Matching Deformable 3D Shapes

David Dao, Johannes Rausch, Michal Szymczak

Technische Universität München
Department of Informatics
Computer Vision Group

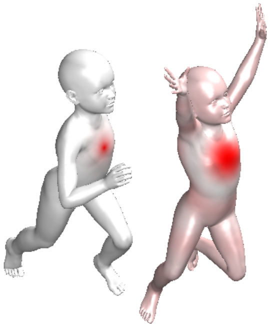
October 6, 2015

Outline

Outline

Introduction

- Point-wise Map Recovery and Refinement from Functional Correspondence by Rodola et al. [E. Rodola, 2015]



Outline

3D Shape Matching

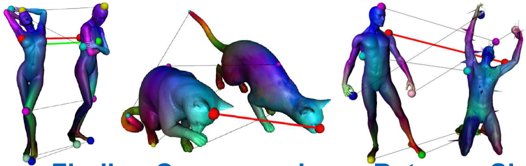


Figure : Finding Correspondence Between Shapes

0	1	0	0	0
0	0	0	1	0
1	0	0	0	0
0	0	0	0	1
0	0	1	0	0

Figure : Correspondence Matrix

3D Shape Matching

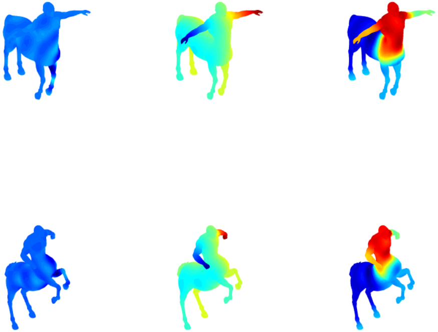


Figure : Functional Mapping

3D Shape Matching

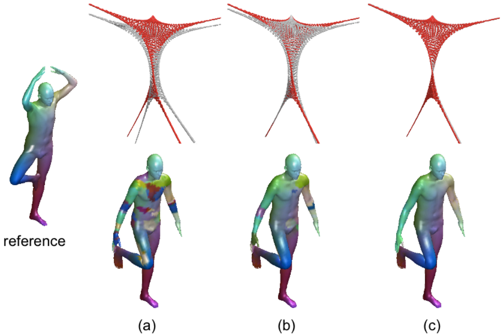


Figure : Point-To-Point Recovery

Outline

Coherent Point Drift (CPD) - Algorithm

Non-rigid point set registration algorithm:

- Initialization: $\mathbf{W} = 0, \sigma^2 = \frac{1}{DNM} \sum_{m,n=1}^{M,N} \|\mathbf{x}_n - \mathbf{y}_m\|^2$
- Initialize $w (0 \leq w \leq 1)$, $\beta > 0$, $\lambda > 0$,
- Construct $\mathbf{G}$: $g_{ij} = \exp^{-\frac{1}{2\beta^2}\|\mathbf{y}_i - \mathbf{y}_j\|^2}$,
- EM optimization, repeat until convergence:
  - E-step: Compute $\mathbf{P}$,

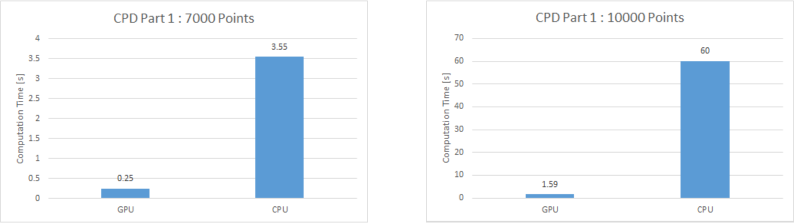
    $$p_{mn} = \frac{\exp^{-\frac{1}{2\sigma^2}\|\mathbf{x}_n - (\mathbf{y}_m + \mathbf{G}(m,\cdot)\mathbf{W})\|^2}}{\sum_{k=1}^{M} \exp^{-\frac{1}{2\sigma^2}\|\mathbf{x}_n - (\mathbf{y}_k + \mathbf{G}(k,\cdot)\mathbf{W})\|^2} + \frac{w}{1-w}\frac{(2\pi\sigma^2)^{D/2} M}{N}}$$
  - M-step:
    - Solve $(\mathbf{G} + \lambda\sigma^2 d(\mathbf{P1})^{-1})\mathbf{W} = d(\mathbf{P1})^{-1}\mathbf{PX} - \mathbf{Y}$
    - $N_{\mathbf{P}} = \mathbf{1}^T\mathbf{P1}$, $\mathbf{T} = \mathbf{Y} + \mathbf{GW}$,
    - $\sigma^2 = \frac{1}{N_{\mathbf{P}} D}(\mathrm{tr}(\mathbf{X}^T \mathrm{d}(\mathbf{P}^T\mathbf{1})\mathbf{X}) - 2\,\mathrm{tr}((\mathbf{PX})^T\mathbf{T}) + \mathrm{tr}(\mathbf{T}^T \mathrm{d}(\mathbf{P1})\mathbf{T}))$,
- The aligned point set is $\mathbf{T} = \mathcal{T}(\mathbf{Y}, \mathbf{W}) = \mathbf{Y} + \mathbf{GW}$,
- The probability of correspondence is given by $\mathbf{P}$.

Figure : CPD Algorithm: P is $M \times N$ [Myronenko and Song, 2010]

Coherent Point Drift (CPD) - Implementation

- Algorithm
  - CPU version uses several loops
  - Matrix P ($M \times N$) is never actually calculated
- GPU Implementation
  - Utilize optimized libraries (CuBLAS)
  - Vectorize operations
  - Use matrix slicing to circumvent memory limitations

Coherent Point Drift (CPD) -Evaluation



(a) (b)

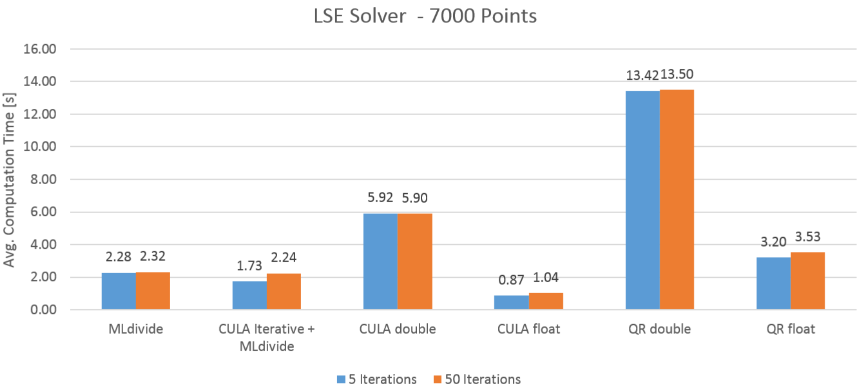
Figure : Average Runtime for 7000 and 10000 Points

Outline

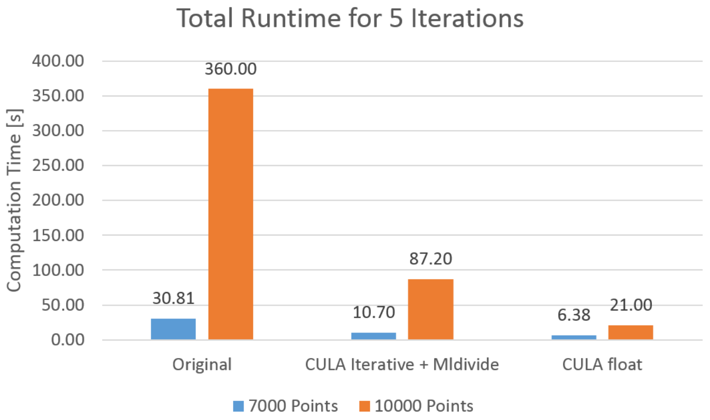
Linear System Solver

- Large, Dense Linear System of Equations (LSE)
- Memory Limitations
- Libraries and approaches (CuSolver, CULA, MAGMA, CuBLAS, Matlab)

Linear System Solver - Evaluation



Total Runtime



Outline

Conclusion and Future Work

- Consider single precision for whole computation
  - Current implementation relies on double precision in CPD
- Utilize the new GPU Cluster at the Vision Chair
- Develop approach that does not rely on huge, dense LSE

Bibliography I

[E. Rodola, 2015] E. Rodola, M. Moeller, D. C. (2015).
Point-wise map recovery and refinement from functional correspondence.

[Myronenko and Song, 2010] Myronenko, A. and Song, X. (2010).
Point set registration: Coherent point drift.
Pattern Analysis and Machine Intelligence, IEEE Transactions on, 32(12):2262--2275.