Technical University of Munich

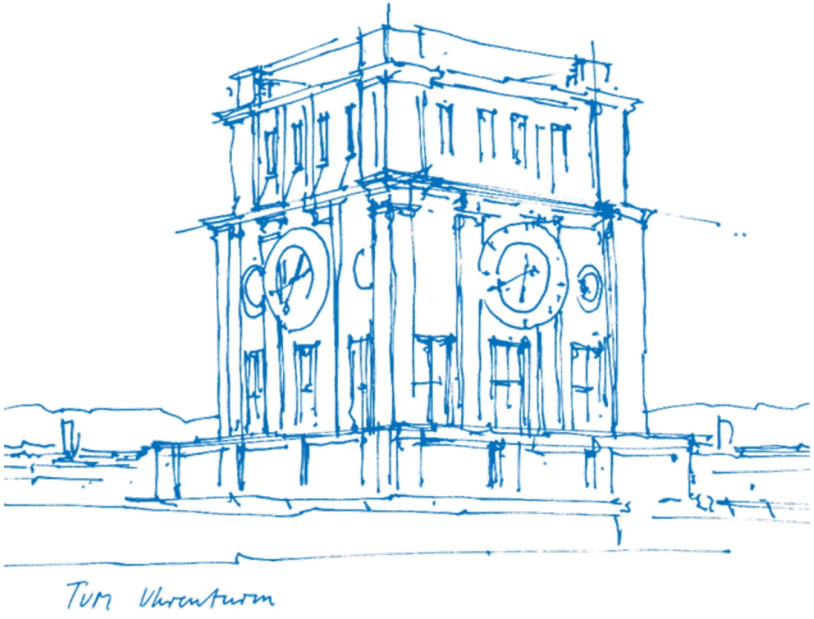
On Joint Estimation of Pose, Geometry and svBRDF from a Handheld Scanner

Matthias Stübinger

Department of Informatics, Technical University of Munich (TUM)

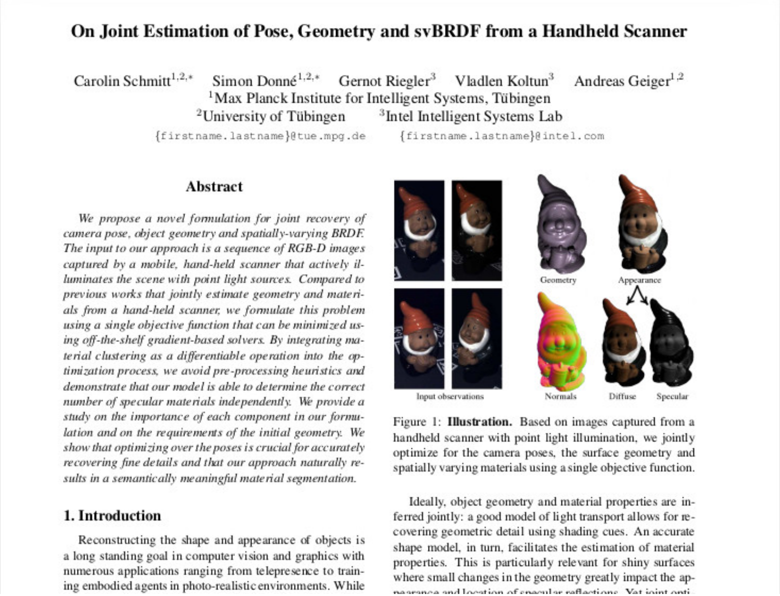
Computer Vision Seminar

October 6, 2020



On Joint Estimation of Pose, Geometry and svBRDF from a Handheld Scanner

- by Carolin Schmitt, Simon Donne, Gernot Riegler, Vladlen Koltun, Andreas Geiger
- Presented at CVPR 2020, published in the proceedings
- `https://openaccess.thecvf.com/content_CVPR_2020/html/Schmitt_On_Joint_Estimation_of_Pose_Geometry_and_svBRDF_From_a_CVPR_2020_paper.html`
- Video demonstration: `http://www.youtube.com/watch?v=_xxSQPD9qU0`



On Joint Estimation of Pose, Geometry and svBRDF from a Handheld Scanner

Carolin Schmitt[1,2,*] Simon Donné[1,2,*] Gernot Riegler[3] Vladlen Koltun[3] Andreas Geiger[1,2]
[1]Max Planck Institute for Intelligent Systems, Tübingen
[2]University of Tübingen [3]Intel Intelligent Systems Lab
{firstname.lastname}@tue.mpg.de {firstname.lastname}@intel.com

Abstract

We propose a novel formulation for joint recovery of camera pose, object geometry and spatially-varying BRDF. The input to our approach is a sequence of RGB-D images captured by a mobile, hand-held scanner that actively illuminates the scene with point light sources. Compared to previous works that jointly estimate geometry and materials from a hand-held scanner, we formulate this problem using a single objective function that can be minimized using off-the-shelf gradient-based solvers. By integrating material clustering as a differentiable operation into the optimization process, we avoid pre-processing heuristics and demonstrate that our model is able to determine the correct number of specular materials independently. We provide a study on the importance of each component in our formulation and on the requirements of the initial geometry. We show that optimizing over the poses is crucial for accurately recovering fine details and that our approach naturally results in a semantically meaningful material segmentation.

1. Introduction

Reconstructing the shape and appearance of objects is a long standing goal in computer vision and graphics with numerous applications ranging from telepresence to training embodied agents in photo-realistic environments. While

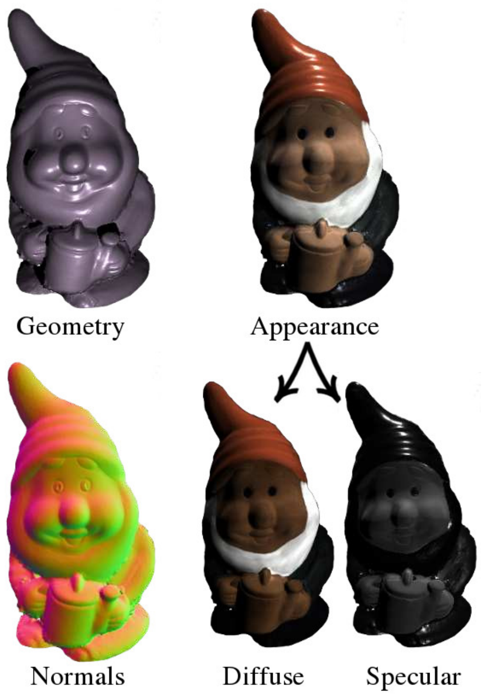
Figure 1: **Illustration.** Based on images captured from a handheld scanner with point light illumination, we jointly optimize for the camera poses, the surface geometry and spatially varying materials using a single objective function.

Ideally, object geometry and material properties are inferred jointly: a good model of light transport allows for recovering geometric detail using shading cues. An accurate shape model, in turn, facilitates the estimation of material properties. This is particularly relevant for shiny surfaces where small changes in the geometry greatly impact the appearance and location of specular reflections. Yet joint opti-

What's the aim of the method?

"On Joint Estimation of Pose, Geometry and svBRDF from a Handheld Scanner"

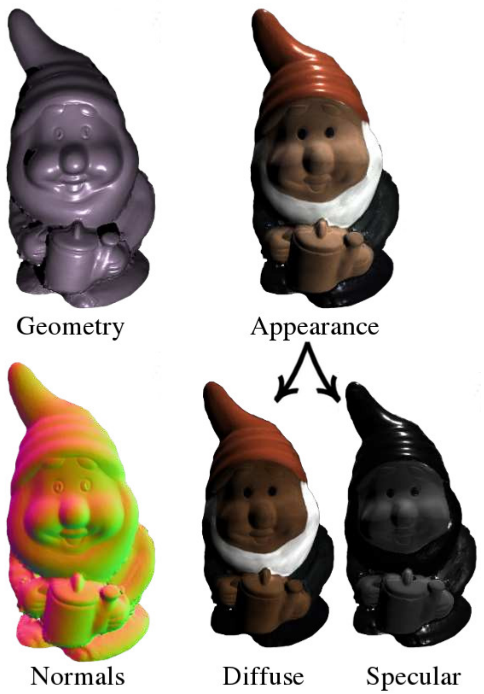
- Pose: Camera Rotation/Translation



What's the aim of the method?

"On Joint Estimation of Pose, Geometry and svBRDF from a Handheld Scanner"

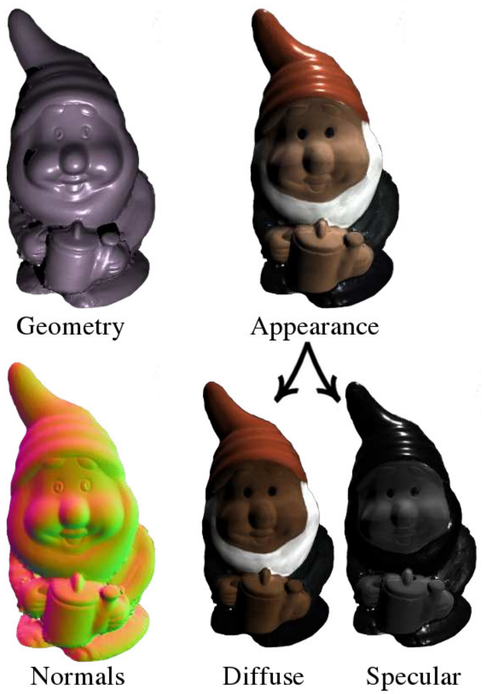
- Pose: Camera Rotation/Translation
- Geometry: Geometry of the Scene



What’s the aim of the method?

“On Joint Estimation of Pose, Geometry and svBRDF from a Handheld Scanner”

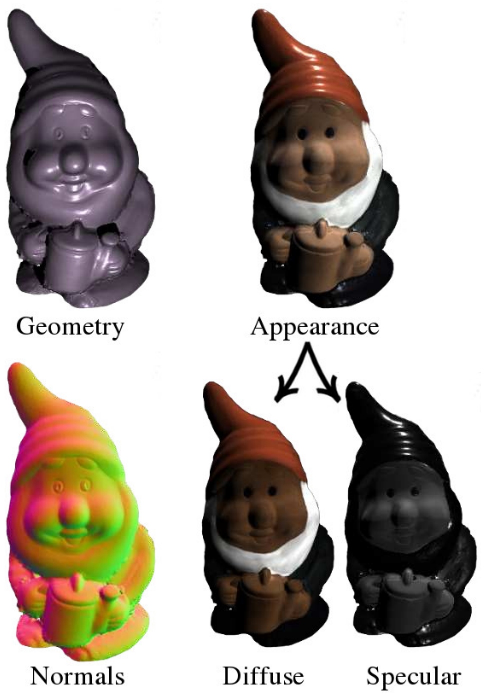
- Pose: Camera Rotation/Translation
- Geometry: Geometry of the Scene
- svBRDR: ***s****patially* ***v****arying* ***B****idirectional* ***R****eflectance* ***D****istribution* ***F****unction*
  (instead of “just” colour)



What’s the aim of the method?

“On Joint Estimation of Pose, Geometry and svBRDF from a Handheld Scanner”

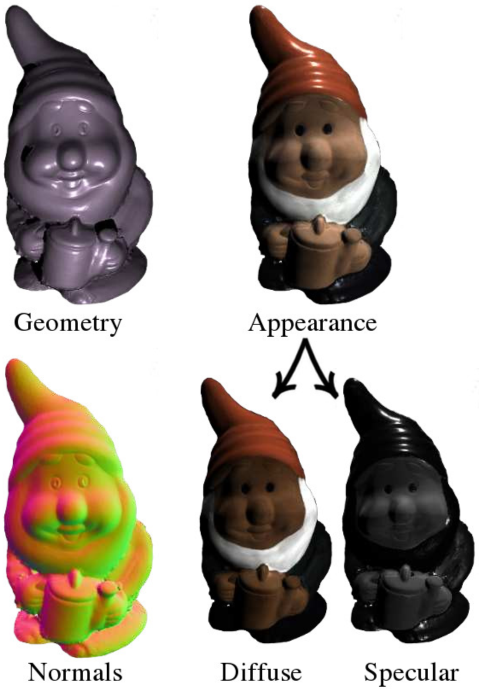
- Pose: Camera Rotation/Translation
- Geometry: Geometry of the Scene
- svBRDR: ***s****patially* ***v****arying* ***B****idirectional* ***R****eflectance* ***D****istribution* ***F****unction* (instead of “just” colour)
- Joint Estimation: Joint Optimisation



What’s the aim of the method?

“On Joint Estimation of Pose, Geometry and svBRDF from a Handheld Scanner”

- Pose: Camera Rotation/Translation
- Geometry: Geometry of the Scene
- svBRDR: ***s**patially **v**arying **B**idirectional **R**eflectance **D**istribution **F**unction* (instead of “just” colour)
- Joint Estimation: Joint Optimisation
- from a Handheld Scanner: previous work assumed multiple images from the same position, i.e. a tripod / camera rig

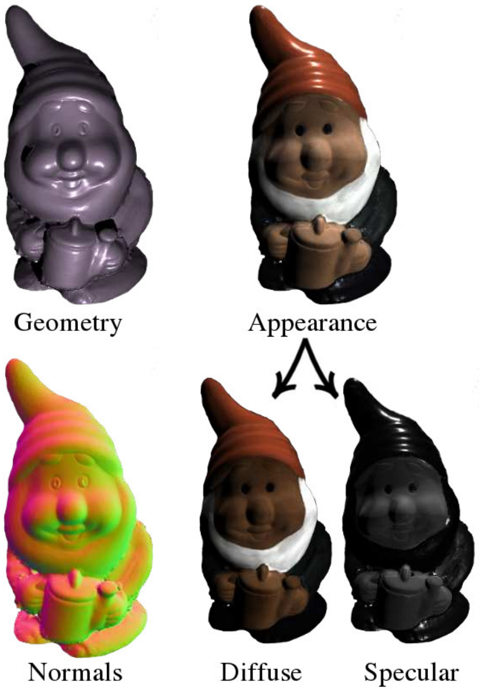


What’s the aim of the method?

“On Joint Estimation of Pose, Geometry and svBRDF from a Handheld Scanner”

- Pose: Camera Rotation/Translation
- Geometry: Geometry of the Scene
- svBRDR: ***s****patially* ***v****arying* ***B****idirectional* ***R****eflectance* ***D****istribution* ***F****unction* (instead of “just” colour)
- Joint Estimation: Joint Optimisation
- from a Handheld Scanner: previous work assumed multiple images from the same position, i.e. a tripod / camera rig

Additional Assumptions



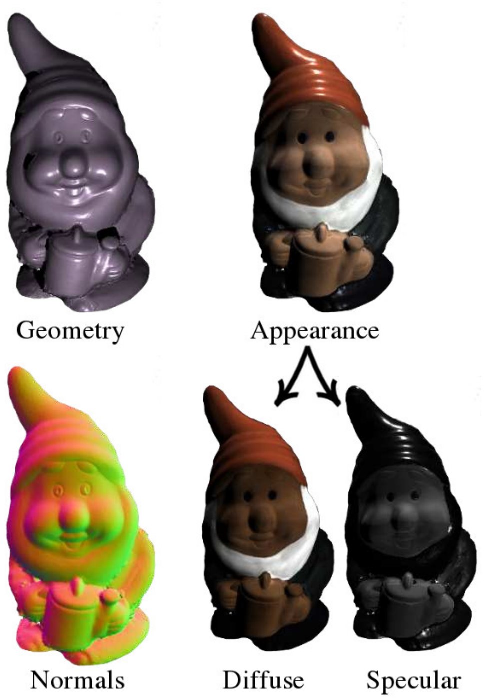
What’s the aim of the method?

“On Joint Estimation of Pose, Geometry and svBRDF from a Handheld Scanner”

- Pose: Camera Rotation/Translation
- Geometry: Geometry of the Scene
- svBRDR: ***s**patially **v**arying **B**idirectional **R**eflectance **D**istribution **F**unction* (instead of “just” colour)
- Joint Estimation: Joint Optimisation
- from a Handheld Scanner: previous work assumed multiple images from the same position, i.e. a tripod / camera rig

Additional Assumptions

- RGBD camera



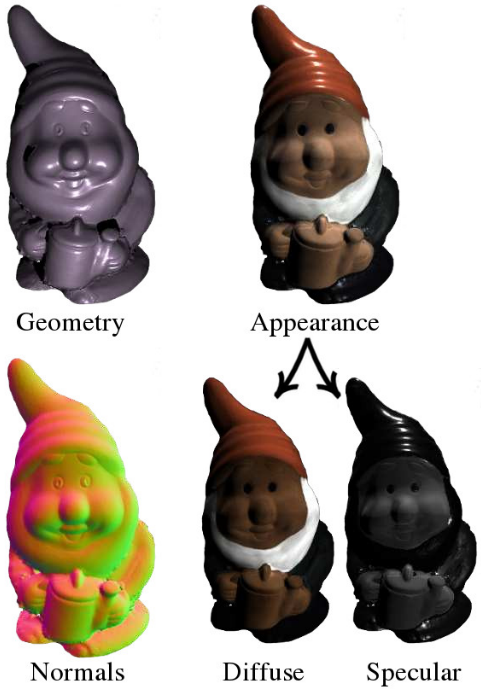
What’s the aim of the method?

“On Joint Estimation of Pose, Geometry and svBRDF from a Handheld Scanner”

- Pose: Camera Rotation/Translation
- Geometry: Geometry of the Scene
- svBRDR: ***s**patially **v**arying **B**idirectional **R**eflectance **D**istribution **F**unction* (instead of “just” colour)
- Joint Estimation: Joint Optimisation
- from a Handheld Scanner: previous work assumed multiple images from the same position, i.e. a tripod / camera rig

Additional Assumptions

- RGBD camera
- Exactly one point light source in each input image



Model

$$\mathcal{X} = \{\{(z_p, n_p, f_p)\}_{p=1}^{P}, \{\pi_i\}_{i=2}^{N}\}$$

- N undistorted images from a pinhole camera with vignetting removed; the first is called *reference view*
- π_i: projective mapping from view i back to the reference view
- z_p: depth for every pixel
- n_p: normals for every pixel
- f_p: material for every pixel

Representation of Geometry and Depth

Surface points are defined as depth $Z_1 = \{z_p\}$ of pixels p

- The depth is always given in the reference view

Representation of Geometry and Depth

Surface points are defined as depth $Z_1 = \{z_p\}$ of pixels p

- The depth is always given in the reference view
- Can derive the 3D point positions by doing an inverse projection

Representation of Geometry and Depth

Surface points are defined as depth $Z_1 = \{z_p\}$ of pixels p

- The depth is always given in the reference view
- Can derive the 3D point positions by doing an inverse projection
- Simpler than having “real” 3D points

Representation of Geometry and Depth

Surface points are defined as depth $Z_1 = \{z_p\}$ of pixels p

- The depth is always given in the reference view
- Can derive the 3D point positions by doing an inverse projection
- Simpler than having “real” 3D points
- Only represents what is visible from the reference view

Representation of Geometry and Depth

Surface points are defined as depth $Z_1 = \{z_p\}$ of pixels p

- The depth is always given in the reference view
- Can derive the 3D point positions by doing an inverse projection
- Simpler than having “real” 3D points
- Only represents what is visible from the reference view

We also have normals $N_1 = \{n_p\}$ in the reference view

Representation of Geometry and Depth

Surface points are defined as depth $Z_1 = \{z_p\}$ of pixels p

- The depth is always given in the reference view
- Can derive the 3D point positions by doing an inverse projection
- Simpler than having “real” 3D points
- Only represents what is visible from the reference view

We also have normals $N_1 = \{n_p\}$ in the reference view

- Represented as unit vectors

Representation of Geometry and Depth

Surface points are defined as depth $Z_1 = \{z_p\}$ of pixels p

- The depth is always given in the reference view
- Can derive the 3D point positions by doing an inverse projection
- Simpler than having “real” 3D points
- Only represents what is visible from the reference view

We also have normals $N_1 = \{n_p\}$ in the reference view

- Represented as unit vectors
- Rotated slightly in each optimisation step

Representation of Geometry and Depth

Surface points are defined as depth $Z_1 = \{z_p\}$ of pixels p

- The depth is always given in the reference view
- Can derive the 3D point positions by doing an inverse projection
- Simpler than having “real” 3D points
- Only represents what is visible from the reference view

We also have normals $N_1 = \{n_p\}$ in the reference view

- Represented as unit vectors
- Rotated slightly in each optimisation step
- Integrate into depth

svBRDF

Models the fraction of light reflected from direction ω^{in} into direction ω^{out} at a pixel p:

$$f_p(n_p, \omega^{\text{in}}, \omega^{\text{out}}) = d_p + s_p \frac{D(r_p) G(n_p, \omega^{\text{in}}, \omega^{\text{out}}, r_p)}{\pi (n_p \cdot \omega^{\text{in}})(n_p \cdot \omega^{\text{out}})}$$

- Functions D and G are called *microfacet slope distribution* and *geometric attenuation factor*

svBRDF

Models the fraction of light reflected from direction ω^{in} into direction ω^{out} at a pixel p:

$$f_p(n_p, \omega^{\text{in}}, \omega^{\text{out}}) = d_p + s_p \frac{D(r_p)G(n_p, \omega^{\text{in}}, \omega^{\text{out}}, r_p)}{\pi(n_p \cdot \omega^{\text{in}})(n_p \cdot \omega^{\text{out}})}$$

- Functions D and G are called *microfacet slope distribution* and *geometric attenuation factor*
- 3 parameters:
  - $d_p \in \mathbb{R}^3$ is the diffuse albedo
  - $s_p \in \mathbb{R}^3$ the specular albedo
  - $r_p \in \mathbb{R}$ the roughness of the surface

svBRDF

Models the fraction of light reflected from direction ω^{in} into direction ω^{out} at a pixel p:

$$f_p(n_p, \omega^{\text{in}}, \omega^{\text{out}}) = d_p + s_p \frac{D(r_p)G(n_p, \omega^{\text{in}}, \omega^{\text{out}}, r_p)}{\pi(n_p \cdot \omega^{\text{in}})(n_p \cdot \omega^{\text{out}})}$$

- Functions D and G are called *microfacet slope distribution* and *geometric attenuation factor*
- 3 parameters:
  - $d_p \in \mathbb{R}^3$ is the diffuse albedo
  - $s_p \in \mathbb{R}^3$ the specular albedo
  - $r_p \in \mathbb{R}$ the roughness of the surface

$\rightarrow$ These are optimised as part of the method

Specular Materials

In practice, the paper takes some simplifying assumptions:

Specular Materials

In practice, the paper takes some simplifying assumptions:

- exactly one point light

Specular Materials

In practice, the paper takes some simplifying assumptions:

- exactly one point light
- Fresnel effect cannot occur with only one light source which is close to the camera

Specular Materials

In practice, the paper takes some simplifying assumptions:

- exactly one point light
- Fresnel effect cannot occur with only one light source which is close to the camera
- (s_p, r_p) can vary only between a few specular base materials, with weights $\alpha_p^t \in [0, 1]$:

$$\begin{pmatrix} s_p \\ r_p \end{pmatrix} = \sum_{t=1}^{T} \alpha_p^t \begin{pmatrix} s_t \\ r_t \end{pmatrix}$$

- $T \leq 3$ is enough for all except very complex objects

Specular Materials

In practice, the paper takes some simplifying assumptions:

- exactly one point light
- Fresnel effect cannot occur with only one light source which is close to the camera
- (s_p, r_p) can vary only between a few specular base materials, with weights $\alpha_p^t \in [0, 1]$:

$$\begin{pmatrix} s_p \\ r_p \end{pmatrix} = \sum_{t=1}^{T} \alpha_p^t \begin{pmatrix} s_t \\ r_t \end{pmatrix}$$

- $T \leq 3$ is enough for all except very complex objects
- svBRDF is fully determined by
  – diffuse and specular material weights: $\{d_p, \alpha_p\}_{p=1}^{P}$
  – the specular materials: $\{(s_p, r_p)\}_{t=1}^{T}$

Specular Materials

In practice, the paper takes some simplifying assumptions:

- exactly one point light
- Fresnel effect cannot occur with only one light source which is close to the camera
- (s_p, r_p) can vary only between a few specular base materials, with weights $\alpha_p^t \in [0, 1]$:

$$\begin{pmatrix} s_p \\ r_p \end{pmatrix} = \sum_{t=1}^{T} \alpha_p^t \begin{pmatrix} s_t \\ r_t \end{pmatrix}$$

- $T \leq 3$ is enough for all except very complex objects
- svBRDF is fully determined by
  - diffuse and specular material weights: $\{d_p, \alpha_p\}_{p=1}^{P}$
  - the specular materials: $\{(s_p, r_p)\}_{t=1}^{T}$
- only these are optimised

Optimisation: Basic Idea

Optimisation: Basic Idea

- Don’t try to derive new types of information from input (e.g. via 8-point algorithm)

Optimisation: Basic Idea

- Don’t try to derive new types of information from input (e.g. via 8-point algorithm)
- Instead take rough guesses as input, and gradually optimise these within a set of constraints

Optimisation: Basic Idea

- Don’t try to derive new types of information from input (e.g. via 8-point algorithm)
- Instead take rough guesses as input, and gradually optimise these within a set of constraints
- Use established optimisation methods

Constraints:

Optimisation: Basic Idea

- Don’t try to derive new types of information from input (e.g. via 8-point algorithm)
- Instead take rough guesses as input, and gradually optimise these within a set of constraints
- Use established optimisation methods

Constraints:

- Photoconsistency

Optimisation: Basic Idea

- Don’t try to derive new types of information from input (e.g. via 8-point algorithm)
- Instead take rough guesses as input, and gradually optimise these within a set of constraints
- Use established optimisation methods

Constraints:

- Photoconsistency
- Geometric Consistency

Optimisation: Basic Idea

- Don't try to derive new types of information from input (e.g. via 8-point algorithm)
- Instead take rough guesses as input, and gradually optimise these within a set of constraints
- Use established optimisation methods

Constraints:

- Photoconsistency
- Geometric Consistency
- Depth Compatibility

Optimisation: Basic Idea

- Don’t try to derive new types of information from input (e.g. via 8-point algorithm)
- Instead take rough guesses as input, and gradually optimise these within a set of constraints
- Use established optimisation methods

Constraints:

- Photoconsistency
- Geometric Consistency
- Depth Compatibility
- Normal Smoothness

Optimisation: Basic Idea

- Don’t try to derive new types of information from input (e.g. via 8-point algorithm)
- Instead take rough guesses as input, and gradually optimise these within a set of constraints
- Use established optimisation methods

Constraints:

- Photoconsistency
- Geometric Consistency
- Depth Compatibility
- Normal Smoothness
- Material Smoothness

Constraints: Photoconsistency

$$\psi_P(\mathcal{X}) = \frac{1}{N} \sum_i \sum_p \left\| \varphi_p^i \left[\mathcal{I}_i\left(\pi_i(x_p)\right) - f_p\left(n_p, \omega_i^{\text{in}}(x_p), w_i^{\text{out}}(x_p)\right) \cdot \frac{a_i(x_p) n_p^T \omega_i^{\text{in}}(x_p)}{d_i(x_p)^2} L \right] \right\|_1$$

Constraints: Photoconsistency

$$\psi_P(\mathcal{X}) = \frac{1}{N}\sum_i \sum_p \left\| \varphi_p^i \left[\mathcal{I}_i\left(\pi_i(x_p)\right) - f_p\left(n_p, \omega_i^{\text{in}}(x_p), w_i^{\text{out}}(x_p)\right) \cdot \frac{a_i(x_p) n_p^T \omega_i^{\text{in}}(x_p)}{d_i(x_p)^2} L \right] \right\|_1$$

- Does the model fit the observations $\mathcal{I}_i$ in each pixel p?

Constraints: Photoconsistency

$$\psi_P(\mathcal{X}) = \frac{1}{N} \sum_i \sum_p \left\| \varphi_p^i \left[\mathcal{I}_i \left(\pi_i(x_p) \right) - f_p \left(n_p, \omega_i^{\text{in}}(x_p), w_i^{\text{out}}(x_p) \right) \cdot \frac{a_i(x_p) n_p^T \omega_i^{\text{in}}(x_p)}{d_i(x_p)^2} L \right] \right\|_1$$

- Does the model fit the observations $\mathcal{I}_i$ in each pixel p?
- Takes the difference of observation and rendering output

Constraints: Photoconsistency

$$\psi_P(\mathcal{X}) = \frac{1}{N} \sum_i \sum_p \left\| \varphi_p^i \left[\mathcal{I}_i \left(\pi_i(x_p) \right) - f_p \left(n_p, \omega_i^{\text{in}}(x_p), w_i^{\text{out}}(x_p) \right) \cdot \frac{a_i(x_p) n_p^T \omega_i^{\text{in}}(x_p)}{d_i(x_p)^2} L \right] \right\|_1$$

- Does the model fit the observations $\mathcal{I}_i$ in each pixel p?
- Takes the difference of observation and rendering output
- $\varphi_p^i = 1$ iff p contains an observation and is visible in image i

Constraints: Photoconsistency

$$\psi_P(\mathcal{X}) = \frac{1}{N} \sum_i \sum_p \left\| \varphi_p^i \left[\mathcal{I}_i \left(\pi_i(x_p) \right) - f_p \left(n_p, \omega_i^{\text{in}}(x_p), w_i^{\text{out}}(x_p) \right) \cdot \frac{a_i(x_p) n_p^T \omega_i^{\text{in}}(x_p)}{d_i(x_p)^2} L \right] \right\|_1$$

- Does the model fit the observations $\mathcal{I}_i$ in each pixel p?
- Takes the difference of observation and rendering output
- $\varphi_p^i = 1$ iff p contains an observation and is visible in image i
- $f_p \left(n_p, \omega_i^{\text{in}}(x_p), w_i^{\text{out}}(x_p) \right)$ is the svBRDF term assuming a single point light

Constraints: Photoconsistency

$$\psi_P(\mathcal{X}) = \frac{1}{N}\sum_i \sum_p \left\| \varphi_p^i \left[\mathcal{I}_i\left(\pi_i(x_p)\right) - f_p\left(n_p, \omega_i^{\text{in}}(x_p), w_i^{\text{out}}(x_p)\right) \cdot \frac{a_i(x_p) n_p^T \omega_i^{\text{in}}(x_p)}{d_i(x_p)^2} L \right] \right\|_1$$

- Does the model fit the observations $\mathcal{I}_i$ in each pixel p?
- Takes the difference of observation and rendering output
- $\varphi_p^i = 1$ iff p contains an observation and is visible in image i
- $f_p\left(n_p, \omega_i^{\text{in}}(x_p), w_i^{\text{out}}(x_p)\right)$ is the svBRDF term assuming a single point light
- $\frac{a_i(x_p) n_p^T \omega_i^{\text{in}}(x_p)}{d_i(x_p)^2} L$ gives the intesity of that point light

Constraints: Geometric Consistency

$$\psi_{\mathcal{G}}(\mathcal{X}) = -\sum_{p} \vec{n}_p^T \left(\frac{\frac{\partial z_p}{\partial x} \times \frac{\partial z_p}{\partial y}}{\|\frac{\partial z_p}{\partial x} \times \frac{\partial z_p}{\partial y}\|_2} \right) \qquad \frac{\partial z_p}{\partial x} \propto \left[1, 0, \vec{\nabla} \mathcal{Z}_1(\pi_1(\vec{x}_p))^T [f/z_p, 0]^T\right]^T$$

- Geometry and Normals must be consistent

Constraints: Geometric Consistency

$$\psi_{\mathcal{G}}(\mathcal{X}) = -\sum_{p} \vec{n}_p^T \left(\frac{\frac{\partial z_p}{\partial x} \times \frac{\partial z_p}{\partial y}}{\| \frac{\partial z_p}{\partial x} \times \frac{\partial z_p}{\partial y} \|_2} \right) \qquad \frac{\partial z_p}{\partial x} \propto \left[1, 0, \vec{\nabla} \mathcal{Z}_1(\pi_1(\vec{x}_p))^T [f/z_p, 0]^T \right]^T$$

- Geometry and Normals must be consistent
- Normals $\{n_p\}$ must integrate to the depth map $\{z_p\}$

Constraints: Geometric Consistency

$$\psi_{\mathcal{G}}(\mathcal{X}) = -\sum_{p} \vec{n}_p^T \left(\frac{\frac{\partial z_p}{\partial x} \times \frac{\partial z_p}{\partial y}}{\|\frac{\partial z_p}{\partial x} \times \frac{\partial z_p}{\partial y}\|_2} \right) \qquad \frac{\partial z_p}{\partial x} \propto \left[1, 0, \vec{\nabla}\mathcal{Z}_1(\pi_1(\vec{x}_p))^T [f/z_p, 0]^T\right]^T$$

- Geometry and Normals must be consistent
- Normals $\{n_p\}$ must integrate to the depth map $\{z_p\}$
- Align n_p and the cross product of surface tangents by minimising the scalar product

Constraints: Geometric Consistency

$$\psi_{\mathcal{G}}(\mathcal{X}) = -\sum_p \vec{n}_p^T \left(\frac{\frac{\partial z_p}{\partial x} \times \frac{\partial z_p}{\partial y}}{\|\frac{\partial z_p}{\partial x} \times \frac{\partial z_p}{\partial y}\|_2} \right) \qquad \frac{\partial z_p}{\partial x} \propto \left[1, 0, \vec{\nabla}\mathcal{Z}_1(\pi_1(\vec{x}_p))^T [f/z_p, 0]^T\right]^T$$

- Geometry and Normals must be consistent
- Normals $\{n_p\}$ must integrate to the depth map $\{z_p\}$
- Align n_p and the cross product of surface tangents by minimising the scalar product
- “Soft coupling” between normals and depth

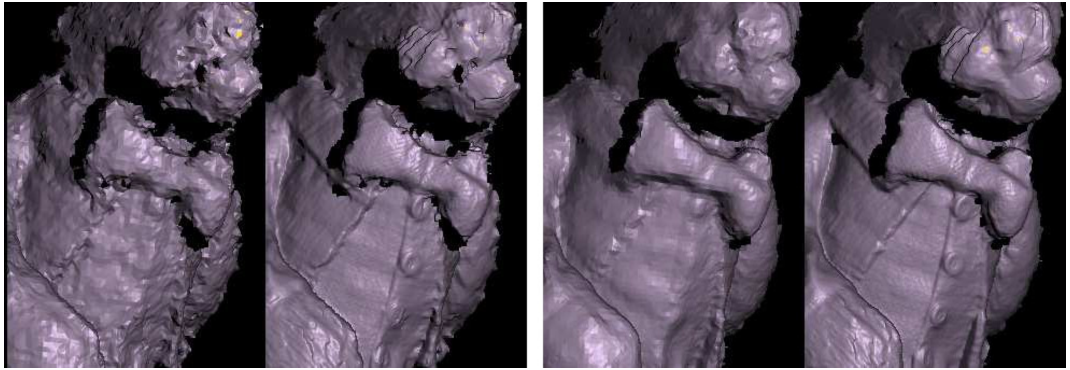
Constraints: Geometric Consistency

$$\psi_{\mathcal{G}}(\mathcal{X}) = -\sum_{p} \vec{n}_p^T \left(\frac{\frac{\partial z_p}{\partial x} \times \frac{\partial z_p}{\partial y}}{\|\frac{\partial z_p}{\partial x} \times \frac{\partial z_p}{\partial y}\|_2} \right) \qquad \frac{\partial z_p}{\partial x} \propto \left[1, 0, \vec{\nabla}\mathcal{Z}_1(\pi_1(\vec{x}_p))^T [f/z_p, 0]^T\right]^T$$

- Geometry and Normals must be consistent
- Normals $\{n_p\}$ must integrate to the depth map $\{z_p\}$
- Align n_p and the cross product of surface tangents by minimising the scalar product
- “Soft coupling” between normals and depth
- Could also enforce equality, but makes the method less robust

Constraints: Depth Compatibility

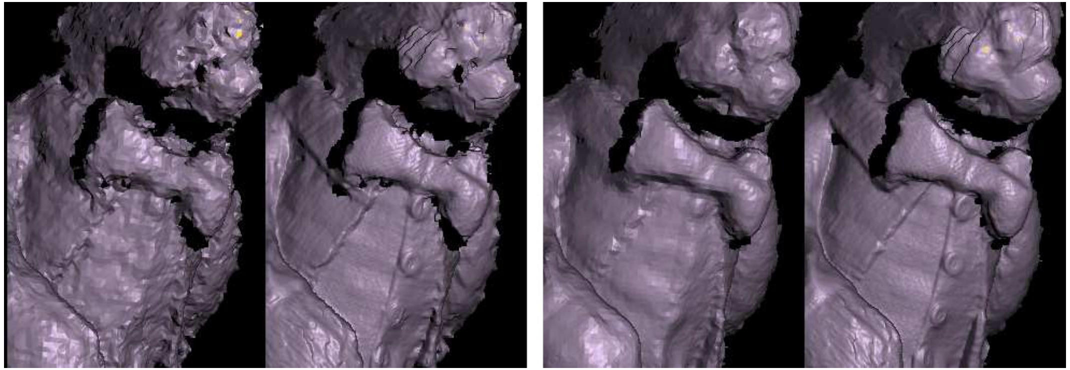
$$\psi_{\mathcal{D}}(\mathcal{X}) = \sum_{p} \| z_p - \mathcal{Z}_1(u_p, v_p) \|_2^2$$



Constraints: Depth Compatibility

$$\psi_{\mathcal{D}}(\mathcal{X}) = \sum_{p} \|z_p - \mathcal{Z}_1(u_p, v_p)\|_2^2$$

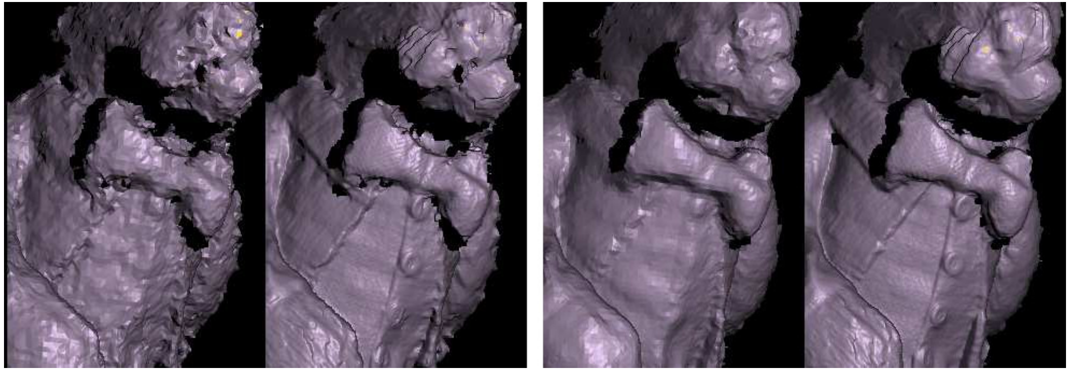
- Regularise against depth measurements in the reference view



Constraints: Depth Compatibility

$$\psi_{\mathcal{D}}(\mathcal{X}) = \sum_{p} \|z_p - \mathcal{Z}_1(u_p, v_p)\|_2^2$$

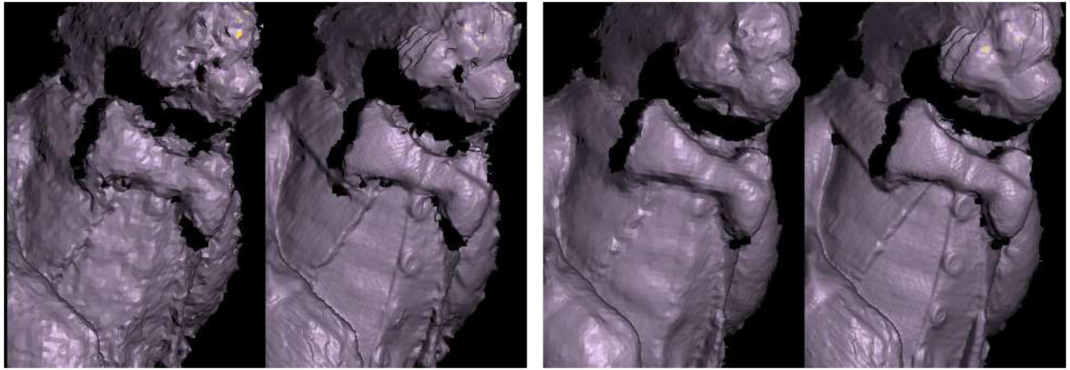
- Regularise against depth measurements in the reference view
- Before optimisation, several measured depth maps can be integrated into the reference view



Constraints: Depth Compatibility

$$\psi_{\mathcal{D}}(\mathcal{X}) = \sum_{p} \| z_p - \mathcal{Z}_1(u_p, v_p) \|_2^2$$

- Regularise against depth measurements in the reference view
- Before optimisation, several measured depth maps can be integrated into the reference view
- Final result will improve on the bare measurements through shading cues



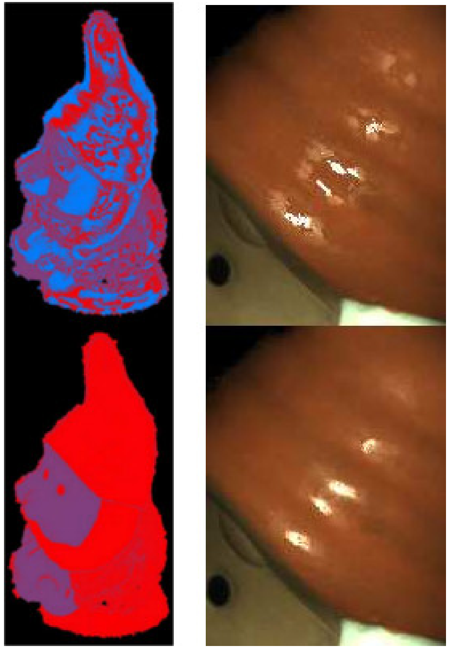
Constraints: Normal Smoothness

$$\psi_{\mathcal{N}}(\mathcal{X}) = \sum_{p \sim q} \|n_p - n_q\|_2^2$$

- Standard smoothness term to encourage smooth surfaces
- Minimise the difference of normals of adjacent pixels $p \sim q$

Constraints: Material Smoothness

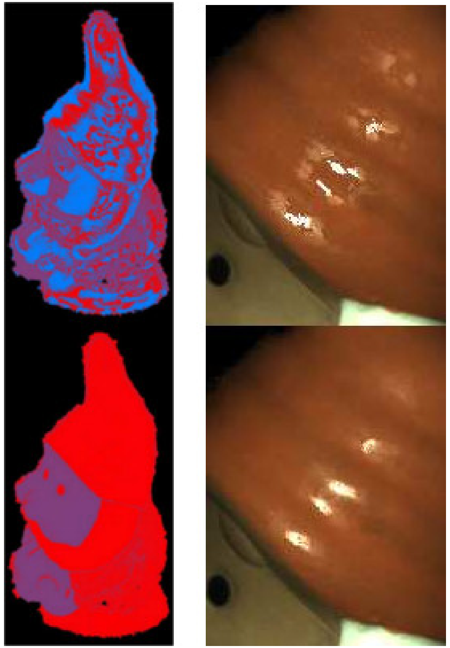
$$\psi_{\mathcal{M}}(\mathcal{X}) = \sum_p \left\| \alpha_p - \frac{\sum_q \alpha_q w_q k_{q,p}}{\sum_q q_q k_{p,q}} \right\|_1 - \sum_p \left\| \alpha_p - \frac{1}{P} \sum_q \alpha_q \right\|_1$$



Constraints: Material Smoothness

$$\psi_{\mathcal{M}}(\mathcal{X}) = \sum_p \left\| \alpha_p - \frac{\sum_q \alpha_q w_q k_{q,p}}{\sum_q q_q k_{p,q}} \right\|_1 - \sum_p \left\| \alpha_p - \frac{1}{P}\sum_q \alpha_q \right\|_1$$

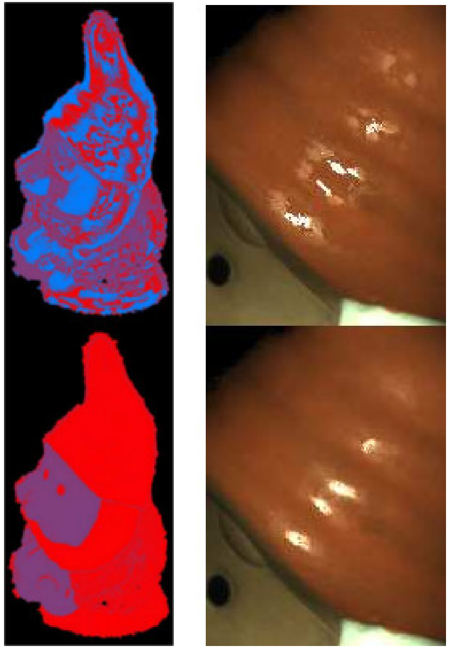
- Only a few pixel will actually contain specular information to reconstruct



Constraints: Material Smoothness

$$\psi_{\mathcal{M}}(\mathcal{X}) = \sum_p \left\| \alpha_p - \frac{\sum_q \alpha_q w_q k_{q,p}}{\sum_q q_q k_{p,q}} \right\|_1 - \sum_p \left\| \alpha_p - \frac{1}{P} \sum_q \alpha_q \right\|_1$$

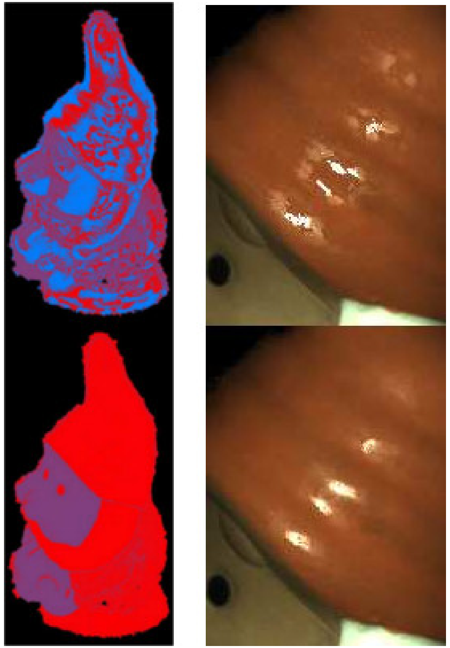
- Only a few pixel will actually contain specular information to reconstruct
- Assumption: areas with similar diffuse properties have similar specular properties



Constraints: Material Smoothness

$$\psi_{\mathcal{M}}(\mathcal{X}) = \sum_p \left\| \alpha_p - \frac{\sum_q \alpha_q w_q k_{q,p}}{\sum_q q_q k_{p,q}} \right\|_1 - \sum_p \left\| \alpha_p - \frac{1}{P} \sum_q \alpha_q \right\|_1$$

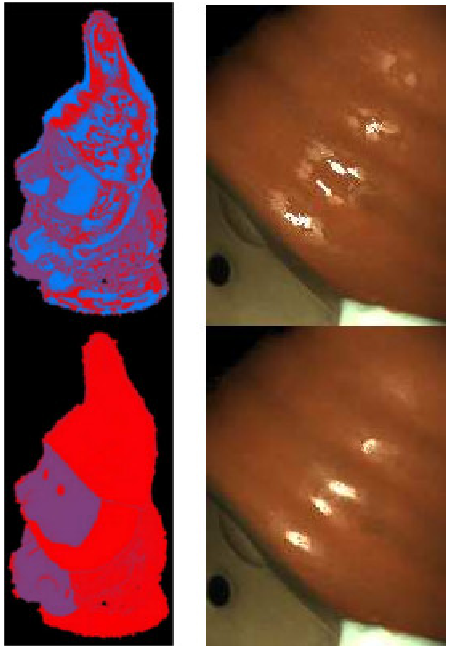
- Only a few pixel will actually contain specular information to reconstruct
- Assumption: areas with similar diffuse properties have similar specular properties
- Therefore regularise across regions with similar appearance



Constraints: Material Smoothness

$$\psi_{\mathcal{M}}(\mathcal{X}) = \sum_p \left\| \alpha_p - \frac{\sum_q \alpha_q w_q k_{q,p}}{\sum_q q_q k_{p,q}} \right\|_1 - \sum_p \left\| \alpha_p - \frac{1}{P} \sum_q \alpha_q \right\|_1$$

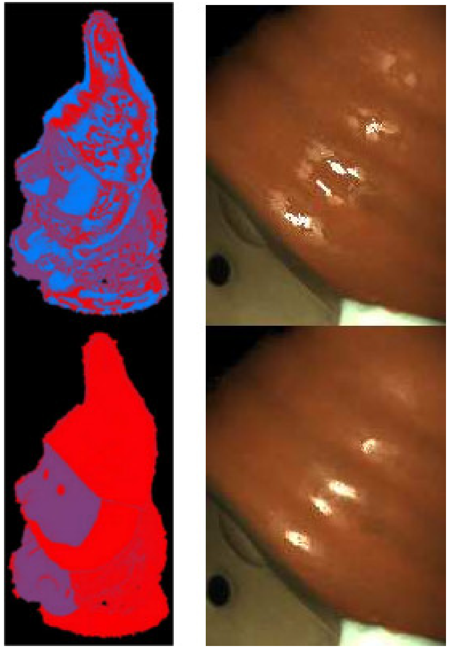
- Only a few pixel will actually contain specular information to reconstruct
- Assumption: areas with similar diffuse properties have similar specular properties
- Therefore regularise across regions with similar appearance
- $k_{p,q}$ is a Gaussian kernel



Constraints: Material Smoothness

$$\psi_{\mathcal{M}}(\mathcal{X}) = \sum_p \left\| \alpha_p - \frac{\sum_q \alpha_q w_q k_{q,p}}{\sum_q q_q k_{p,q}} \right\|_1 - \sum_p \left\| \alpha_p - \frac{1}{P} \sum_q \alpha_q \right\|_1$$

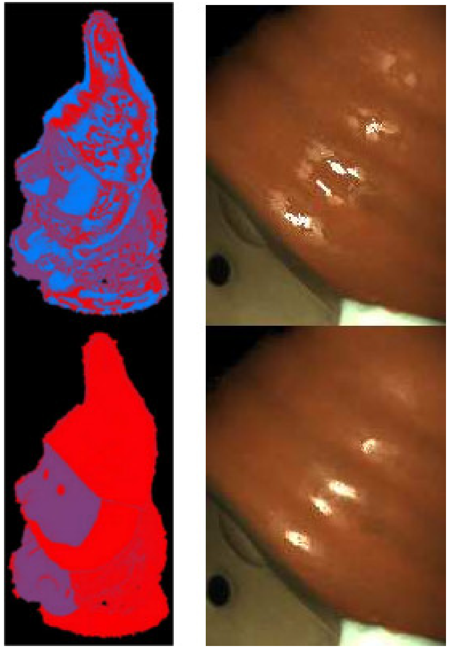
- Only a few pixel will actually contain specular information to reconstruct
- Assumption: areas with similar diffuse properties have similar specular properties
- Therefore regularise across regions with similar appearance
- $k_{p,q}$ is a Gaussian kernel
- $w_q = \max_i \mathrm{acos}^{-1}(n_q \cdot h^i_q)$ determines if q is a highlight in some input image



Constraints: Material Smoothness

$$\psi_{\mathcal{M}}(\mathcal{X}) = \sum_p \left\| \alpha_p - \frac{\sum_q \alpha_q w_q k_{q,p}}{\sum_q q_q k_{p,q}} \right\|_1 - \sum_p \left\| \alpha_p - \frac{1}{P} \sum_q \alpha_q \right\|_1$$

- Only a few pixel will actually contain specular information to reconstruct
- Assumption: areas with similar diffuse properties have similar specular properties
- Therefore regularise across regions with similar appearance
- $k_{p,q}$ is a Gaussian kernel
- $w_q = \max_i \text{acos}^{-1}(n_q \cdot h^i_q)$ determines if q is a highlight in some input image
- Also encourage material sparsity: maximise distance from the average weights (second term)



Optimisation: Putting it all together

$$\mathcal{X}^\star = \operatorname*{argmin}_{\mathcal{X}} \psi_\mathcal{P} + \psi_\mathcal{D} + \psi_\mathcal{N} + \psi_\mathcal{M}$$

Optimisation: Putting it all together

$$\mathcal{X}^\star = \operatorname*{argmin}_{\mathcal{X}} \psi_{\mathcal{P}} + \psi_{\mathcal{D}} + \psi_{\mathcal{N}} + \psi_{\mathcal{M}}$$

- Implemented with PyTorch, code runs on the GPU (with cuda)

Optimisation: Putting it all together

$$\mathcal{X}^{\star} = \underset{\mathcal{X}}{\text{argmin}}\ \psi_{\mathcal{P}} + \psi_{\mathcal{D}} + \psi_{\mathcal{N}} + \psi_{\mathcal{M}}$$

- Implemented with PyTorch, code runs on the GPU (with cuda)
- Optimisation with ADAM (Adaptive Moment Estimation)

Optimisation: Putting it all together

$$\mathcal{X}^\star = \underset{\mathcal{X}}{\text{argmin}}\ \psi_\mathcal{P} + \psi_\mathcal{D} + \psi_\mathcal{N} + \psi_\mathcal{M}$$

- Implemented with PyTorch, code runs on the GPU (with cuda)
- Optimisation with ADAM (Adaptive Moment Estimation)
- Code available under the MIT licence: `https://github.com/autonomousvision/handheld_svbrdf_geometry`

Optimisation: Putting it all together

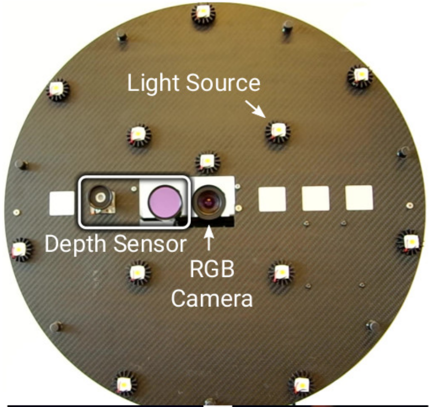
$$\mathcal{X}^\star = \underset{\mathcal{X}}{\text{argmin}}\ \psi_\mathcal{P} + \psi_\mathcal{D} + \psi_\mathcal{N} + \psi_\mathcal{M}$$

- Implemented with PyTorch, code runs on the GPU (with cuda)
- Optimisation with ADAM (Adaptive Moment Estimation)
- Code available under the MIT licence: `https://github.com/autonomousvision/handheld_svbrdf_geometry`

```
Traceback (most recent call last):
  File "main.py", line 358, in <module>
    run_experiment(settings_dict)
  File "main.py", line 218, in run_experiment
    for observation in tqdm(data_adapter.images, desc="Preloading training images")
  File "main.py", line 218, in <listcomp>
    for observation in tqdm(data_adapter.images, desc="Preloading training images")
  File "/home/matthias/Dokumente/TUM/Seminar Computer Vision/handheld_svbrdf_geometry/code/data.py", line 153, i
n get_image
    image = torch.tensor(np.transpose(image_data, (2,0,1)), dtype=torch.float32, device=self.device)
RuntimeError: CUDA out of memory. Tried to allocate 142.00 MiB (GPU 0; 1.95 GiB total capacity; 826.80 MiB alrea
dy allocated; 120.25 MiB free; 29.20 MiB cached)
Preloading training images:  93%|██████████████████████████████████████████    | 42/45 [00:09<00:00,  4.59it/s]
```

Initialisation

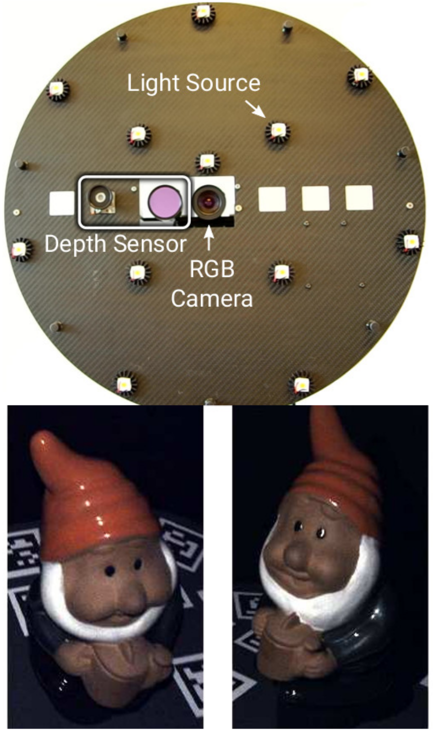
$$\mathcal{X} = \{\{(z_p, n_p, f_p)\}_{p=1}^{P}, \{\pi_i\}_{i=2}^{N}\}$$



Initialisation

$$\mathcal{X} = \{\{(z_p, n_p, f_p)\}_{p=1}^{P}, \{\pi_i\}_{i=2}^{N}\}$$

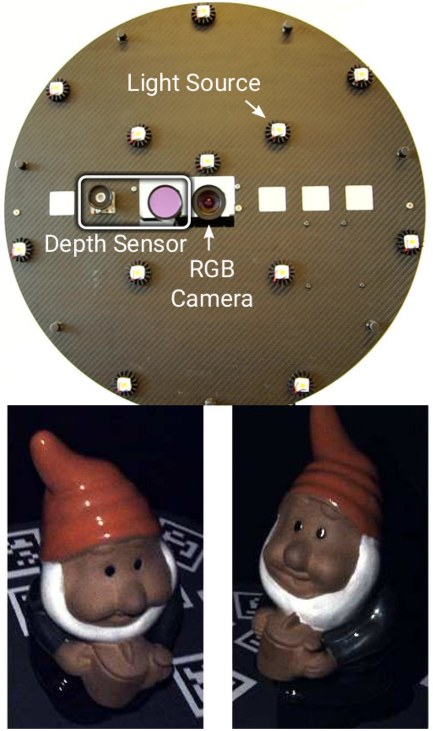
- 45 images per object, in a 30° cone around the reference view
  - Taken in sensor rig with LED point lights, ambient light negligible
  - Calibrated cameras, distortion and vignetting are removed



Initialisation

$$\mathcal{X} = \{\{(z_p, n_p, f_p)\}_{p=1}^{P}, \{\pi_i\}_{i=2}^{N}\}$$

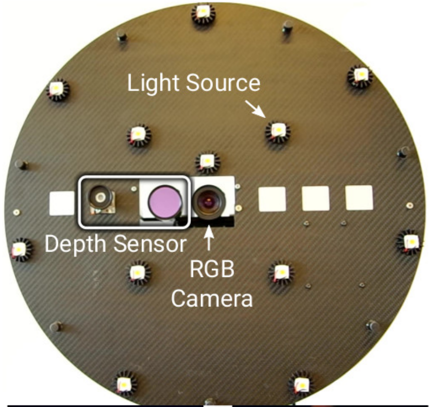
- 45 images per object, in a 30° cone around the reference view
  - Taken in sensor rig with LED point lights, ambient light negligible
  - Calibrated cameras, distortion and vignetting are removed
- Camera poses: reconstructed using tracking tags



Initialisation

$$\mathcal{X} = \{\{(z_p, n_p, f_p)\}_{p=1}^{P}, \{\pi_i\}_{i=2}^{N}\}$$

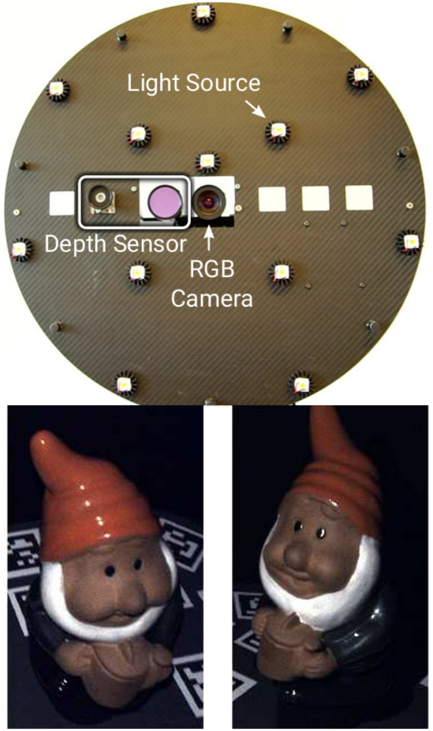
- 45 images per object, in a 30° cone around the reference view
  - Taken in sensor rig with LED point lights, ambient light negligible
  - Calibrated cameras, distortion and vignetting are removed
- Camera poses: reconstructed using tracking tags
- Geometry/depth: input from RGBD images, integrated with volumetric fusion



Initialisation

$$\mathcal{X} = \{\{(z_p, n_p, f_p)\}_{p=1}^{P}, \{\pi_i\}_{i=2}^{N}\}$$

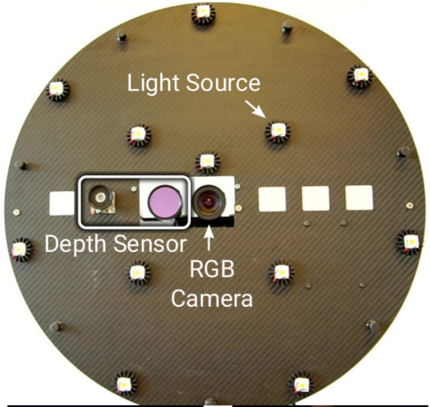
- 45 images per object, in a 30° cone around the reference view
  - Taken in sensor rig with LED point lights, ambient light negligible
  - Calibrated cameras, distortion and vignetting are removed
- Camera poses: reconstructed using tracking tags
- Geometry/depth: input from RGBD images, integrated with volumetric fusion
- Normals and Albedo: can be recovered in closed form in a Lambertian scene; specularity outliers are rejected using RANSAC



Initialisation

$$\mathcal{X} = \{\{(z_p, n_p, f_p)\}_{p=1}^{P}, \{\pi_i\}_{i=2}^{N}\}$$

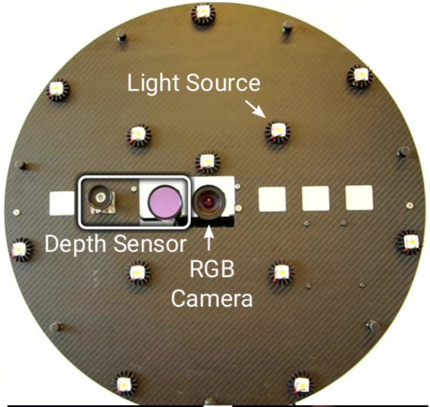
- 45 images per object, in a 30° cone around the reference view
  - Taken in sensor rig with LED point lights, ambient light negligible
  - Calibrated cameras, distortion and vignetting are removed
- Camera poses: reconstructed using tracking tags
- Geometry/depth: input from RGBD images, integrated with volumetric fusion
- Normals and Albedo: can be recovered in closed form in a Lambertian scene; specularity outliers are rejected using RANSAC
- Number of Base Materials: optimise for $T \in \{1, 2, 3\}$, choose model with the smallest photometric error



Initialisation

$$\mathcal{X} = \{\{(z_p, n_p, f_p)\}_{p=1}^{P}, \{\pi_i\}_{i=2}^{N}\}$$

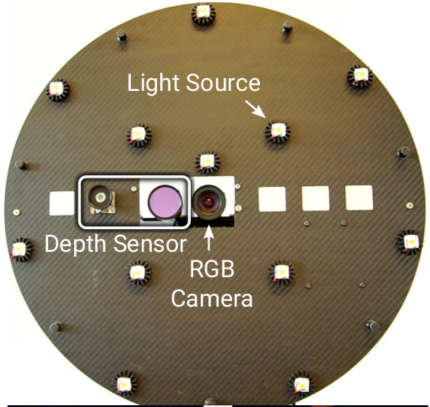
- 45 images per object, in a 30° cone around the reference view
  - Taken in sensor rig with LED point lights, ambient light negligible
  - Calibrated cameras, distortion and vignetting are removed
- Camera poses: reconstructed using tracking tags
- Geometry/depth: input from RGBD images, integrated with volumetric fusion
- Normals and Albedo: can be recovered in closed form in a Lambertian scene; specularity outliers are rejected using RANSAC
- Number of Base Materials: optimise for $T \in \{1, 2, 3\}$, choose model with the smallest photometric error
- Specular BRDF: set each pixel to a uniform mix of the base materials



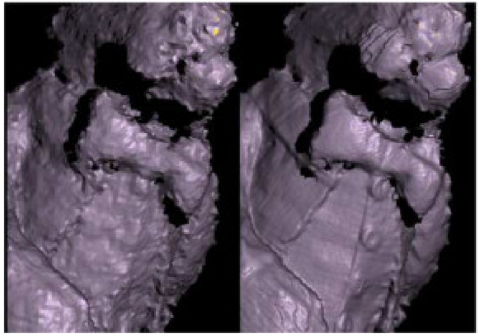
Initialisation

$$\mathcal{X} = \{\{(z_p, n_p, f_p)\}_{p=1}^{P}, \{\pi_i\}_{i=2}^{N}\}$$

- 45 images per object, in a 30° cone around the reference view
  - Taken in sensor rig with LED point lights, ambient light negligible
  - Calibrated cameras, distortion and vignetting are removed
- Camera poses: reconstructed using tracking tags
- Geometry/depth: input from RGBD images, integrated with volumetric fusion
- Normals and Albedo: can be recovered in closed form in a Lambertian scene; specularity outliers are rejected using RANSAC
- Number of Base Materials: optimise for $T \in \{1, 2, 3\}$, choose model with the smallest photometric error
- Specular BRDF: set each pixel to a uniform mix of the base materials
- Specular Base Materials: initialise specularity differently to diversify the output; roughness is set to 0.1 for all

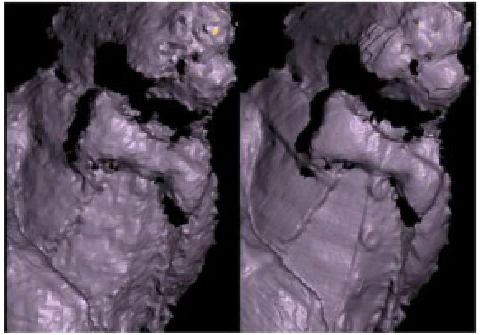


Results



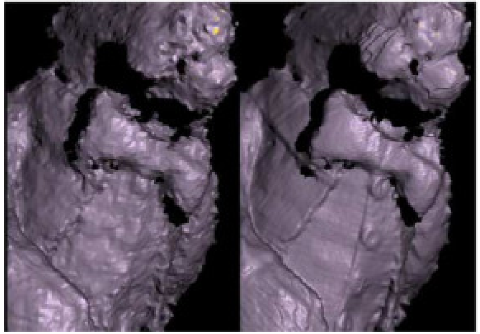
Results

- Slight misalignments of camera poses cause significant errors in geometry and especially specularity reconstruction



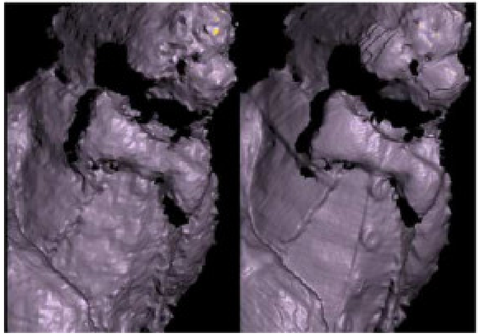
Results

- Slight misalignments of camera poses cause significant errors in geometry and especially specularity reconstruction
- Material Segmentation term is crucial, and works well



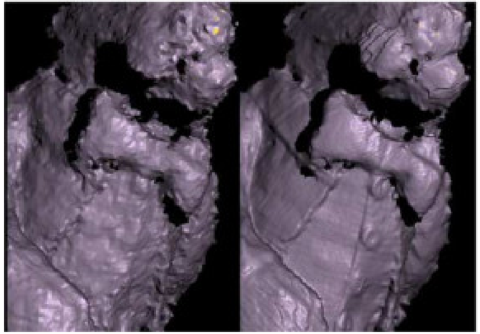
Results

- Slight misalignments of camera poses cause significant errors in geometry and especially specularity reconstruction
- Material Segmentation term is crucial, and works well
- Splitting normals and depth leads to better results



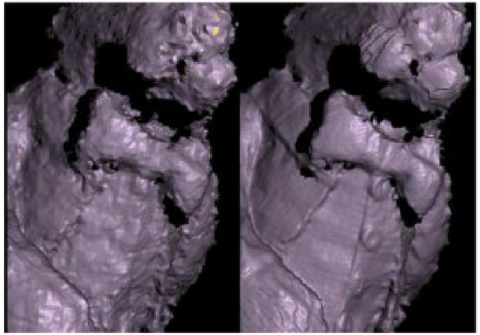
Results

- Slight misalignments of camera poses cause significant errors in geometry and especially specularity reconstruction
- Material Segmentation term is crucial, and works well
- Splitting normals and depth leads to better results
- The Methods degrades gracefully with fewer input images



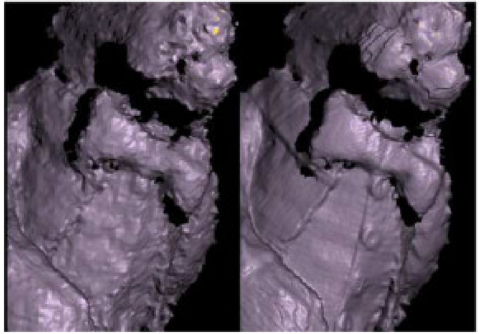
Results

- Slight misalignments of camera poses cause significant errors in geometry and especially specularity reconstruction
- Material Segmentation term is crucial, and works well
- Splitting normals and depth leads to better results
- The Methods degrades gracefully with fewer input images
- Results are robust against fewer input depth maps



Results

- Slight misalignments of camera poses cause significant errors in geometry and especially specularity reconstruction
- Material Segmentation term is crucial, and works well
- Splitting normals and depth leads to better results
- The Methods degrades gracefully with fewer input images
- Results are robust against fewer input depth maps
- Optimisation leads to super-resolution details



How to continue?

How to continue?

- Use a different geometry representation?

How to continue?

- Use a different geometry representation?
- Allow for more point lights?

How to continue?

- Use a different geometry representation?
- Allow for more point lights?
- Use SfM for initialisation instead of a depth sensor?

How to continue?

- Use a different geometry representation?
- Allow for more point lights?
- Use SfM for initialisation instead of a depth sensor?

Questions!