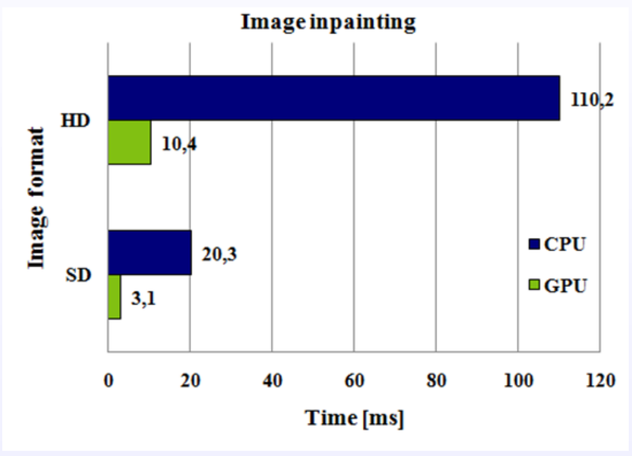
- Distance map (fast methods are inherently serial) replaced by approximation
- Each thread read reads neighbors for largest occuring blur kernel (to reduce warp divergence)

Inpainting Results



Runtime Results

- Intel Xeon 3.0 Ghz Quadcore, NVIDIA GTX 285
- 3-channel, 8bit image
- 1.4 % of pixels to inpaint



Acknowledgements

The author would like to thank Jakub Rosner and several collegues at JOANNEUM RESEARCH - DIGITAL.