Garment retexturing using Kinect V2.0

Egils Avots

Supervisors:

Assoc. Prof. Gholamreza Anbarjafari

Assoc. Prof. Sergio Escalera

Outline

- Virtual fitting room project
- Kinect V2.0
- Infrared-based retexturing method
- 2D to 3D garment matching
- 3D model retexturing

Virtual fitting room

Mannequin [1]

Web application [2]

1. http://www.cross-innovation.eu/wp-content/uploads/2012/12/Fitsme1.jpg
2. https://tctechcrunch2011.files.wordpress.com/2015/07/screen-shot-2015-07-13-at-02-14-40.png

Existing procedure

- Select a garment
- Dress the mannequin
- Capture 100-280 robot shapes
- Image post processing
- Insert human model
- Include garment in the virtual fitting room application

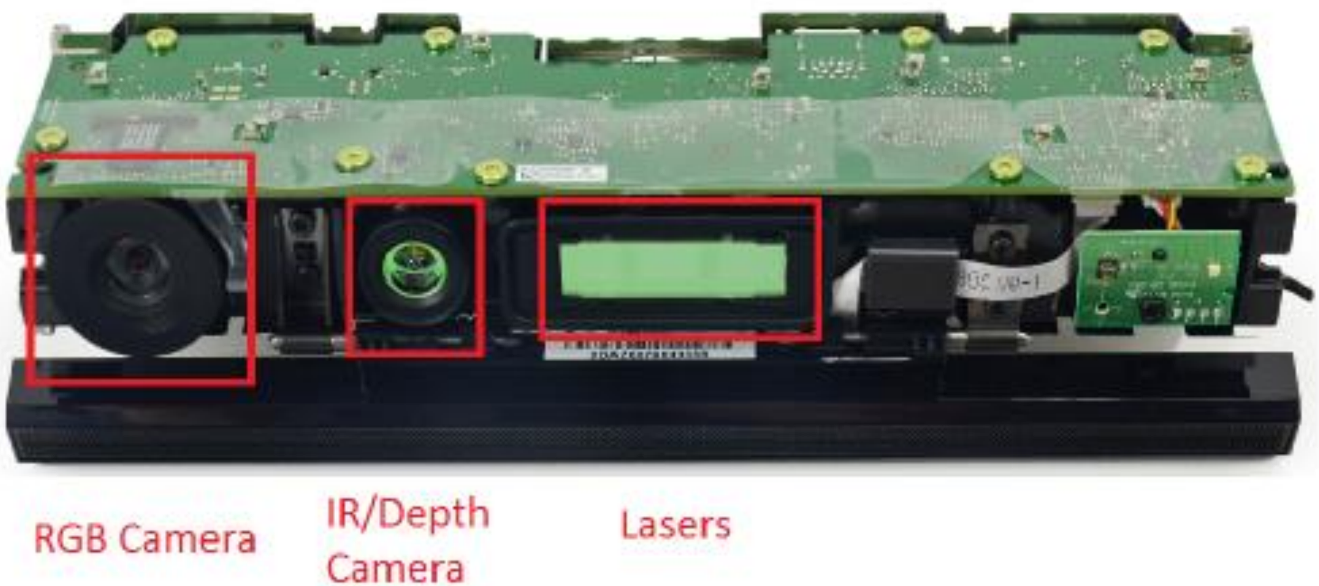
Problems

- Transportation and storage of garments is costly
- Manual and time consuming process

Kinect V2.0 specifications

Feature	Kinect 2
Color Camera	1920 x 1080 @30 fps
Depth Camera	512 x 424
Max Depth Distance	8 M
Min Depth Distance	50 cm
Depth Horizontal Field of View	70 degrees
Depth Vertical Field of View	60 degrees
Tilt Motor	no
Skeleton Joints Defined	25 joints
Full Skeletons Tracked	6
USB Standard	3.0
Supported OS	Win 8, Win 10
Price	$199

Kinect V2.0



Source: Valgma, Lembit. *3D reconstruction using Kinect v2 camera*. Diss. Tartu Ülikool, 2016.

Infrared-based retexturing method

The method consist of

- Segmentation
- Texture mapping
- Shading

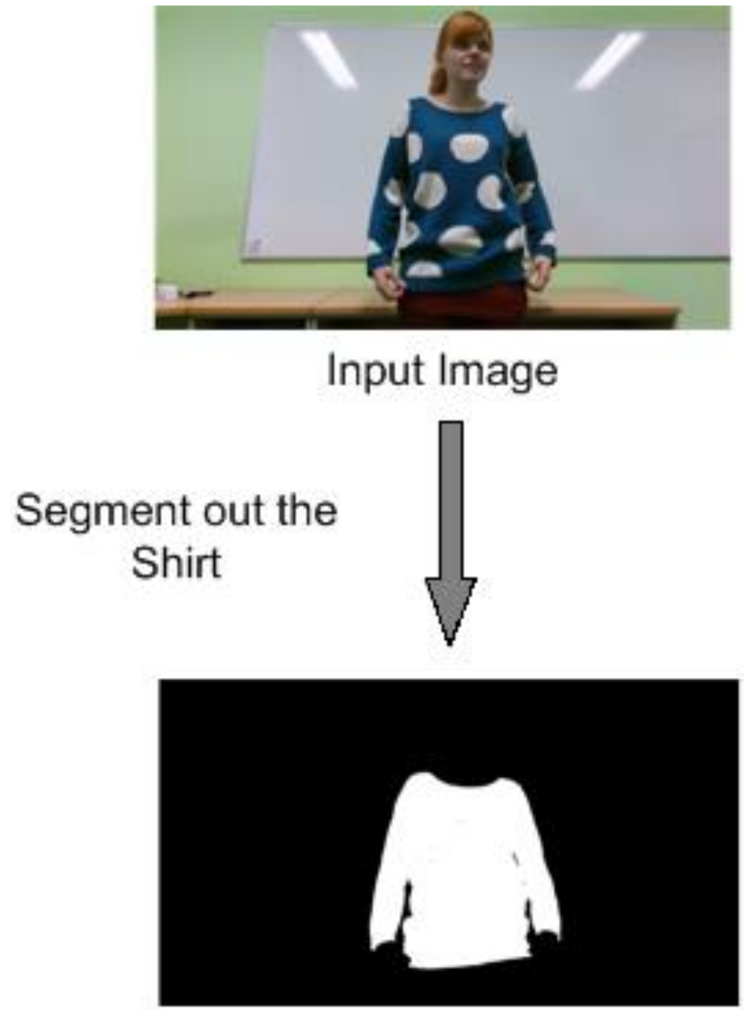
Assumptions

- No self-occlusions in segmented area
- The garment is made form a single fabric
- The input texture is considered as “ideal” texture

Egils Avots, Morteza Daneshmand, Andres Traumann, Sergio Escalera, and Gholamreza Anbarjafari. Automatic garment retexturing based on infrared information. Computers & Graphics, 59:28–38, 2016.

Segmentation

GrabCut, depth segmentation or other methods.



Texture mapping

Use Kinect V2.0 *color to depth mapping* and

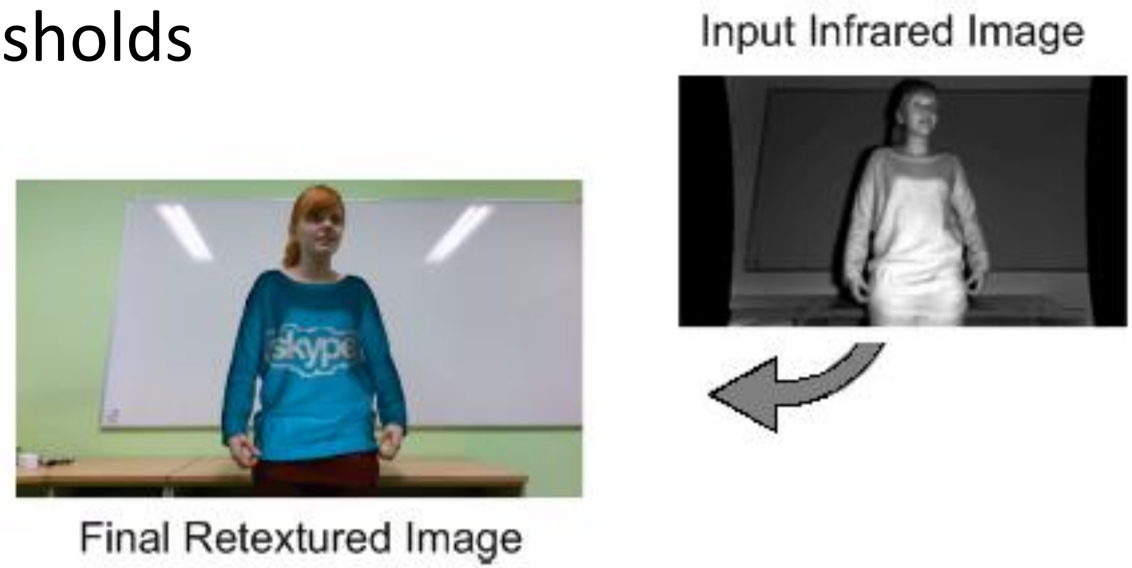
1. find *x, y, z* coordinates for the segmented region
2. normalize the found *x, y* coordinates (*x, y -> u, v*)
3. Replace Kinect FHD pixels with corresponding values from the texture image

Input Texture Image

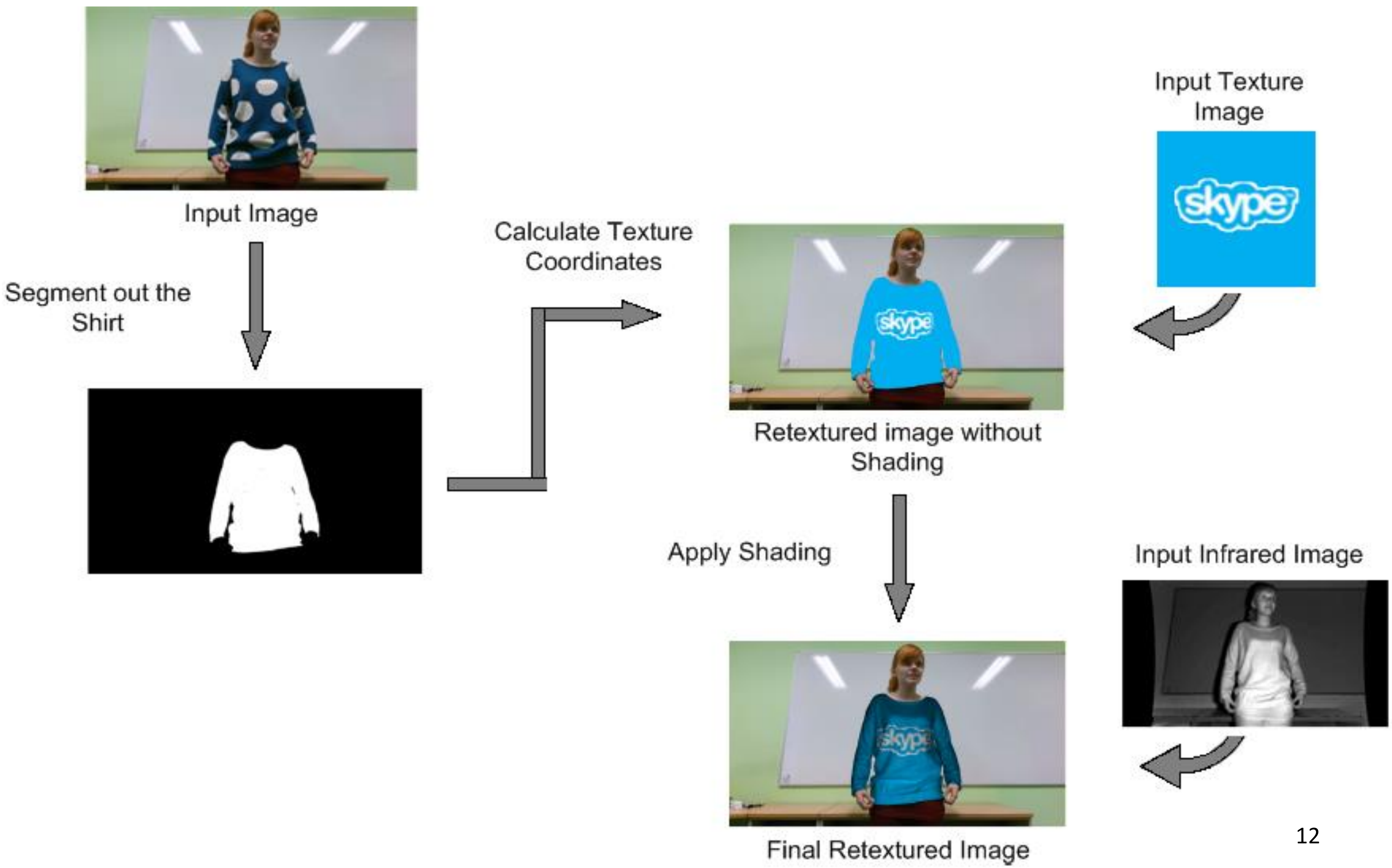
Retextured image without Shading

Pixel shading

- max and min IR values
- user defined thresholds



Retexturing flow chart



Method comparison

From left to right:

- IRT (proposed method)
- Color-mood-aware clothing retexturing [1]
- Image-based material editing [2]

Method	Mean Opinion Score
IRT	566 votes
Shen J. et al.	57 votes
Khan EA. et al.	177 votes

1. Shen J. et al. *Color-mood-aware clothing retexturing.* Computer-Aided Design and Computer Graphics, 2011
2. Khan EA. et al. *Image-based material editing.* ACM Transactions on Graphics (TOG) 2006

2D to 3D garment matching

The method consist of

- Segmentation
- Outer contour matching
- Inner contour matching
- Shading (based on IR)

Assumptions

- No self-occlusions in segmented area
- The garment is made form a single fabric
- The input texture is considered as “ideal” texture

Egils Avots, Meysam Madadi, Sergio Escalera, Jordi Gonzalez, Xavier Baro Sole, Gholamreza Anbarjafari. From 2D to 3D Geodesic-based Garment Matching: A Virtual Fitting Room Approach (***Undergoing revision in IET Computer Vision)***

Segmentation

Semi-automatic (RGB-D)

Real person

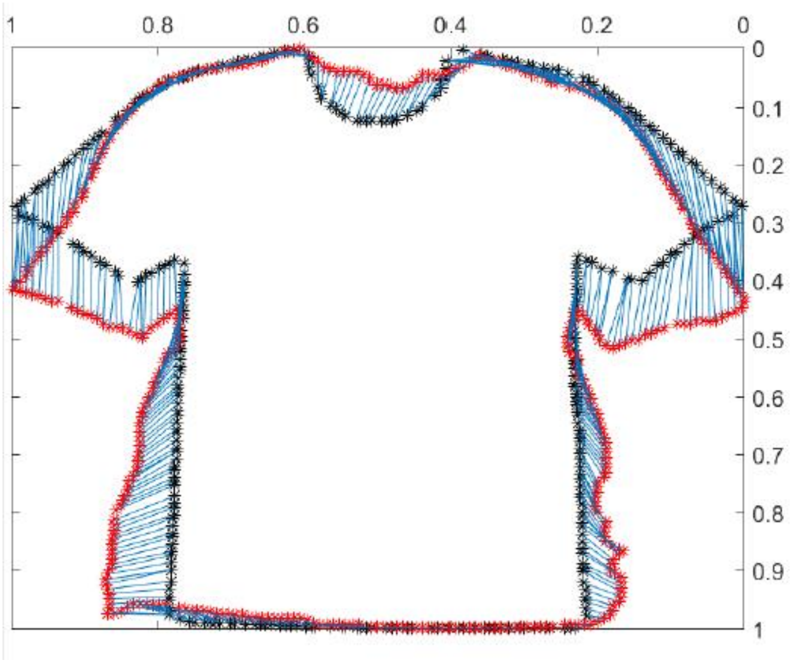
Automatic (RGB-D)

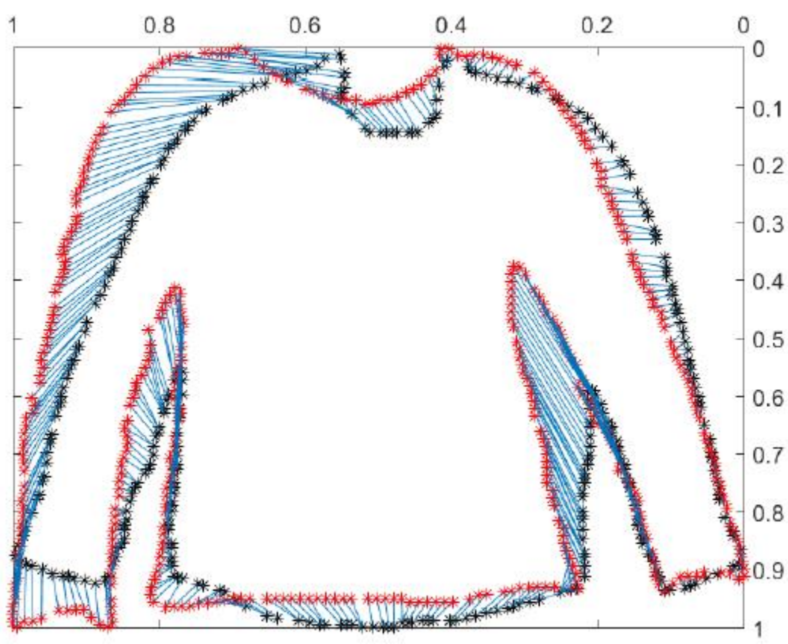
Real person

Semi-automatic (RGB)

Flat garment

Outer contour matching

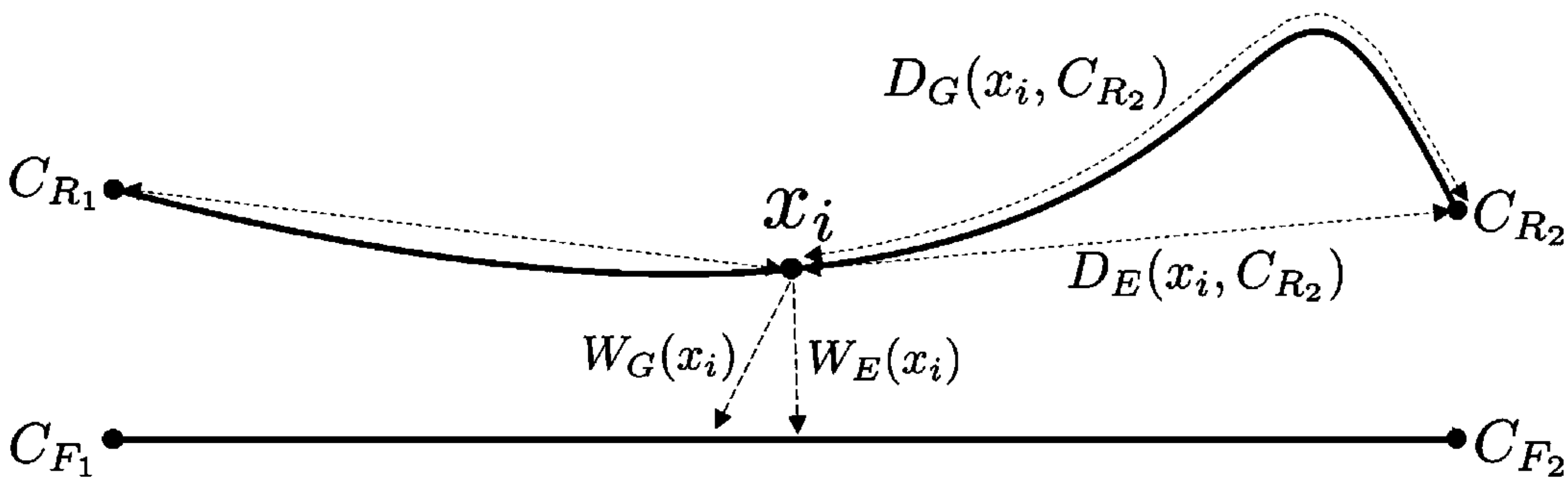




Red contour – *Real person*

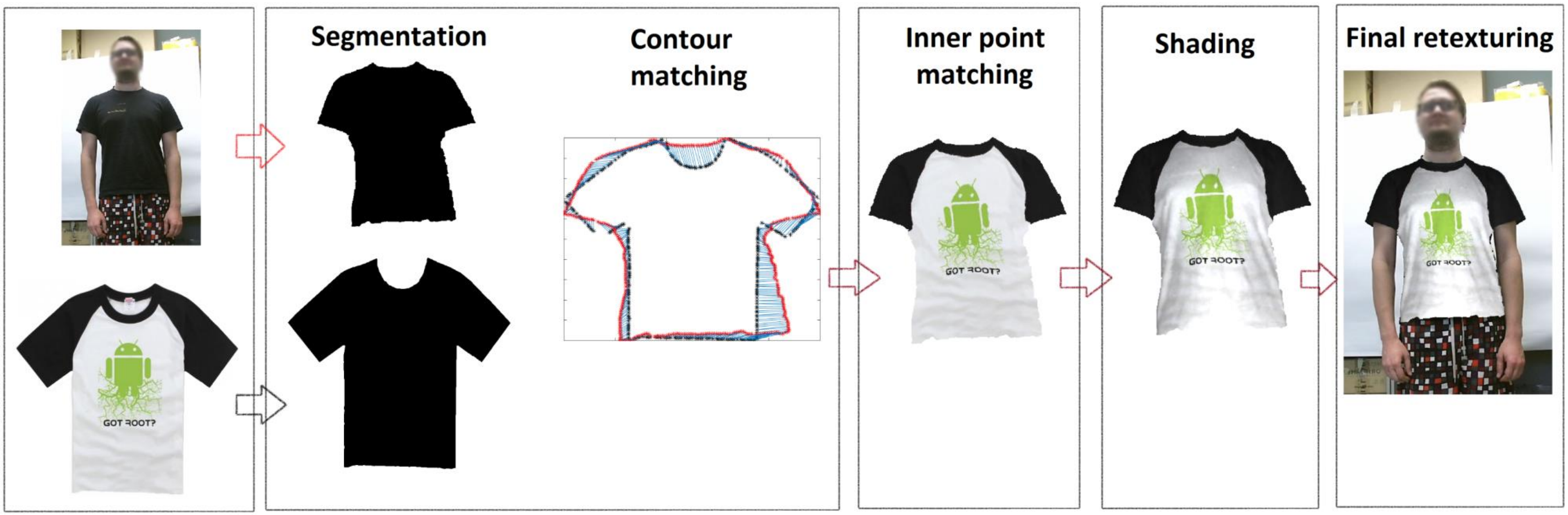
Black contour – *Flat garment*

Inner contour matching



C_R – contour of a real person
C_F – contour of a flat garment
W_E – mapping using Euclidian distance
W_G – mapping using Geodesic distance
D_E – Euclidian distance
D_G – Geodesic distance

2D to 3D retexturing flow chart



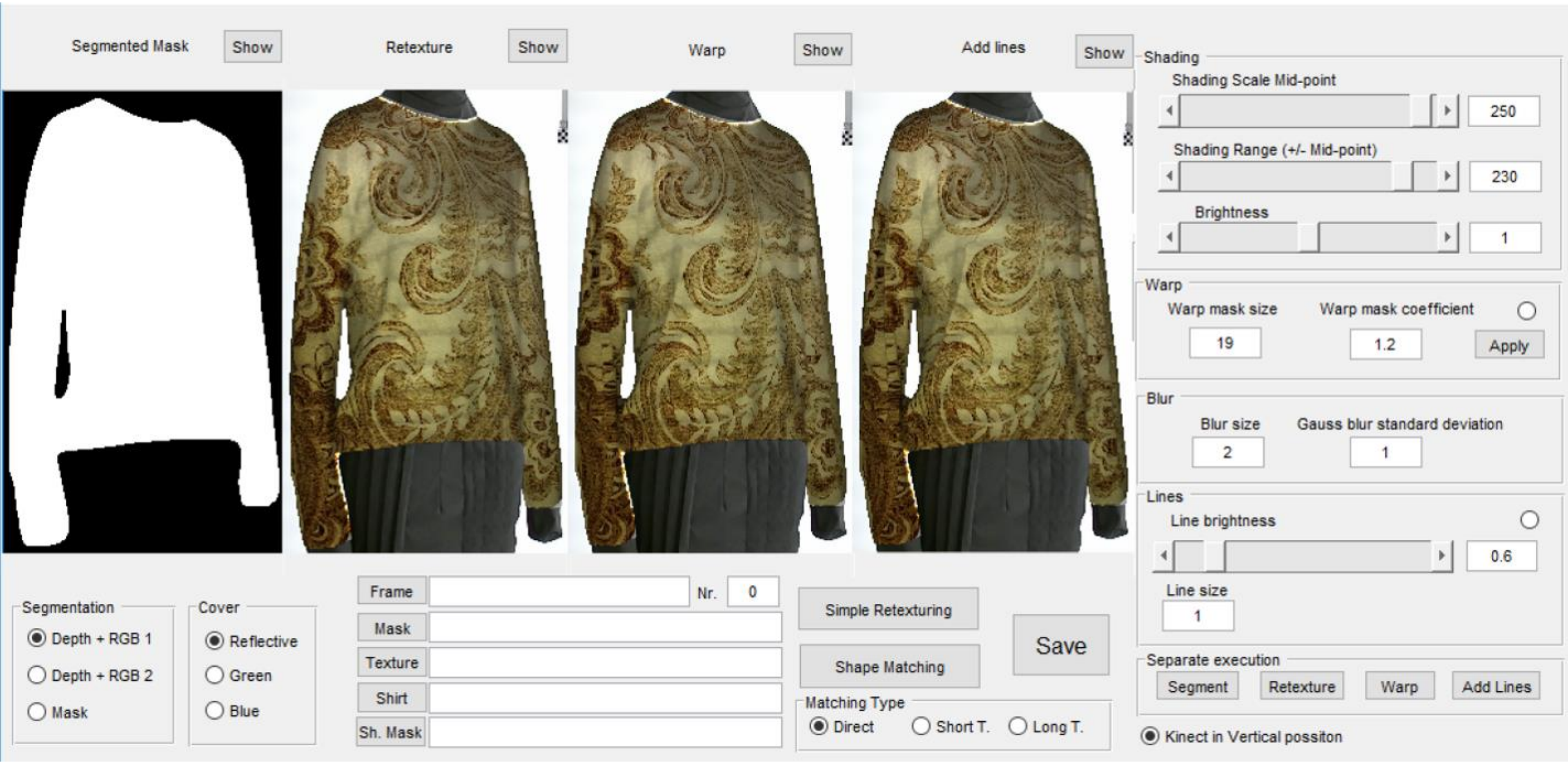
Evaluation - Mean Opinion Score

Method	T-shirt Votes	T-shirt %	Long sleeve Votes	Long sleeve %
NRICP	77	2.68%	32	3.69%
CPD	485	16.88%	245	28.23%
2D to 3D g.m.	2311	80.44%	591	68.09%

Evaluation - Marker mapping error

Method	MSE for T-shirts	MSE for Long sleeves
NRICP	115.400 px	215.349 px
CPD	83.850 px	190.618 px
2D to 3D g.m.	75.005 px	105.884 px

GUI for testing IRT and 2D to 3D shape matching

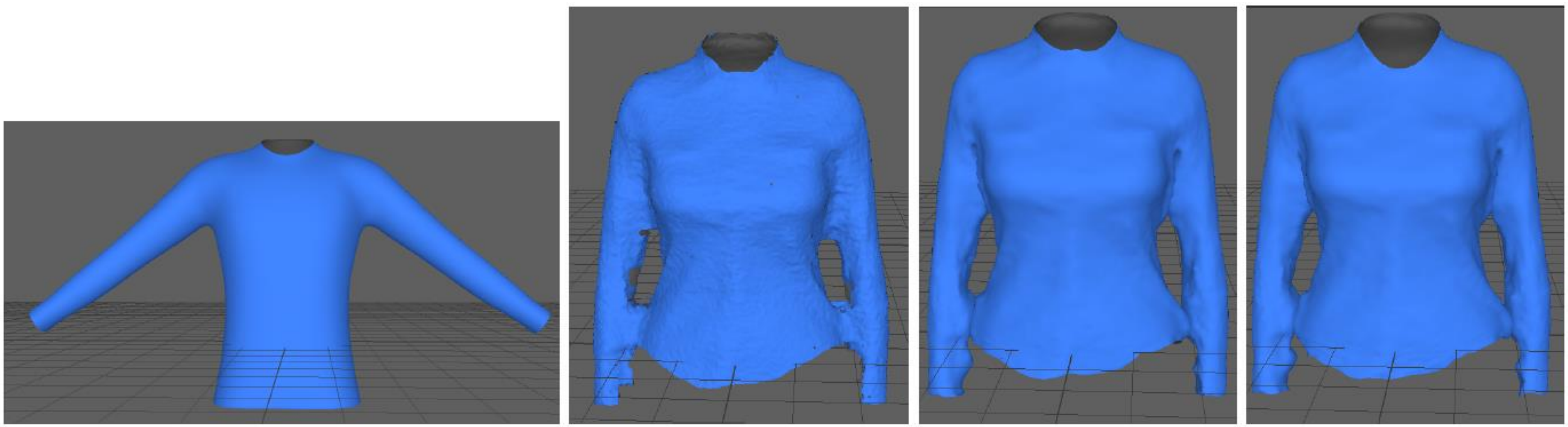


3D model creation using Kinect V2.0

The process of creating a 3D model:

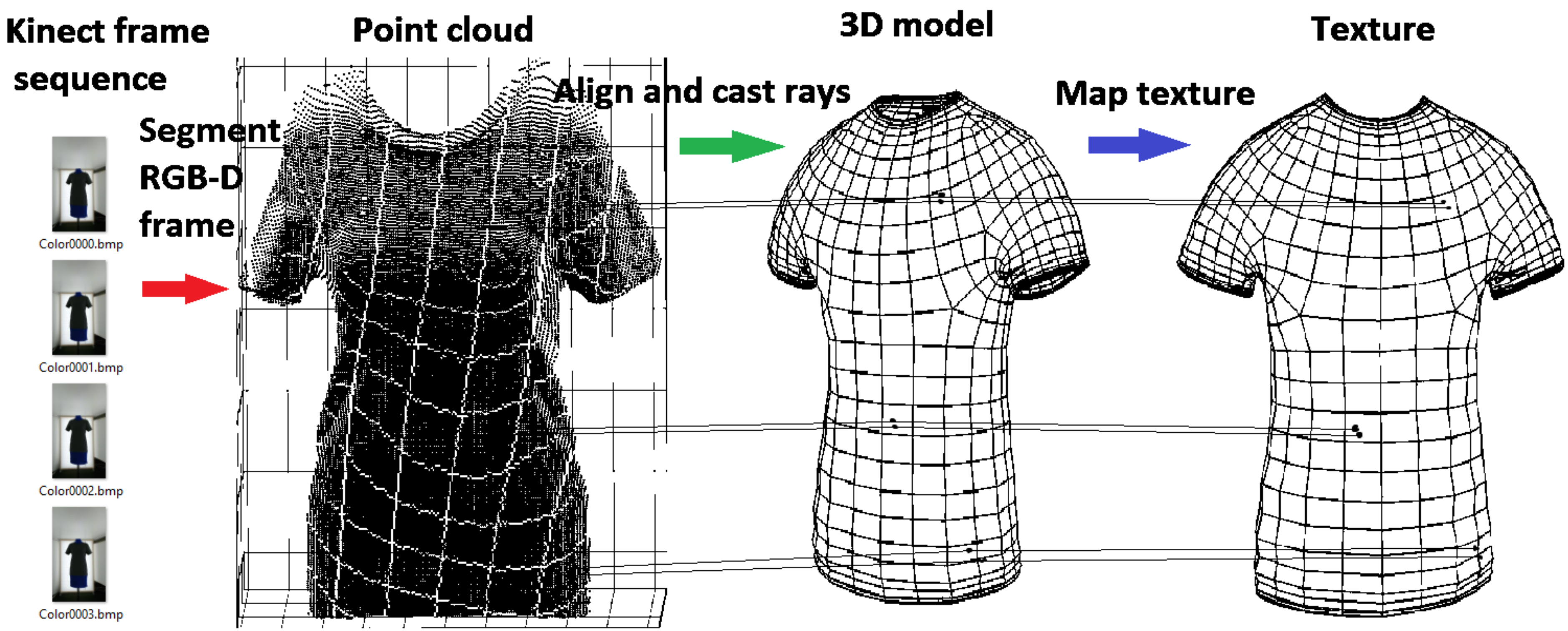
1. capture a sequence with Kinect
2. garment segmentation
3. align depth frames using ICP
4. correct errors using loop closure
5. denoise the point cloud
6. create a mesh from the point cloud

3D model wrapping



(a) Base model (b) Scanned garment (c) Wrapped model with reference points (d) Wrapped model without reference pts.

3D model retexturing process



Texture quality comparison

Thank you for attention!

Question 1

It seems that the evaluation of the proposed method in Section 3.4 is not as thorough as of its counterpart in Section 4.4. Moreover, it seems a bit difficult do draw conclusions from a few images presented in Figure 3.1.

What is the reason for less thorough evaluation of the method proposed in Chapter 3?

Are there any objective parameters that can be used in order to compare the performance of different methods?

Question 1 part 1

What is the reason for less thorough evaluation of the method proposed in Chapter 3?

Answer

While writing the article, the focus was placed on providing visually pleasing results, therefore MOS results were deemed sufficient for a publication.

Question 1 part 2

Are there any objective parameters that can be used in order to compare the performance of different methods?

Answer

Image similarity index

Feature tracking

Question 2

In Section 4.3.3, page 19, you talk about a set of coefficients ω. Later, in equation (4.3), ω is used as a matrix. Please explain:

- (a) How ω is defined?
- (b) How the coefficients in ω are computed? Are they computed as in equation (4.3)? - then why do you call them “trained”?

Question 2 part 1

- How ω is defined?

Radial Basis Function model

Each $(\mathbf{x}_n, y_n) \in \mathcal{D}$ influences $h(\mathbf{x})$ based on $\underbrace{\|\mathbf{x} - \mathbf{x}_n\|}_{\text{radial}}$

Standard form:

$$h(\mathbf{x}) = \sum_{n=1}^{N} w_n \underbrace{\exp\left(-\gamma\ \|\mathbf{x} - \mathbf{x}_n\|^2\right)}_{\text{basis function}}$$

The learning algorithm

Finding $w_1, \cdots, w_N$: $$h(\mathbf{x}) = \sum_{n=1}^{N} w_n \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}_n\|^2\right)$$

based on $\mathcal{D} = (\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$

$E_{\text{in}} = 0$: $\quad h(\mathbf{x}_n) = y_n$ for $n = 1, \cdots, N$:

$$\sum_{m=1}^{N} w_m \exp\left(-\gamma \|\mathbf{x}_n - \mathbf{x}_m\|^2\right)$$

The solution

$$\sum_{m=1}^{N} w_m \exp\left(-\gamma \|\mathbf{x}_n - \mathbf{x}_m\|^2\right) = y_n$$

N equations in N unknowns

$$\underbrace{\begin{bmatrix} \exp(-\gamma \|\mathbf{x}_1 - \mathbf{x}_1\|^2) & \dots & \exp(-\gamma \|\mathbf{x}_1 - \mathbf{x}_N\|^2) \\ \exp(-\gamma \|\mathbf{x}_2 - \mathbf{x}_1\|^2) & \dots & \exp(-\gamma \|\mathbf{x}_2 - \mathbf{x}_N\|^2) \\ \vdots & \vdots & \vdots \\ \exp(-\gamma \|\mathbf{x}_N - \mathbf{x}_1\|^2) & \dots & \exp(-\gamma \|\mathbf{x}_N - \mathbf{x}_N\|^2) \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}}_{\mathbf{w}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\mathbf{y}}$$

If Φ is invertible, $\boxed{\mathbf{w} = \Phi^{-1}\mathbf{y}}$

Source: https://www.youtube.com/watch?v=O8CfrnOPtLc&t=1443s

Question 2 part 2

- How the coefficients in ω are computed? Are they computed as in equation (4.3)?

Yes

- Why do you call them “trained”?

The (ω) weights are initially unknow variables that minimize error in training data.

Question 3

Please explain how the graph (p.19, line 18) for fast marching algorithm is constructed. What is geodesic distance, and why it is helpful to use it here?

Question 3 part 1

Please explain how the graph (p.19, line 18) for fast marching algorithm is constructed.

Pseudocode

Input parameters used

- *real_person_mask(HxW)*
- *real_person_depth_image (HxW)*
- *real_person_contour(160x2)*

Steps

1. *depth(HxW) <= real_person_depth_image.*real_person_mask*
2. *vertices(Nx3) <= get_world_cordinates(depth)*
3. *faces(Mx3) <=* traverse *depth* using *2x2 mask and register triangles*
4. *real_person_contour_index(160x1) <= find_faces_corresponding_to(real_person_contour)*
5. *Distance(Nx160) <= perform_fast_marching_mesh(vertices, faces, real_person_contour_index)*

Step 5 is performed using Matlab Toolbox Fast Marching [1] function which performs fast Fast Marching algorithm on a 3D mesh.

The distance is calculated for all *real_person_contour_index.*

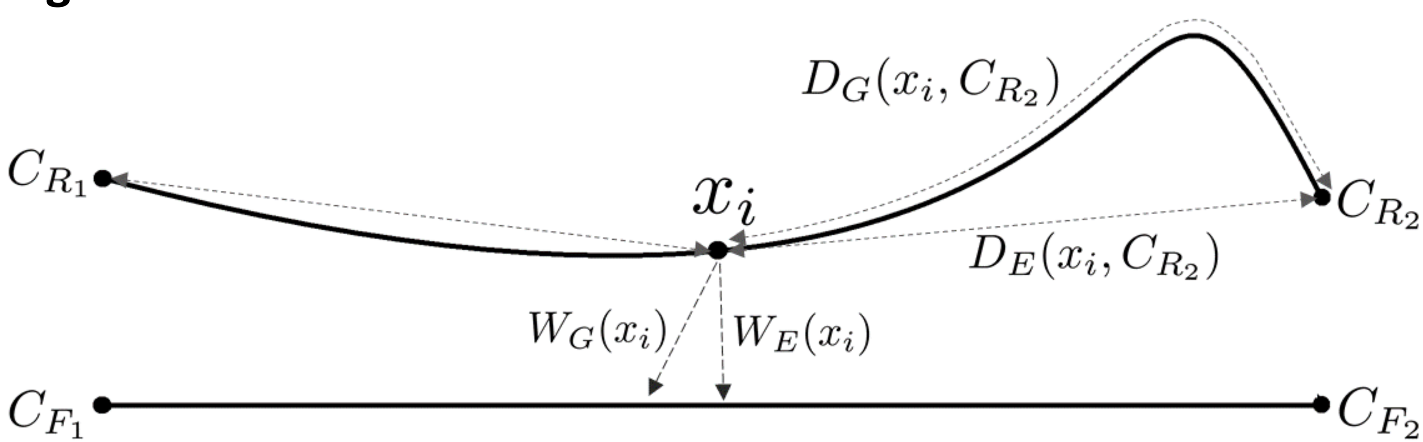
[1] - https://www.mathworks.com/matlabcentral/fileexchange/6110-toolbox-fast-marching

Question 3 part 2

What is geodesic distance, and why it is helpful to use it here?

Answer

A shortest path, or **geodesic path**, between two nodes in a graph is a path with the minimum number of edges. If the graph is weighted, it is a path with the minimum sum of edge weights. The length of a geodesic path is called **geodesic distance** or shortest distance.



Question 4

Please explain how the numbers in Table 4.1 were obtained.

(Did the voters have to choose the most realistic image among the alternatives shown to them?)

Answer

The MOS score was measured by showing 91 sets of images to 41 people.

Hello, we want your opinion!
Which of the A, B and C pictures look more realistic?

Image 1

Question 1

- ○ A
- ○ B
- ○ C