GPU Accelerated Time-of-Flight Super-Resolution for Image-Guided Surgery

Jens Wetzl[1], Oliver Taubmann[1], Sven Haase[1],
Thomas Köhler[1,2], Martin Kraus[1,2], Joachim Hornegger[1,2]

BVM Heidelberg, 04.03.2013

[1] Pattern Recognition Lab, FAU Erlangen-Nuremberg
[2] Erlangen Graduate School in Advanced Optical Technologies (SAOT)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Outline

Introduction

Image-Guided Surgery

Range Imaging

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Image-Guided Surgery

- Scan surface in operating room, register with preoperative CT/MR data
- Show tracked instruments aligned with CT/MR data
- Example application: Image-guided abdominal surgery

Image source: transplantsurg.wustl.edu

Range Imaging

- Surface information can be acquired with a Time-of-Flight camera
- Pros:
  - (+) Dense range data
  - (+) Real-time frame rates
- Cons:
  - (−) Low spatial resolution
  - (−) Poor signal-to-noise ratio

Range Imaging

- Surface information can be acquired with a Time-of-Flight camera
- Pros:
  - (+) Dense range data
  - (+) Real-time frame rates
- Cons:
  - (−) Low spatial resolution
  - (−) Poor signal-to-noise ratio

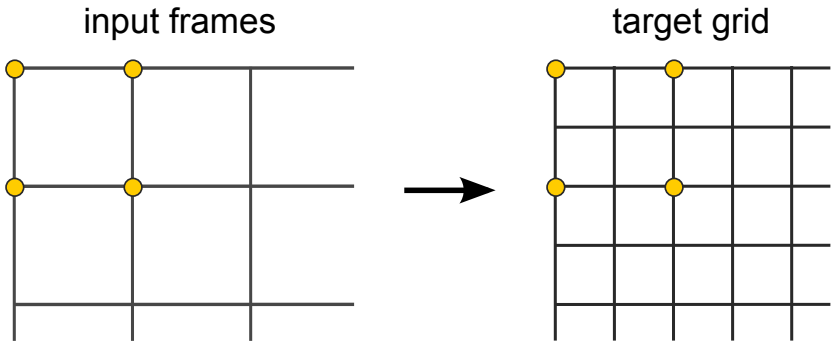
Methods

- Super-Resolution
- Nonlinear Optimization

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Super-Resolution

- Multi-frame super-resolution
  - Given: **Sequence** of preregistered **low resolution images**
  - Goal: Combine information in a **single high resolution image**

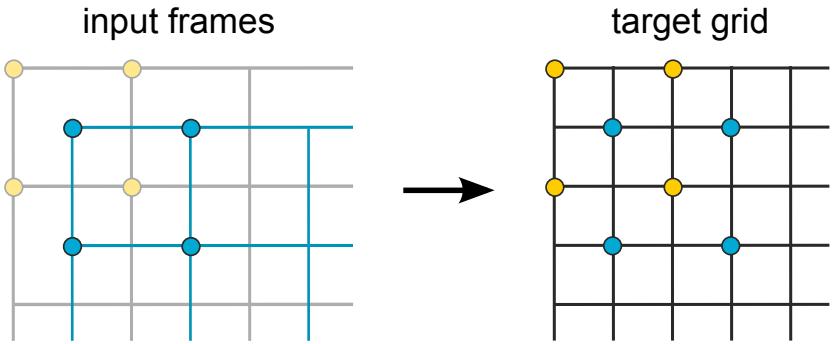


Idealized Sub-Pixel Motion Gain

- Utilizing **sub-pixel motion** improves effective resolution

Super-Resolution

- Multi-frame super-resolution
  - Given: **Sequence** of preregistered **low resolution images**
  - Goal: Combine information in a **single high resolution image**



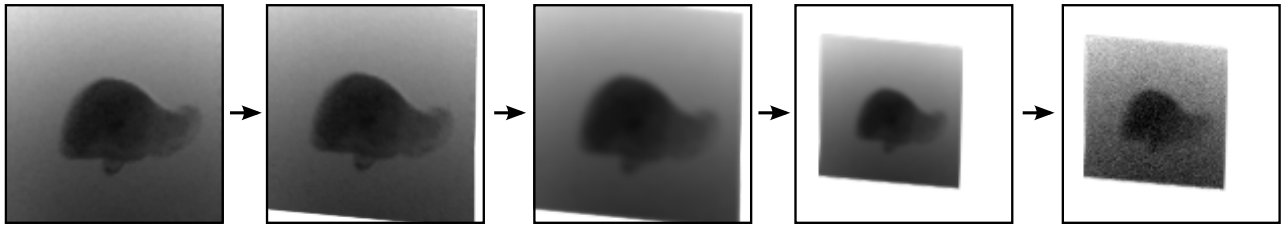
Idealized Sub-Pixel Motion Gain

- Utilizing **sub-pixel motion** improves effective resolution

Method

See: *Pickup LC. Machine learning in multi-frame image super-resolution. PhD Thesis, University of Oxford; 2008.*

- Based on a **forward model**
  - Low-resolution (LR) images generated from ideal high-resolution (HR) image
  - Affine motion, a Gaussian point spread function (PSF) and decimation



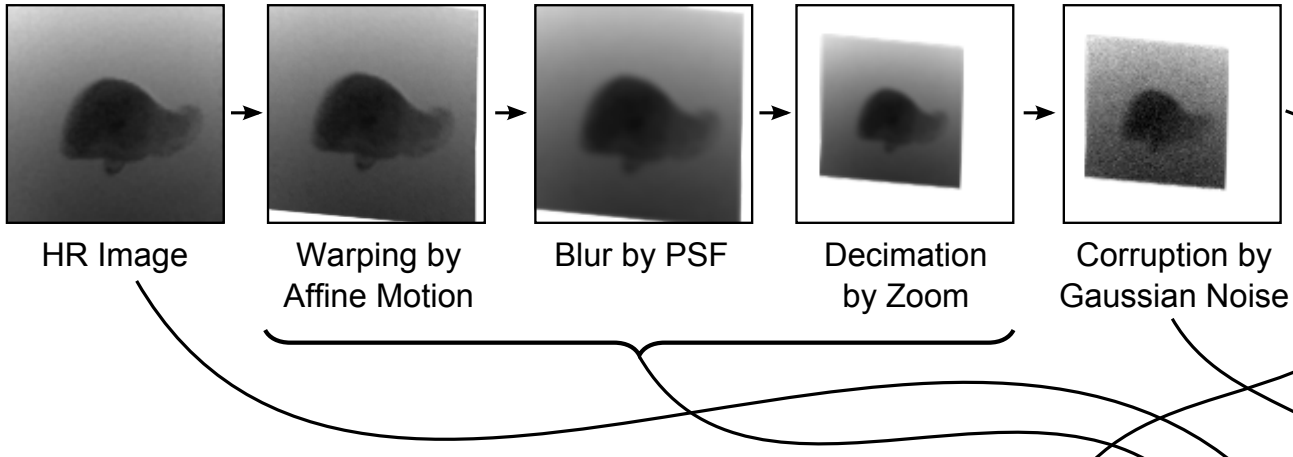
HR Image | Warping by Affine Motion | Blur by PSF | Decimation by Zoom | Corruption by Gaussian Noise

$$W_{mn} = \exp\left(-\frac{(\boldsymbol{v}_n - \boldsymbol{H}\boldsymbol{u}_m)^T \nabla\boldsymbol{H}(\nabla\boldsymbol{H})^T (\boldsymbol{v}_n - \boldsymbol{H}\boldsymbol{u}_m)}{2\sigma^2}\right),$$

$$\boldsymbol{y}^{(k)} = \boldsymbol{W}^{(k)}\boldsymbol{x} + \varepsilon^{(k)}$$

Method

- Based on a **forward model**
  - Low-resolution (LR) images generated from ideal high-resolution (HR) image
  - Affine motion, a Gaussian point spread function (PSF) and decimation

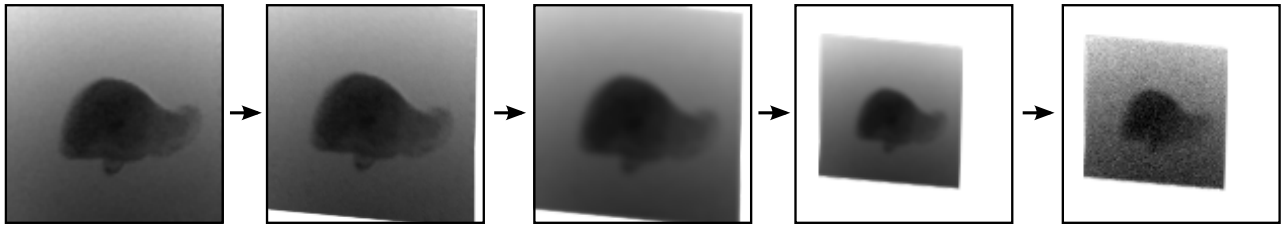


$$W_{mn} = \exp\left(-\frac{(\boldsymbol{v}_n - \boldsymbol{H}\boldsymbol{u}_m)^T \nabla \boldsymbol{H} (\nabla \boldsymbol{H})^T (\boldsymbol{v}_n - \boldsymbol{H}\boldsymbol{u}_m)}{2\sigma^2}\right),$$

$$\boldsymbol{y}^{(k)} = \boldsymbol{W}^{(k)}\boldsymbol{x} + \boldsymbol{\varepsilon}^{(k)}$$

Method

- Based on a **forward model**
  - Low-resolution (LR) images generated from ideal high-resolution (HR) image
  - Affine motion, a Gaussian point spread function (PSF) and decimation

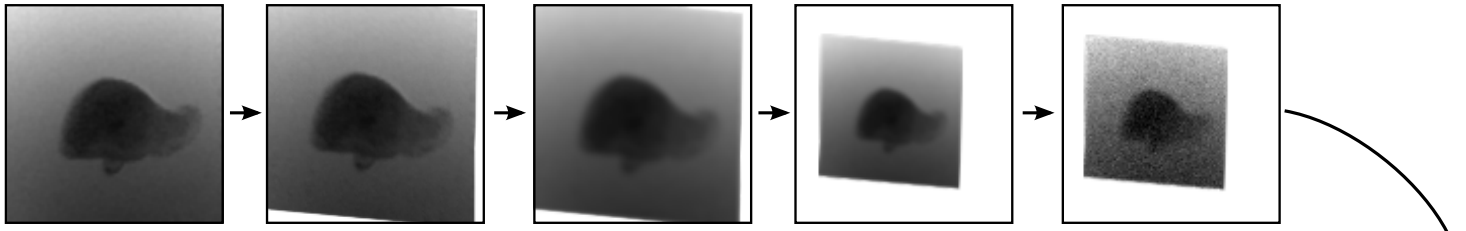


HR Image | Warping by Affine Motion | Blur by PSF | Decimation by Zoom | Corruption by Gaussian Noise

$$W_{mn} = \exp\left(-\frac{(\boldsymbol{v}_n - \boldsymbol{H}\boldsymbol{u}_m)^T \nabla\boldsymbol{H}(\nabla\boldsymbol{H})^T (\boldsymbol{v}_n - \boldsymbol{H}\boldsymbol{u}_m)}{2\sigma^2}\right), \qquad \boldsymbol{y}^{(k)} = \boldsymbol{W}^{(k)}\boldsymbol{x} + \varepsilon^{(k)}$$

Method

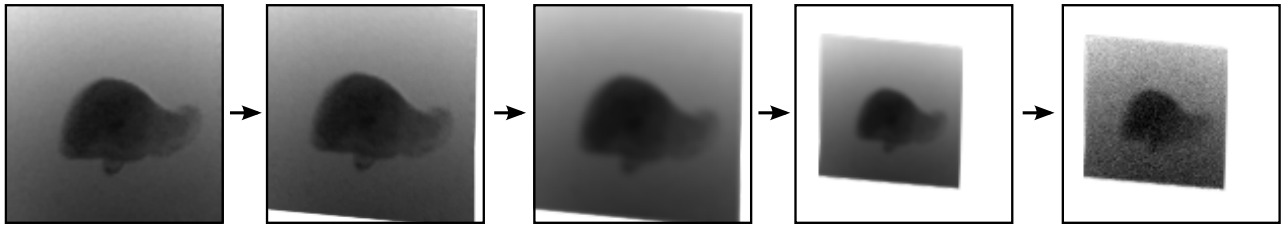
- Based on a **forward model**
  - Low-resolution (LR) images generated from ideal high-resolution (HR) image
  - Affine motion, a Gaussian point spread function (PSF) and decimation



$$W_{mn} = \exp\left(-\frac{(\boldsymbol{v}_n - \boldsymbol{H}\boldsymbol{u}_m)^T \nabla \boldsymbol{H} (\nabla \boldsymbol{H})^T (\boldsymbol{v}_n - \boldsymbol{H}\boldsymbol{u}_m)}{2\sigma^2}\right), \qquad \boldsymbol{y}^{(k)} = \boldsymbol{W}^{(k)}\boldsymbol{x} + \varepsilon^{(k)}$$

Method

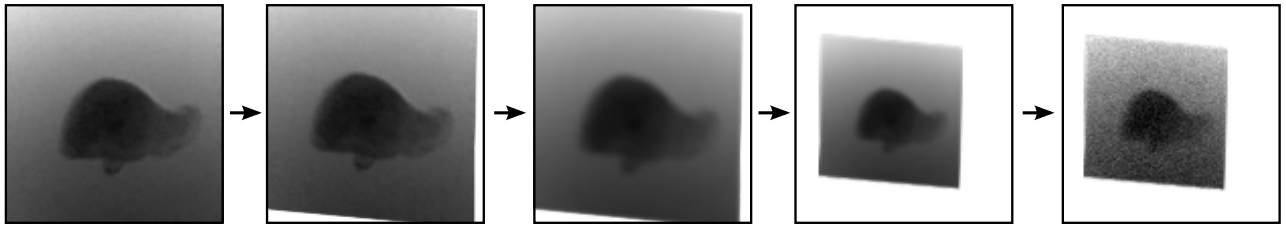
- Based on a **forward model**
  - Low-resolution (LR) images generated from ideal high-resolution (HR) image
  - Affine motion, a Gaussian point spread function (PSF) and decimation



$$W_{mn} = \exp\left(-\frac{(\boldsymbol{v}_n - \boldsymbol{H}\boldsymbol{u}_m)^T \nabla\boldsymbol{H}(\nabla\boldsymbol{H})^T (\boldsymbol{v}_n - \boldsymbol{H}\boldsymbol{u}_m)}{2\sigma^2}\right), \qquad \boldsymbol{y}^{(k)} = \boldsymbol{W}^{(k)}\boldsymbol{x} + \varepsilon^{(k)}$$

Method

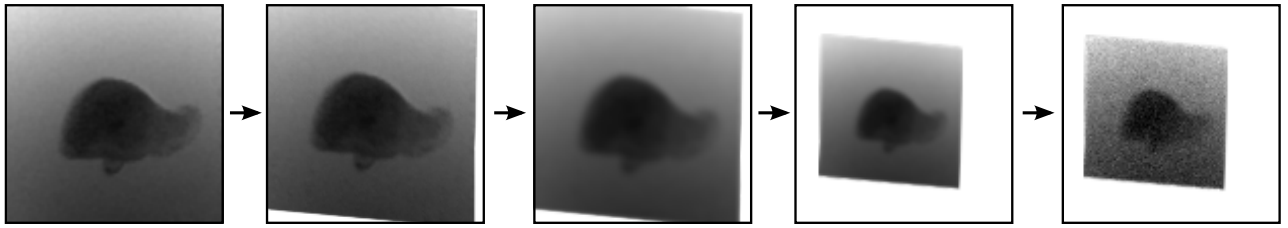
- Based on a **forward model**
  - Low-resolution (LR) images generated from ideal high-resolution (HR) image
  - Affine motion, a Gaussian point spread function (PSF) and decimation



$$W_{mn} = \exp\left(-\frac{(\boldsymbol{v}_n - \boldsymbol{H}\boldsymbol{u}_m)^T \nabla\boldsymbol{H}(\nabla\boldsymbol{H})^T(\boldsymbol{v}_n - \boldsymbol{H}\boldsymbol{u}_m)}{2\sigma_r^2}\right), \qquad \boldsymbol{y}^{(k)} = \boldsymbol{W}^{(k)}\boldsymbol{x} + \varepsilon^{(k)}$$

Method

- Based on a **forward model**
  - Low-resolution (LR) images generated from ideal high-resolution (HR) image
  - Affine motion, a Gaussian point spread function (PSF) and decimation



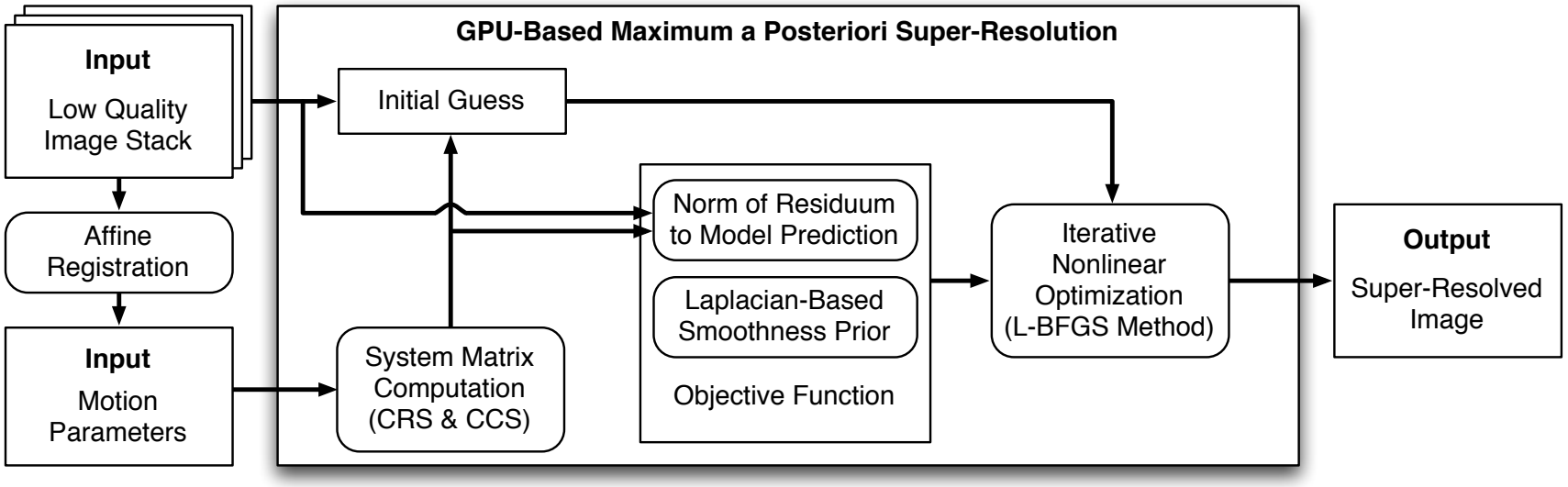
- **Inversion** of that model combined with a **smoothness prior**
  - Formulated as a nonlinear optimization problem
  - Yields a **maximum a posteriori** (MAP) estimate

Objective Function

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left(\| \boldsymbol{W}\boldsymbol{x} - \boldsymbol{y} \|_2^2 + \lambda \cdot \| \mathrm{h}_\delta(\boldsymbol{D}\boldsymbol{x}) \|_1 \right) \tag{1}$$

$\boldsymbol{W}$	System matrix, models image generation
$\boldsymbol{x}$	HR image, linearized
$\boldsymbol{y}$	LR images, linearized and stacked
$\boldsymbol{W}\boldsymbol{x} - \boldsymbol{y}$	Residuum to model prediction
λ	Prior strength, adjusts smoothness
$\mathrm{h}_\delta(\cdot)$	Pseudo-Huber loss function, applied element-wise
$\boldsymbol{D}\boldsymbol{x}$	Discrete Laplacian of the ideal image

System Overview



Computation of system matrix, evaluation of objective function and nonlinear optimization all done on the GPU using NVIDIA CUDA.

Sparse System Matrix – Challenges

- $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ is prohibitively large

$$m = \#_{\text{LR pixels}} \cdot \#_{\text{LR images}} \tag{2}$$

$$n = \#_{\text{HR pixels}} \tag{3}$$

$\rightarrow$ Typical size e.g. $(200^2 \cdot 10) \times 400^2 = 64 \cdot 10^9$ elements
$\rightarrow$ Set point spread function to zero beyond $3 \cdot \sigma$
$\rightarrow$ $\boldsymbol{W}$ becomes sparse and can be stored in (GPU) memory

- Computing MULTIPLY **and** TRANSPOSE-MULTIPLY efficiently

$$\mathrm{r}(\boldsymbol{x}) = \|\boldsymbol{W}\boldsymbol{x} - \boldsymbol{y}\|^2 \tag{4}$$

$$\nabla \mathrm{r}(\boldsymbol{x}) = 2\boldsymbol{W}^T(\boldsymbol{W}\boldsymbol{x} - \boldsymbol{y}) \tag{5}$$

$\rightarrow$ Store both CRS* and CCS* representations of $\boldsymbol{W}$

* Compressed {Row, Column} Storage

Sparse System Matrix – Challenges

- $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ is prohibitively large

$$m = \#_{\text{LR pixels}} \cdot \#_{\text{LR images}} \tag{2}$$

$$n = \#_{\text{HR pixels}} \tag{3}$$

  → Typical size e.g. $(200^2 \cdot 10) \times 400^2 = 64 \cdot 10^9$ elements
  → Set point spread function to zero beyond $3 \cdot \sigma$
  → $\boldsymbol{W}$ becomes sparse and can be stored in (GPU) memory

- Computing MULTIPLY **and** TRANSPOSE-MULTIPLY efficiently

$$\mathrm{r}(\boldsymbol{x}) = \|\boldsymbol{W}\boldsymbol{x} - \boldsymbol{y}\|^2 \tag{4}$$

$$\nabla \mathrm{r}(\boldsymbol{x}) = 2\boldsymbol{W}^T(\boldsymbol{W}\boldsymbol{x} - \boldsymbol{y}) \tag{5}$$

  → Store both CRS* and CCS* representations of $\boldsymbol{W}$

  * Compressed {Row, Column} Storage

Nonlinear Optimization

- Objective function gradient **nonlinear** in HR image pixels
  $\rightarrow$ Iterative descent method for optimization
- We chose L-BFGS (limited-memory Broyden-Fletcher-Goldfarb-Shanno)
  - Quasi-Newton variant, high convergence rate
  - Stores **low-rank** approximation of inverse Hessian
    $\rightarrow$ Low memory footprint, well suited for GPU implementation
- Implementation of a **general-purpose library** for unconstrained nonlinear optimization on the GPU

Nonlinear Optimization

- Objective function gradient **nonlinear** in HR image pixels
  $\rightarrow$ Iterative descent method for optimization
- We chose L-BFGS (limited-memory Broyden-Fletcher-Goldfarb-Shanno)
  - Quasi-Newton variant, high convergence rate
  - Stores **low-rank** approximation of inverse Hessian
    $\rightarrow$ Low memory footprint, well suited for GPU implementation
- Implementation of a **general-purpose library** for unconstrained nonlinear optimization on the GPU

Experiments and Results

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Experiments

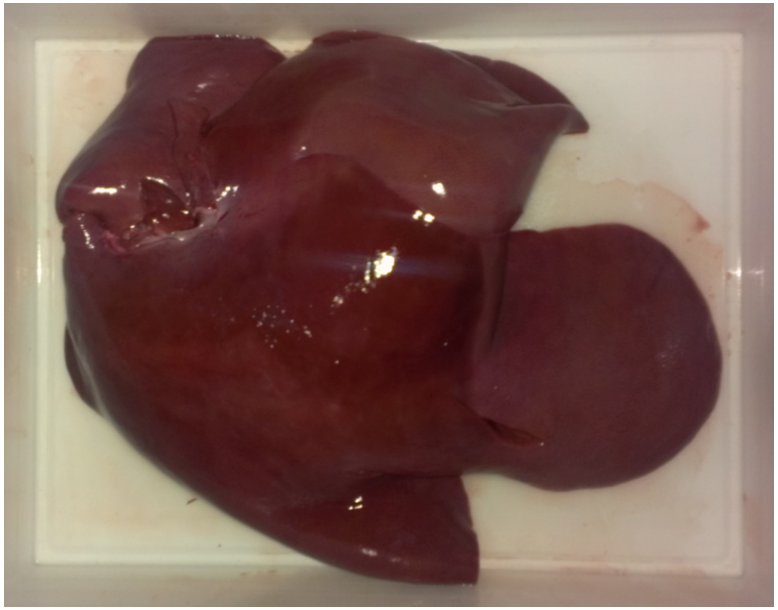
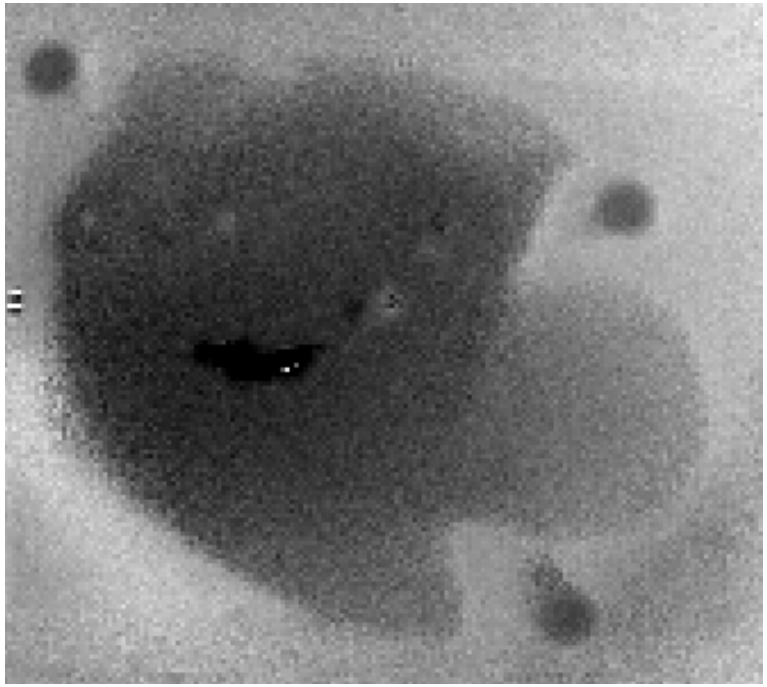
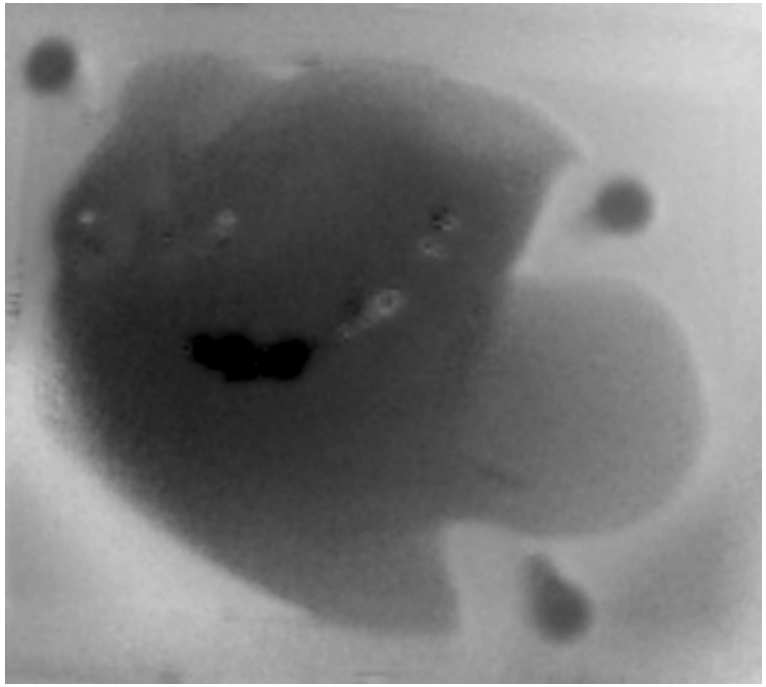


Photo of a porcine liver we measured with a PMD CamCube 3.0

Qualitative Results



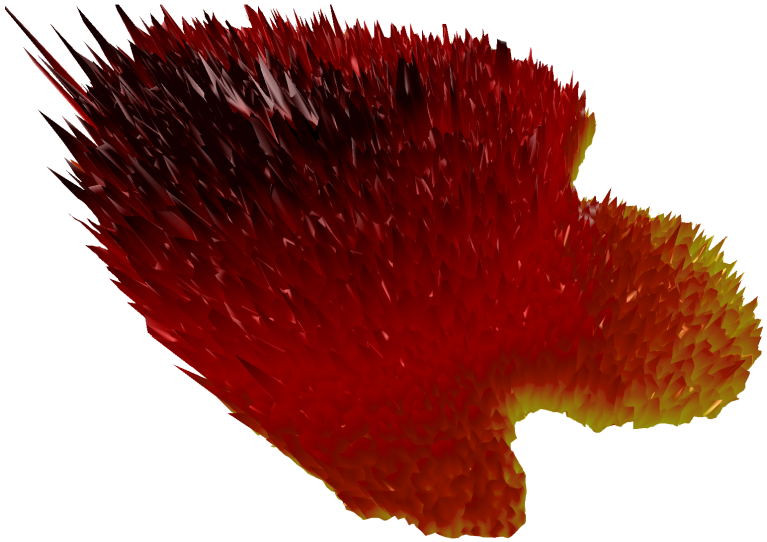
LR frame



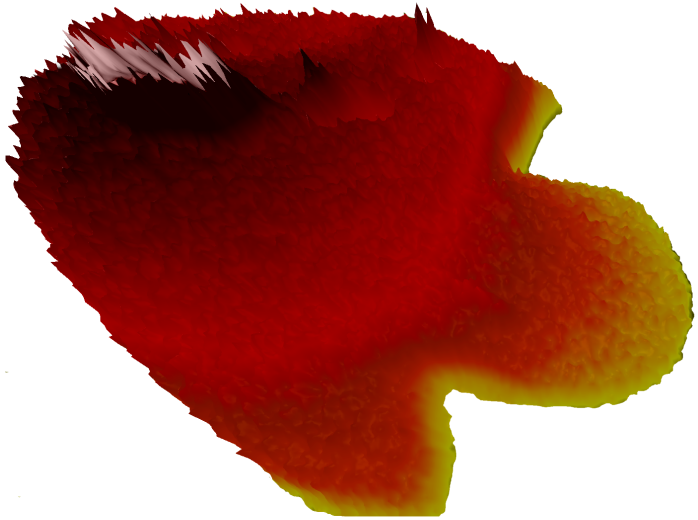
HR frame

(10 input frames, 2× upsampling factor)

Qualitative Results



Mesh from LR frame



Mesh from HR frame
(10 input frames, 2× upsampling factor)

Quantitative Results

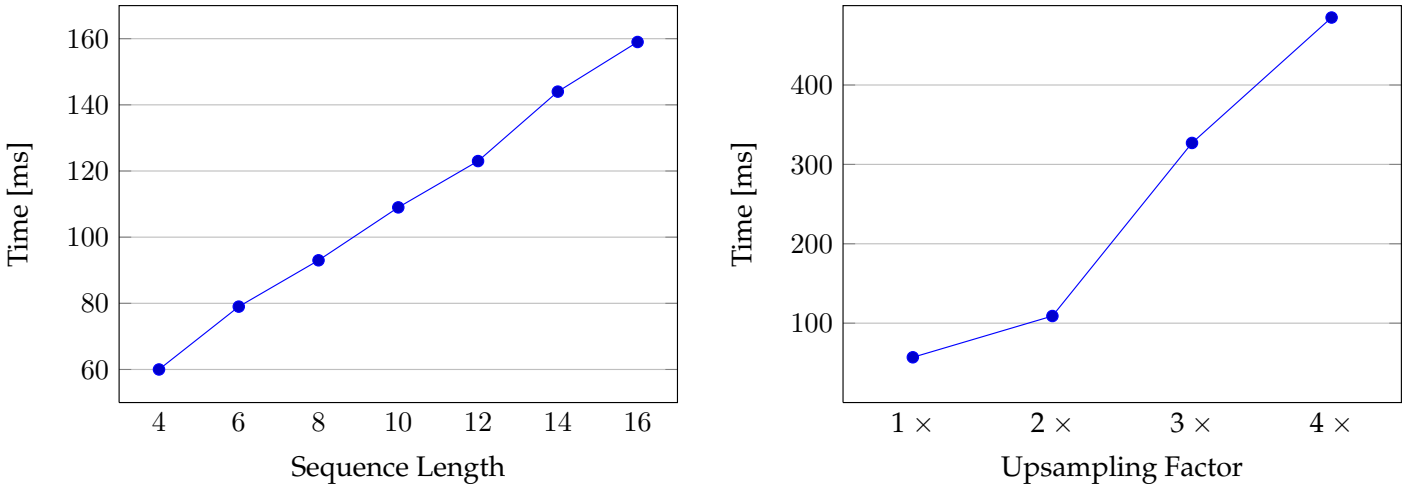
	Raw	SR
RMSE	8.11 mm	6.29 mm
Median	4.92 mm	3.16 mm
Std. dev.	±0.34 mm	±0.16 mm

Error statistics for registered meshes
(Parameters: 10 input frames, 2× upsampling factor)

- RMSE: Root-mean-square error
- Median: Median of absolute distances
- Std. dev.: Standard deviation of absolute distances

$\Rightarrow$ RMSE improved by more than 20%

Quantitative Results



(GPU: NVIDIA GTX 580, input frame size: 200 × 200)

Measured runtimes for varying

- sequence lengths (with 2× upsampling factor)
- upsampling factors (with 10 input frames)

Summary and Outlook

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Summary and Outlook

- Super-resolution framework (including optimizer) on the GPU
  - Comprehensive image generation model
  - MAP estimate ($\rightarrow$ smoothness prior)
  - **Interactive frame rates**
- What's left to do?
  - Integrate robust affine registration
  - Combine registration and super-resolution
- Our code is available under an open source license (CC-BY 3.0)

  $\Rightarrow$ Feel free to download it from our website and experiment!

  `www5.cs.fau.de/map-superresolution` (coming soon)

Summary and Outlook

- Super-resolution framework (including optimizer) on the GPU
  - Comprehensive image generation model
  - MAP estimate ($\rightarrow$ smoothness prior)
  - **Interactive frame rates**
- What's left to do?
  - Integrate robust affine registration
  - Combine registration and super-resolution
- Our code is available under an open source license (CC-BY 3.0)

  $\Rightarrow$ Feel free to download it from our website and experiment!

  `www5.cs.fau.de/map-superresolution` (coming soon)

Summary and Outlook

- Super-resolution framework (including optimizer) on the GPU
  - Comprehensive image generation model
  - MAP estimate ($\rightarrow$ smoothness prior)
  - **Interactive frame rates**
- What's left to do?
  - Integrate robust affine registration
  - Combine registration and super-resolution
- Our code is available under an open source license (CC-BY 3.0)

  $\Rightarrow$ Feel free to download it from our website and experiment!

  `www5.cs.fau.de/map-superresolution` (coming soon)

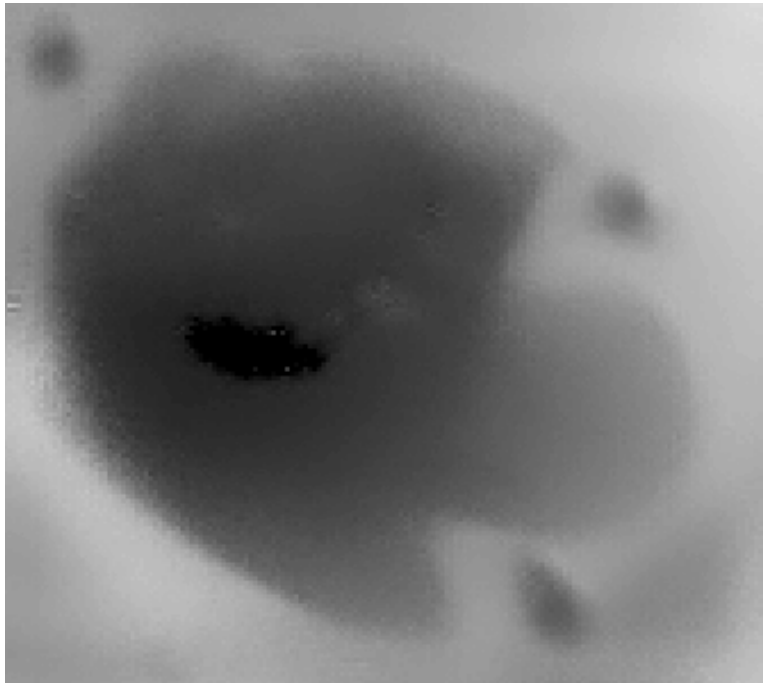
Acknowledgements

We gratefully acknowledge funding of the **Erlangen Graduate School in Advanced Optical Technologies** (SAOT) by the **German Science Foundation** (DFG) in the framework of the excellence initiative, as well as the support by the DFG under Grant No. HO 1791/7-1. This research was funded/supported by the **Graduate School of Information Science in Health** (GSISH) and the TUM Graduate School.

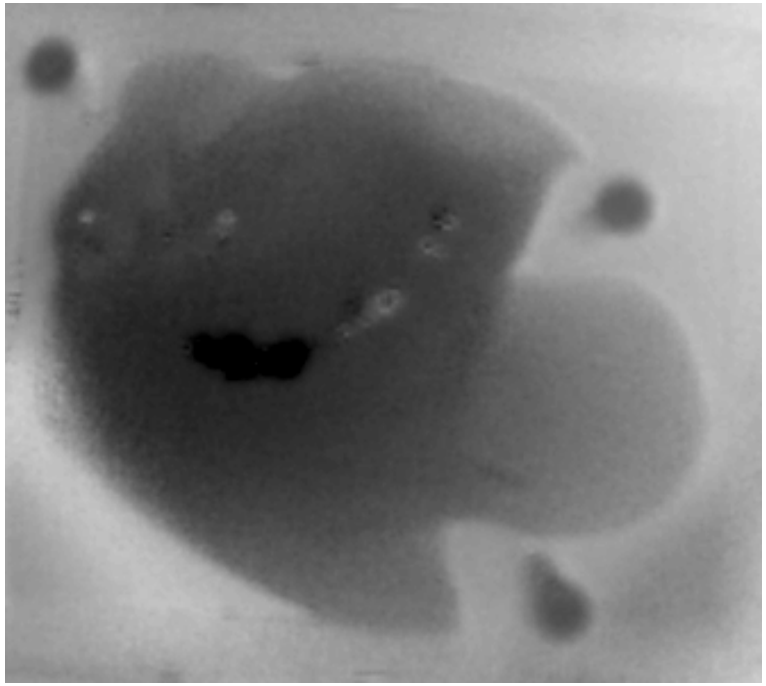
SAOT DFG Deutsche Forschungsgemeinschaft GSISH

Thanks for listening!

Qualitative Results (Addenda)



Preprocessed LR frame



HR frame

(10 raw frames, 2× upsampling factor)