## Merge2-3D: Combining Multiple Normal Maps with 3D Surfaces

Sema Berkiten	Xinyi Fan	Szymon Rusinkiewicz
Princeton University	Princeton University	Princeton University
berkiten@princeton.edu	xinyi@princeton.edu	smr@princeton.edu

## **Data Acquisition**



Figure 1. Left: the multi-light capture setup for photometric stereo. Right: NextEngine laser scanner used for 3D acquisition.

### **Initial alignment**

nSIFT: first two columns show detected nSIFT keypoints with their orientations of the real normal map and the best-matched rendered normal map; and the last columns show the matches between the real and the best-matching rendered normal map.



Real normal map

Rendered normal map

Matches



### **Fine Alignment**

Figure 2 shows the value of the energy function over time, for two different models. Restarts are marked with vertical dashed lines. At the beginning and end, and at some restarts, we visualize the remaining misalignment: the normal map is rendered in red, while the mesh is in blue. We observe that the energy drops quickly during the first several iterations, eventually converging after 2–3 restarts.



Figure 2. Optimized energy as a function of Nelder-Mead iterations for a scarab model (scanned with a laser scanner) and a penguin model (acquired with MS Kinect). False-colored visualizations of the normal maps (in red) blended with the 3D models (in blue) are shown below each graph for the initial alignment, after each restart, and the final alignment from left to right. (Restart positions are marked with vertical dashed lines.)

# Blending



(b) Scarab

Figure 3. Comparison of different blending methods. *Left:* averaging all sample normals. **Center:** choosing the normal which is captured from the most direct view point. **Right:** our proposed blending method. Normals are false-colored by embedding in the RGB color space.

#### **Surface Enhancement**

We solve the following nonlinear energy function via Gradient Descent:

$$\mathcal{E} = \lambda \mathcal{E}_p + (1 - \lambda) \mathcal{E}_n,\tag{1}$$

where  $\mathcal{E}_p$  is the position term and  $\mathcal{E}_n$  is the normal term. These are formulated as follows:

$$\mathcal{E}_p = \sum_{v=1}^{\#vert} \|\boldsymbol{p}_v - \boldsymbol{p}_v^{mesh}\|^2$$
(2)

$$\mathcal{E}_n = \sum_{v=1}^{\#vert} - \|\boldsymbol{n}_v \cdot \boldsymbol{n}_v^{measured}\|^2,$$
(3)

Algorithm 1 Gradient Descent for Surface Enhancement

**Input:** Initial vertex positions  $P_v$  of 3D model, per vertex blended normals from 2D normal maps  $N_v$ .

 $\begin{array}{l} \mbox{Initialize: } P_0 = P_v, \\ Jacobian_0 = 0, \\ Conjugate_0 = 0, \\ n = 0. \\ \mbox{repeat} \\ Jacobian_{n+1} = \mbox{Jacobian} \mbox{Calculation}(Jacobian_n, P_n, P_{n+1}) \\ \beta = \|Jacobian_{n+1}\|/\|Jacobian_n\| \\ Conjugate_{n+1} = Jacobian_n + \beta Conjugate_n \\ \gamma = \mbox{Line search}(P_n, Conjugate_{n+1}) \\ P_{n+1} = P_n + \gamma Conjugate_{n+1} \\ n = n + 1 \\ \mbox{until } \sum \|P_n - P_{n+1}\| < \tau \end{array}$ 

#### Algorithm 2 Jacobian Calculation

return  $P_v = P_{n+1}$ 

 $\begin{array}{l} \textbf{Input:} & (Jacobian_n, P_n, P_{n+1} \\ Jacobian_{vertex} = 2\lambda(P_{n+1} - P_n) \\ \textbf{Area-weighted vertex normal can be written as : } n_v = normalize(\sum_{i=1}^{\#neighbors} P_n[neighbor_i] \times P_n[neighbor_{i+1}]) \\ n_{measured} : \textbf{projected/blended normals from normal maps} \\ \textbf{for } n_i : \textbf{each neighbor of vertex } v \textbf{ do} \\ \textbf{given } n_i \textbf{ current neighbor of } v, n_{prev} \textbf{ and } n_{next} \textbf{ are the previous and the next neighboring vertices} \\ Jacobian_{edge_x} + = n_{measured}[n_i]_y(P_{n+1}[n_{prev}]_z - P_{n+1}[n_{next}]_z) + n_{measured}[n_i]_z(P_{n+1}[n_{next}]_y - P_{n+1}[n_{prev}]_y) \\ Jacobian_{edge_y} + = n_{measured}[n_i]_x(P_{n+1}[n_{next}]_z - P_{n+1}[n_{prev}]_z) + n_{measured}[n_i]_z(P_{n+1}[n_{prev}]_x - P_{n+1}[n_{next}]_x) \\ Jacobian_{edge_z} + = n_{measured}[n_i]_x(P_{n+1}[n_{prev}]_y - P_{n+1}[n_{next}]_y) + n_{measured}[n_i]_y(P_{n+1}[n_{next}]_x - P_{n+1}[n_{prev}]_x) \\ \textbf{end for} \\ Jacobian_{n+1} = Jacobian_{vertex} + Jacobian_{edge} \\ \textbf{return } Jacobian_{n+1} \end{array}$ 

## **Time Analysis**

The algorithm proposed by Nehab et al., Poisson reconstruction, and our algorithm converge in 12, 6, and 3 iterations in average, respectively. Note that the initial alignment code is not optimized and does brute-force search to find the best matching, and we run 3 iterations for each normal map: 72 rendered images compared to each acquired normal map in total.

Synthetic Dataset		
Dataset	Armadillo ( 0.2M vertices)	
# Normal Maps	6	
Initial Alignment (min/normal map)	$\sim 10$ mins	
Alignment Refinement (sec)	14.2 / 12.2 / 13.1 / 12.8 / 20.33 / 13.3	
Blending (sec)	2.2 (DC fixing)/ 1.9	
Nehab et al. (sec/iteration)	0.7	
Poisson Reconstruction (sec/iteration)	79	
Ours (sec/iteration)	9.1	
Acquired Datasets		

Dataset	00496( 1M vertices)	512( 0.5M vertices)
# Normal Maps	4	2
Initial Alignment (min/normal map)	$\sim 27$ mins	$\sim 28  \mathrm{mins}$
Alignment Refinement (sec)	28.1 / 13.6 / 10.8 / 13.3	16.4/17.5
Blending (sec)	2.7 (DC fixing)/14.7	17.8 (DC fixing)/8.3
Nehab et al. (sec/iteration)	11.1	3.9
Poisson Reconstruction (sec/iteration)	400	285
Ours (sec/iteration)	15.3	9.7
_	Denseria ( 0.4M section )	Soldier (0.5M vertices)
Dataset	Penguin( 0.4M vertices)	Soldier ( 0.51vi vertices)
Dataset # Normal Maps	2	2
Dataset # Normal Maps Initial Alignment (min/normal map)	Penguin( 0.4M vertices) $2$ ~18 mins	$\frac{2}{\sim 19 \text{ mins}}$
Dataset # Normal Maps Initial Alignment (min/normal map) Alignment Refinement (sec)	$\frac{2}{\sim 18 \text{ mins}}$ $49.7 / 63.3$	2 ~19 mins 100.8 / 79.4
Dataset # Normal Maps Initial Alignment (min/normal map) Alignment Refinement (sec) Blending(sec)	2 ~18 mins 49.7 / 63.3 25 (DC fixing)/ 5.8	2 ~19 mins 100.8 / 79.4 6.4 (DC fixing)/ 7.4
Dataset # Normal Maps Initial Alignment (min/normal map) Alignment Refinement (sec) Blending(sec) Nehab et al. (sec/iteration)	2 ~18 mins 49.7 / 63.3 25 (DC fixing)/ 5.8 2.3	2 ~19 mins 100.8 / 79.4 6.4 (DC fixing)/ 7.4 2.9
Dataset # Normal Maps Initial Alignment (min/normal map) Alignment Refinement (sec) Blending(sec) Nehab et al. (sec/iteration) Poisson Reconstruction (sec/iteration)	2 ~18 mins 49.7 / 63.3 25 (DC fixing)/ 5.8 2.3 99.5	2 ~19 mins 100.8 / 79.4 6.4 (DC fixing)/ 7.4 2.9 174

## Results

Here we show one of the input normal maps, the coarse scanned 3D model, and our optimized result for each of 5 datasets. Notice the significant additional detail, present in the normal maps, that is introduced into the 3D models using our method. This is a high-resolution PDF — please zoom in to see the full detail.



Normal map

Coarse 3D Model





Our Result





Normal map

Coarse 3D Model





Our Result





Coarse 3D Model



Our Result





Our Result





Our Result





Our Result



Normal map

Coarse 3D Model



Our Result



Coarse 3D Model



