

Disparity Estimation for Depth Enhanced Stereo and Multi-View Format Conversion



Overview

- Generic Distribution Formats for 3D-TV
- Core Module for Stereo Matching: The HRM
- Cross-Bilateral-Filter and Variants
- Iterative Post-Processing
- Experimental Results
- Conclusions

Distribution Formats for 3D-TV

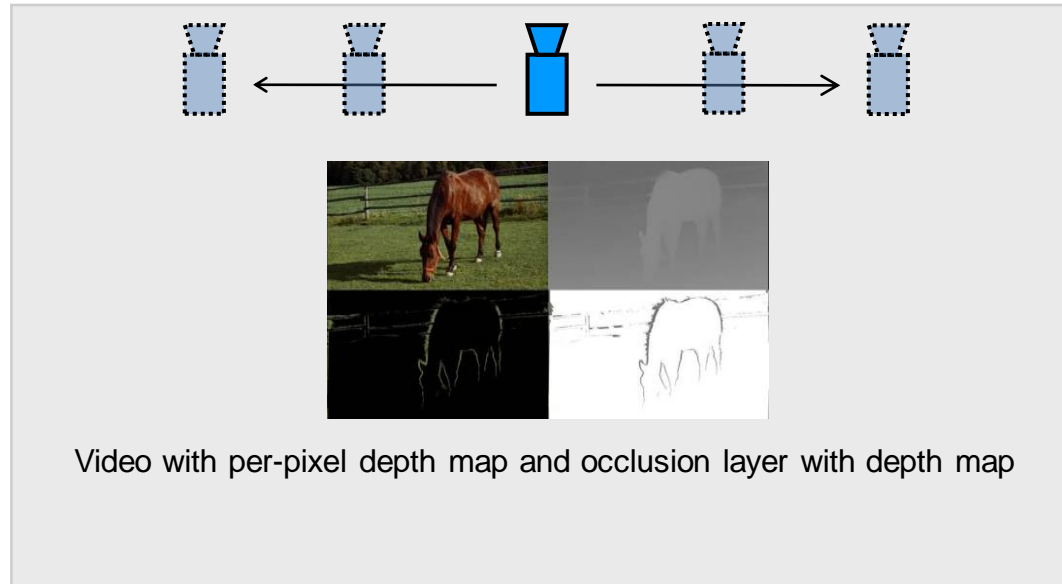
- Challenge: Format must be generic to cover whole range of today's and next-generation 3D displays
- Available formats : MPEG-C Part3 and MVC
- New formats based on Video + Depth :
 - LDV : **L**ayered **D**epth **V**ideo :
 - MVD: **M**ultiple **V**ideo+**D**epth
 - DES : **D**epth **E**nhanced **S**tereo

Objective of Depth-Based Formats

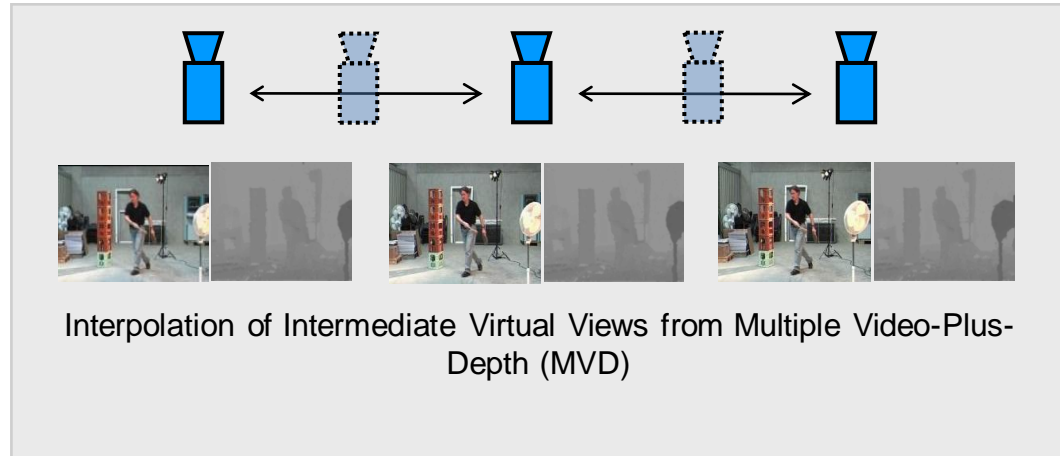
- Advantages of video +depth:
 - Efficient
 - Only a limited number of views are transmitted
 - Generic
 - Enables the synthesis of an arbitrary number of views
 - Flexible
 - Enables stereo-remastering (post-adaption of baseline), production of multiple masters for different screen sizes
 - Adaptation to user preferences and viewing conditons , depth-scaling, zero-parallax adjustment



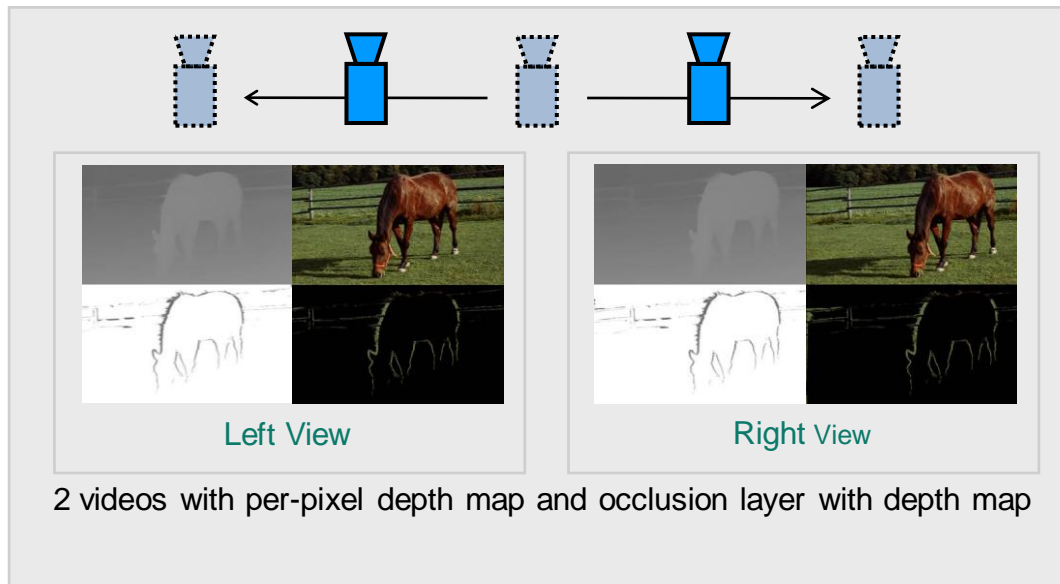
Distribution Formats: LDV



Distribution Formats: MVD



Distribution Formats: DES



Focus on Depth Enhanced Stereo

- Mainly based on traditional stereo shooting with two cameras
- Each view is enhanced by an additional depth-layer
- Optionally, each view may have an occlusion layer (useful for extrapolation of views outside the camera basis)
- Interoperable with ‚standard stereo‘
 - Original stereo production can be displayed unmodified if desired (important requirement of creative industry)



General Performance of Hybrid Recursive Matching

- Core Module for depth estimation
- Combination of recursive block matching and pixel recursion
- **Fast** disparity estimation by an efficient selection of a small number of candidate vectors
- Mainly **temporal and spatial consistent** depths
- Two estimations: From left to right and vice versa
- Left-Right-Consistency Check to detect occlusions and mismatches



Example of HRM Results

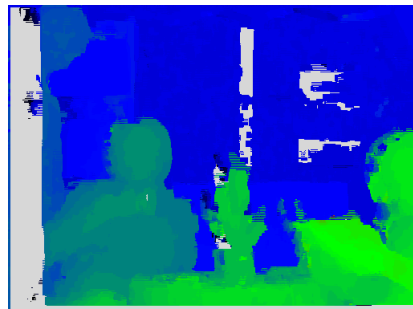
Left Camera



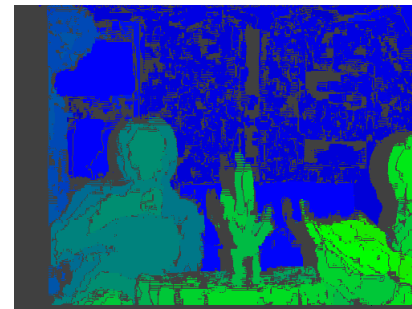
Right Camera



Original



Left-Right Consistent



Shortcomings of HRM Technique

- Problems:
 - Mismatches due to noise, homogenous areas, perspective distortions etc.
 - Foreground fattening
 - Occlusions
- Post-Processing is needed
 - Use Cross-Bilateral Filter



Bilateral Filter

- Bilateral Filter is an edge-preserving smoothing filter

$$g_s = \frac{\sum_{p \in n_s} w(p-s)c(f_p - f_s)f_p}{\sum_{p \in n_s} w(p-s)c(f_p - f_s)}$$

s = pixel location of pixel f_s

p = pixel location of pixel f_p

n_s = neighbourhood of f_s

$w(x)$ = spatial domain kernel

$c(x)$ = intensity domain kernel

with $w(x)$ and $c(x)$ typically Gaussians:

$$w(x) = e^{-\frac{1}{2}\left(\frac{|X|}{\sigma_s}\right)^2} \quad c(x) = e^{-\frac{1}{2}\left(\frac{|X|}{\sigma_c}\right)^2}$$



Cross Bilateral Filter

- Concept:
 - Compute filter weights using gray or color image
 - Apply filter kernel to depth image
- Application:
 - Filling holes
 - Aligning depth and intensity contours
 - Smoothing in homogeneous areas
- Variants



Variants of Cross Bilateral Filter

- Problem:
 - Undesired smoothing at object borders
 - Annoying artifacts during virtual view rendering
- Edge Preserving Solutions:
 - Cross bilateral median filter:
 - Use median of weighted depth list
 - Cross bilateral nearest depth:
 - Use the depth value closest to filter result



Example for Cross-Bilateral Filter Application

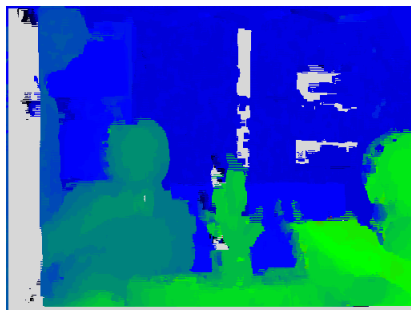
Left Camera



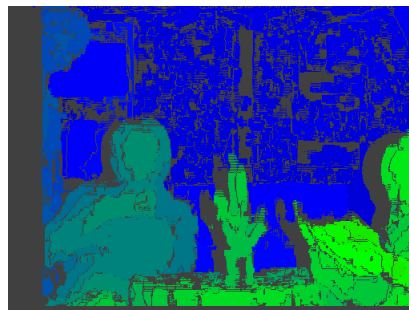
Right Camera



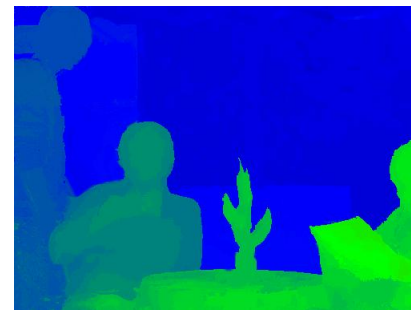
Original



Left-Right Consistent



Filtered

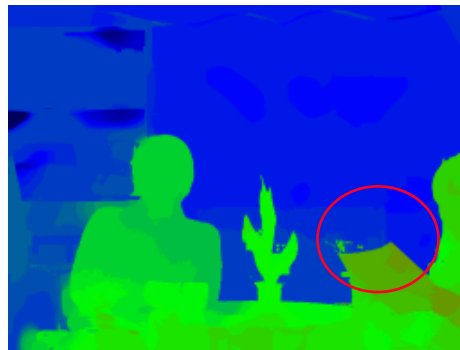
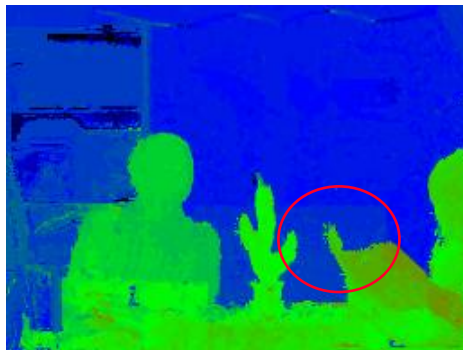


Problems of No-Adaptiv Cross-Bilateral Filtering



Before Filtering

After Filtering



Window-Radius=30

$$\sigma_s = 25$$

$$\sigma_c = 0.75$$



Adaptive cross-bilateral filters

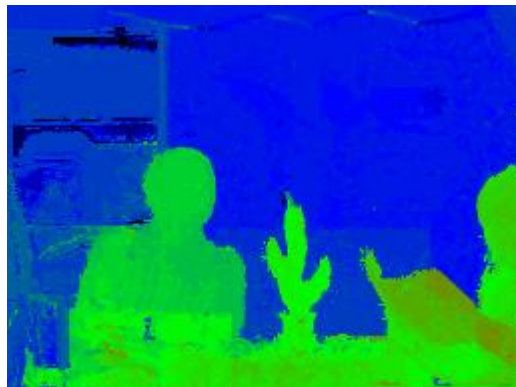
- Adapt the geometric spread σ_s to the local image structure
 - E.g. adapt σ_s to the edge strength of the pixel under consideration
 - Only slight smoothing at object borders
 - Strong smoothing in homogenous areas



Example Adaptive cross-bilateral filters

- Example: Use a small and a large σ_s depending on local color variance
- Window-Radius = 30 , $\sigma_c = 0.75$
- $\sigma_{s_small} = 12$, $\sigma_{s_large} = 25$

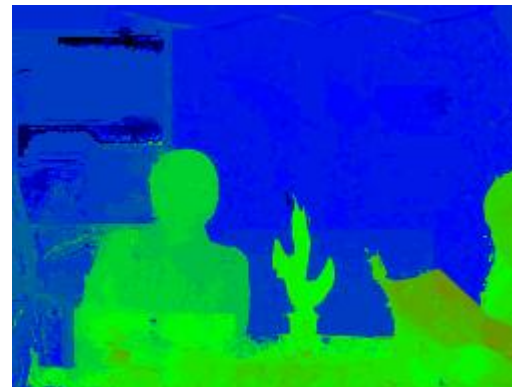
Input Depth Map



Mask for σ_{s_small}

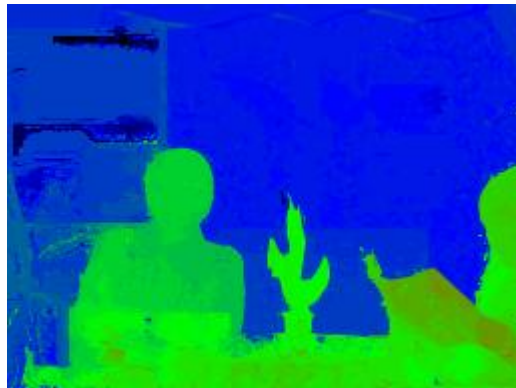


Result



Example Adaptive cross-bilateral filters

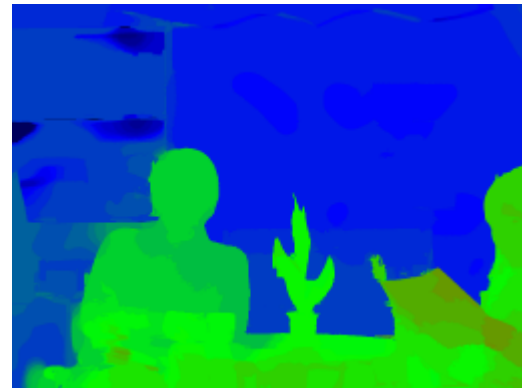
Input Depth Map



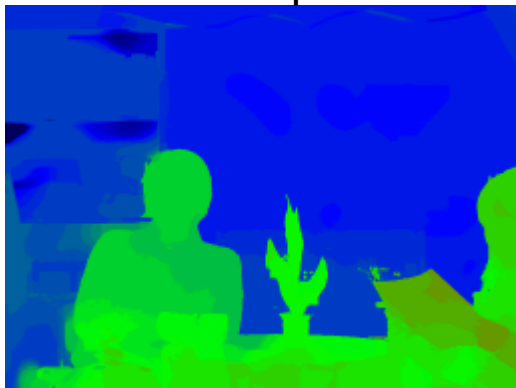
Mask for σ_{s_large}



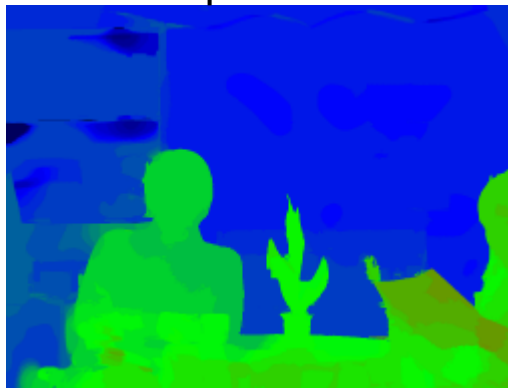
Result



Non-Adaptive



Adaptive



Adaptive cross-bilateral filters

- Adapt the filter weights to the local depth structure
 - E.g for Occlusion Filling:
 - Set the weights for foreground pixel to zero
 - Prevents from filling in foreground pixels
- Adapt the filter weights to the confidence of the depth estimates
 - Prevents from using unreliable depths during filtering



Recursive Filtering Stereo

- Developed by Faysal Bourhorbel
- Idea: calculate the depth recursively by weighted averaging the depths in the neighbourhood of the pixel

$$d_t(i) = \sum_{n \in N} w_{ij} d_{t-1}(n)$$

$d_t(i)$ = disparity of pixel i at iteration t

n = pixel in the neighbourhood N of pixel i

t = actual iteration



Recursive Filtering Stereo

- Weights for the bilateral filter combine the color similarity and matching confidence of the pixel in the left and right image:

$$w_{ij} = \exp(-\alpha |c_i^1 - c_j^1| - \beta |c_j^1 - c_{j'}^2|)$$

- with: c_i^1 and c_j^1 color values of left image at pixels i and j
 $c_{j'}^2$ is the disparity-compensated color, the color at pixel j'
which is at position $d_{t-1}(j)$



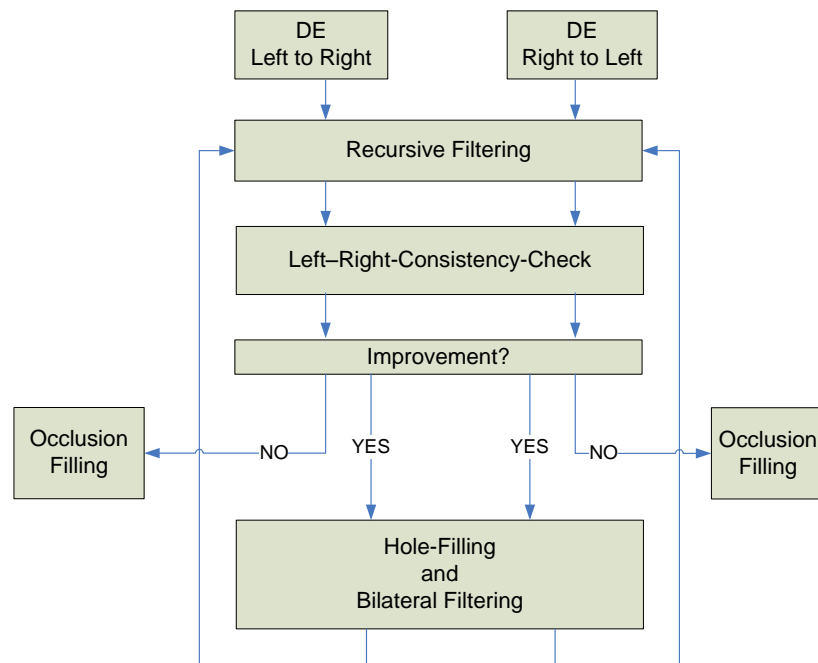
Recursive Filtering Stereo

- $|c_i^1 - c_j^1|$ penalizes the difference in color between the pixels in left image so that the weights for pixels with a different color from the center pixel are low -> averaging pixels with same color -> smooth objects
- $|c_j^1 - c_j^2|$ penalizes pixel matches with different colors
- Original version:
 - Do small random updates to ensure that the depths do not get stucked in a local minimum

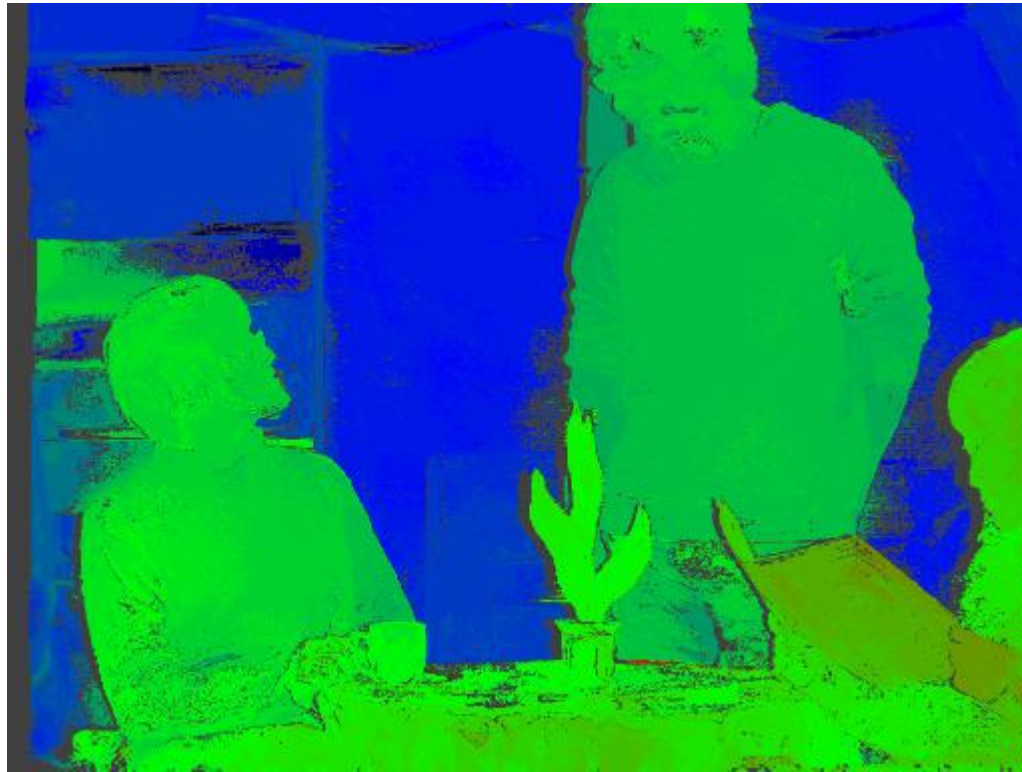


Iterative Post-Processing

- Principle:
 - Do iterative „Recursive Filtering“ and Left-Right-Consistency Check
 - Fill the holes
 - Update results by ‚normal‘ cross-bilateral filter



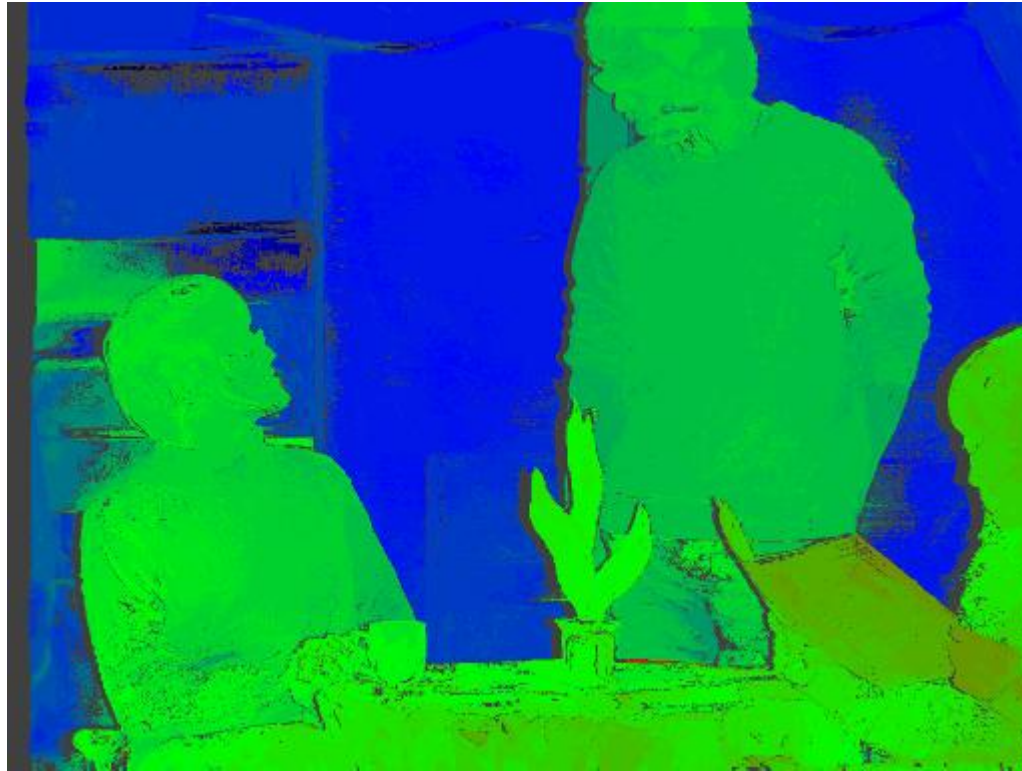
Example Iterative Post-Processing



| Iteration | Matches |
|-----------|---------|
| 1 | 673409 |
| 2 | 702785 |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |



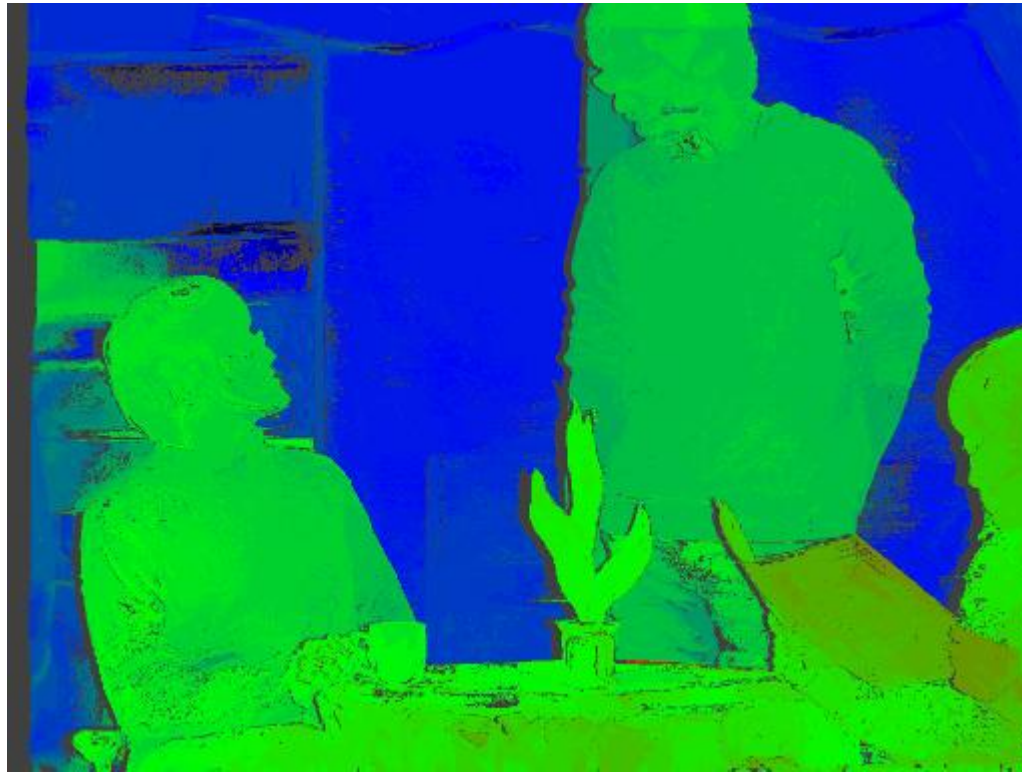
Example Iterative Post-Processing



| Iteration | Matches |
|-----------|---------|
| 1 | 673409 |
| 2 | 702785 |
| 3 | 715166 |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |



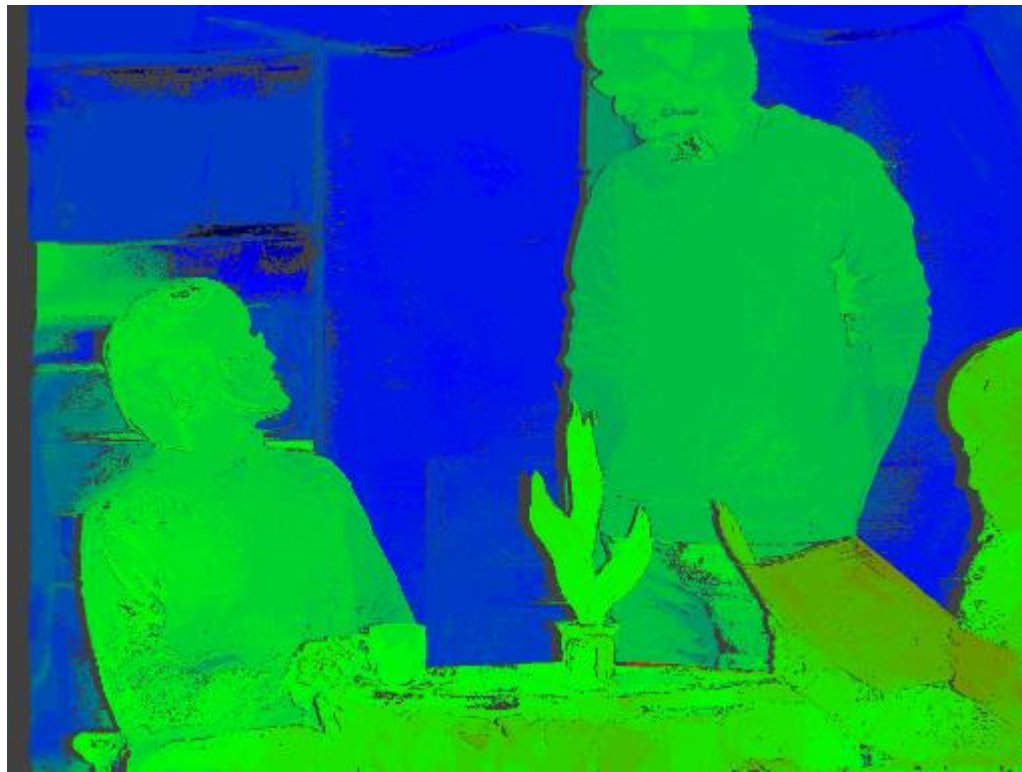
Example Iterative Post-Processing



| Iteration | Matches |
|-----------|---------|
| 1 | 673409 |
| 2 | 702785 |
| 3 | 715166 |
| 4 | 718839 |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |



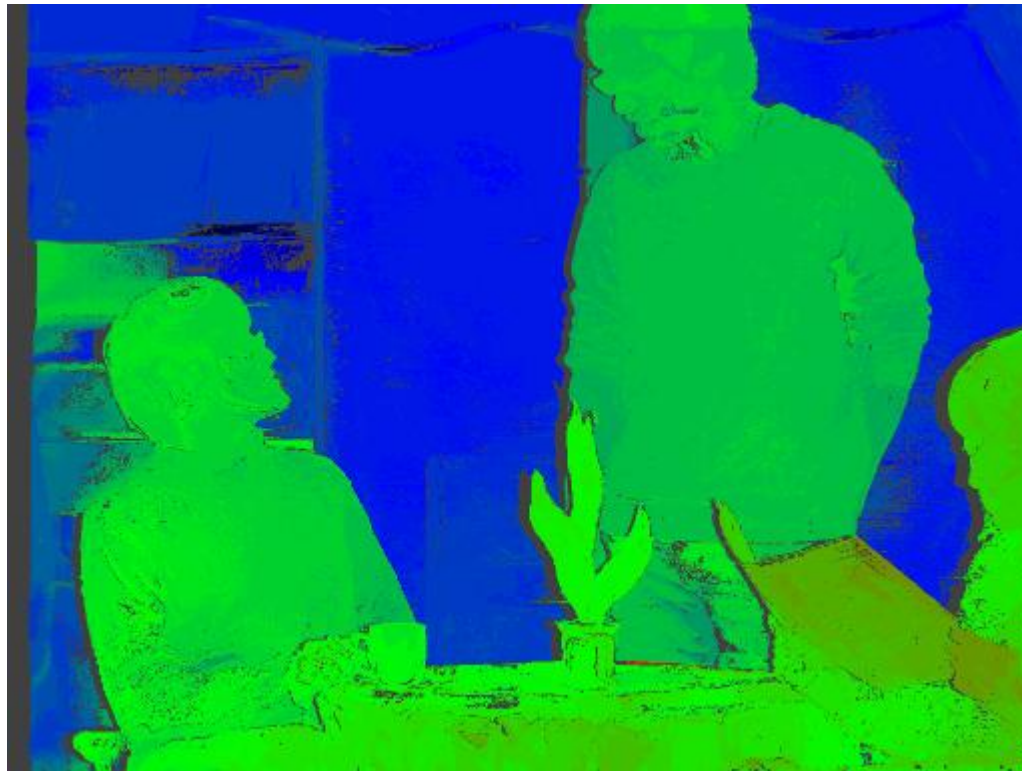
Example Iterative Post-Processing



| Iteration | Matches |
|-----------|---------|
| 1 | 673409 |
| 2 | 702785 |
| 3 | 715166 |
| 4 | 718839 |
| 5 | 721323 |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |



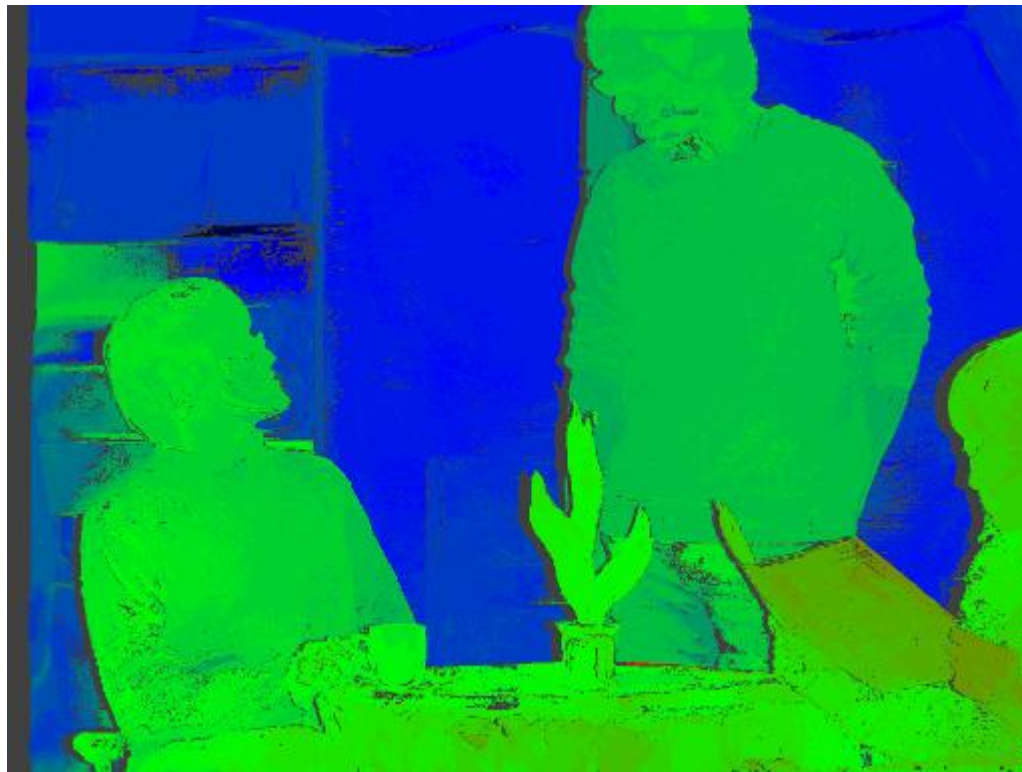
Example Iterative Post-Processing



| Iteration | Matches |
|-----------|---------|
| 1 | 673409 |
| 2 | 702785 |
| 3 | 715166 |
| 4 | 718839 |
| 5 | 721323 |
| 6 | 722950 |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |



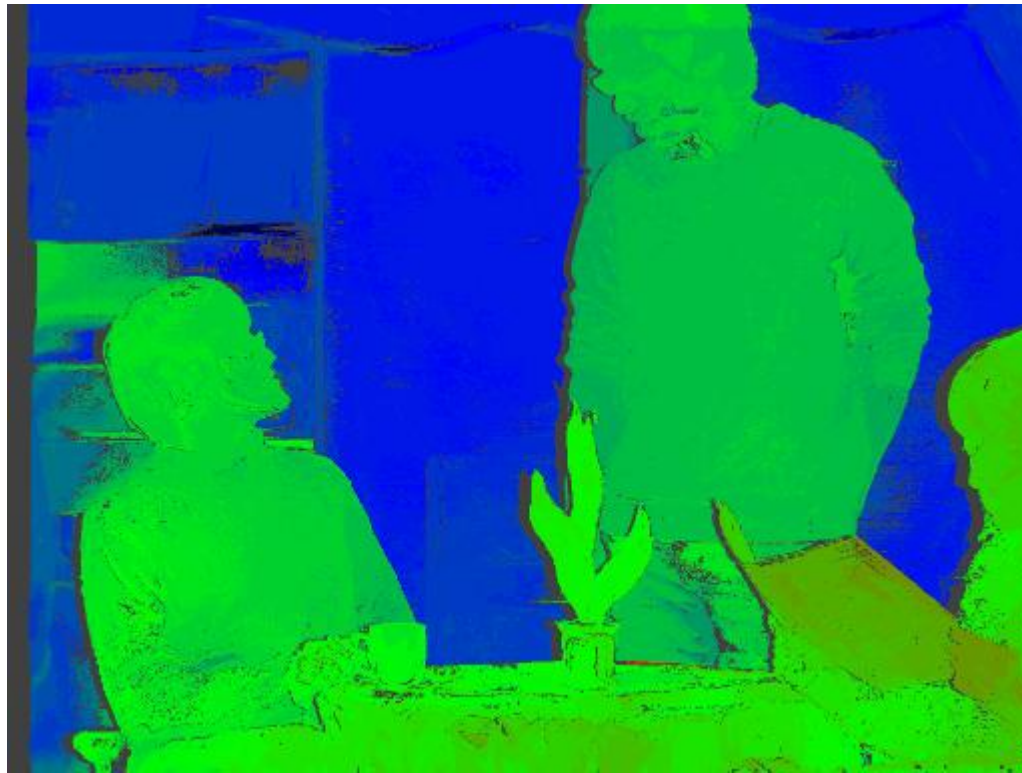
Example Iterative Post-Processing



| Iteration | Matches |
|-----------|---------|
| 1 | 673409 |
| 2 | 702785 |
| 3 | 715166 |
| 4 | 718839 |
| 5 | 721323 |
| 6 | 722950 |
| 7 | 724232 |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |



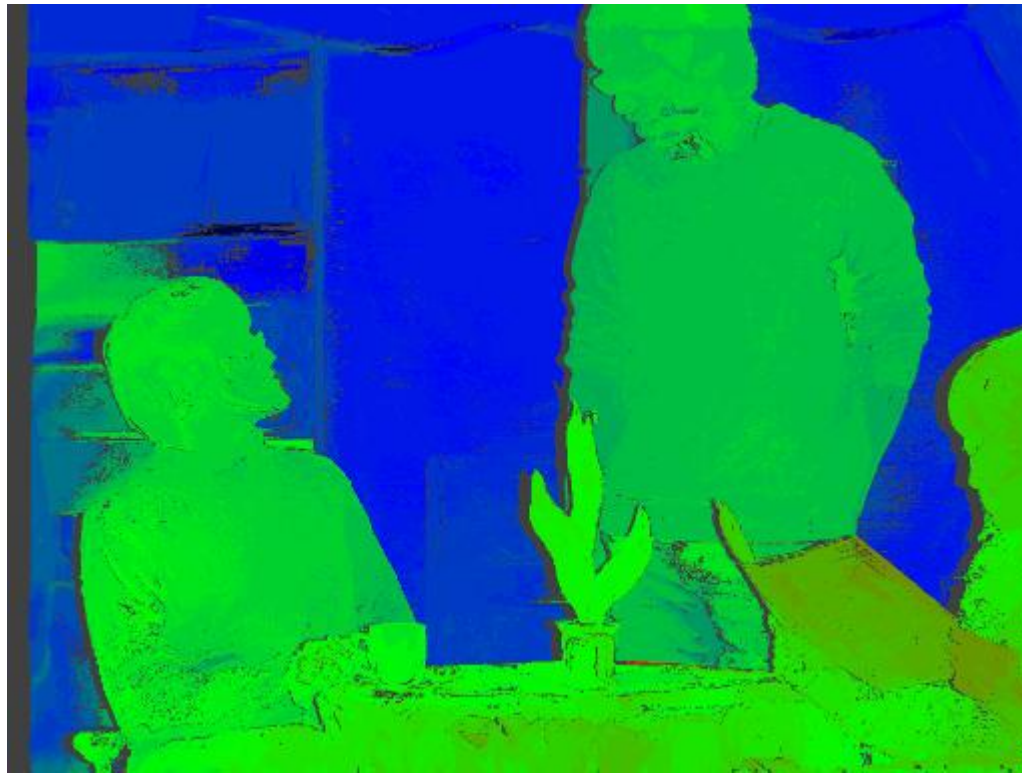
Example Iterative Post-Processing



| Iteration | Matches |
|-----------|---------|
| 1 | 673409 |
| 2 | 702785 |
| 3 | 715166 |
| 4 | 718839 |
| 5 | 721323 |
| 6 | 722950 |
| 7 | 724232 |
| 8 | 724721 |
| | |
| | |
| | |
| | |
| | |
| | |
| | |



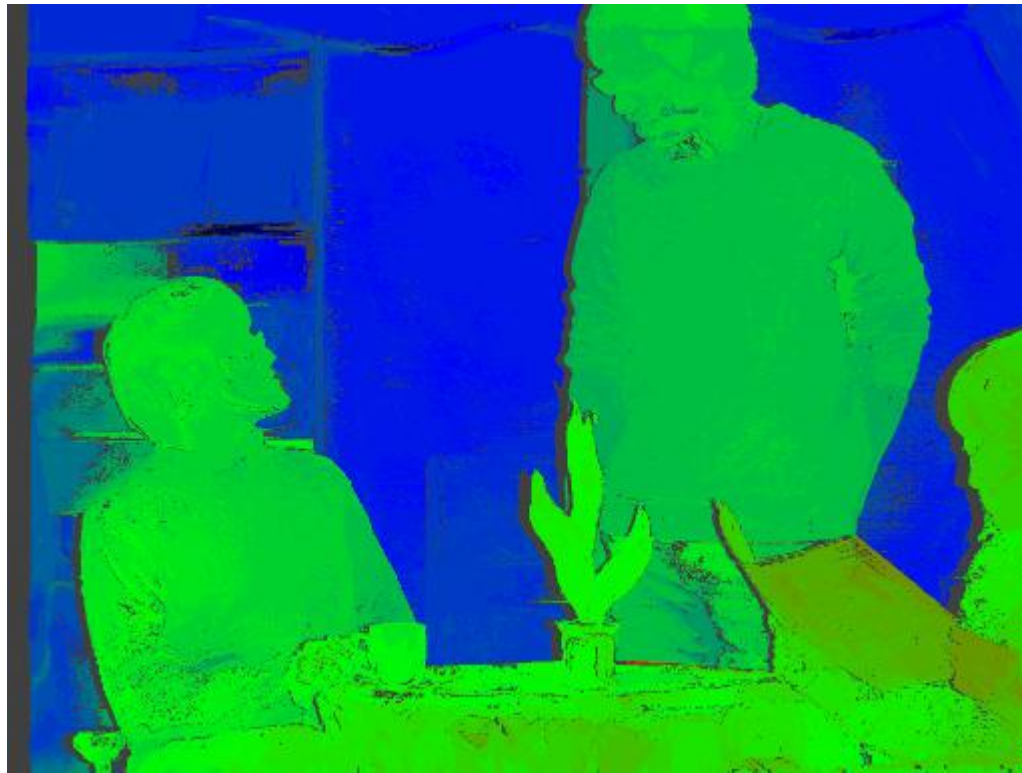
Example Iterative Post-Processing



| Iteration | Matches |
|-----------|---------|
| 1 | 673409 |
| 2 | 702785 |
| 3 | 715166 |
| 4 | 718839 |
| 5 | 721323 |
| 6 | 722950 |
| 7 | 724232 |
| 8 | 724721 |
| 9 | 726105 |
| | |
| | |
| | |
| | |
| | |



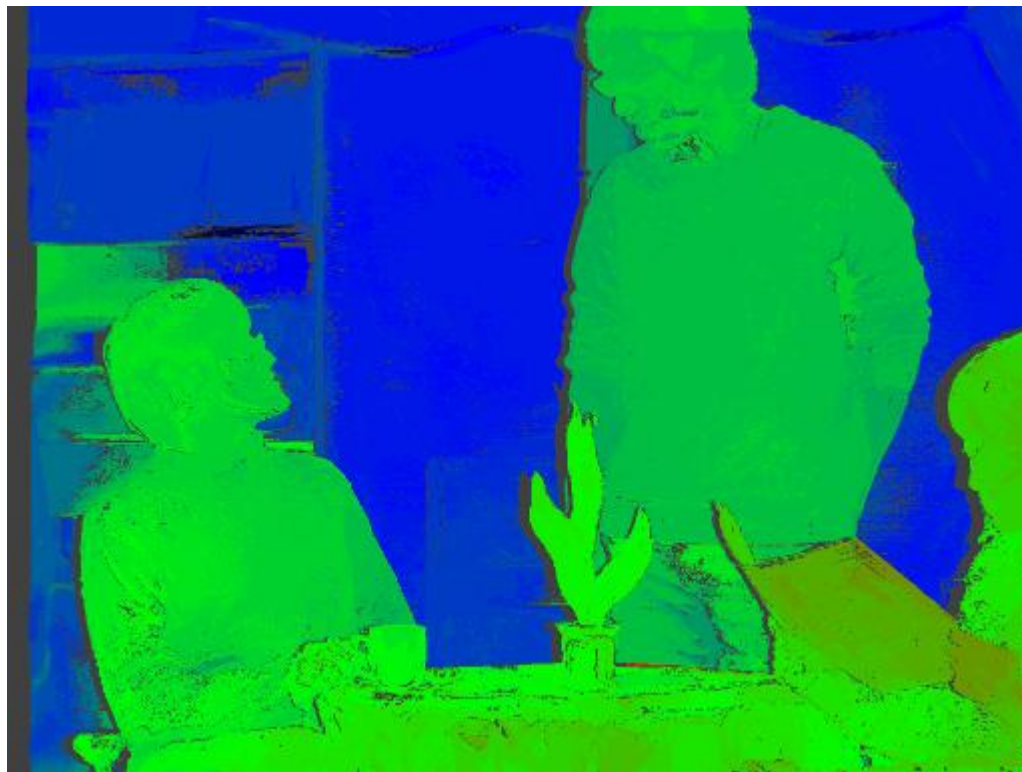
Example Iterative Post-Processing



| Iteration | Matches |
|-----------|---------|
| 1 | 673409 |
| 2 | 702785 |
| 3 | 715166 |
| 4 | 718839 |
| 5 | 721323 |
| 6 | 722950 |
| 7 | 724232 |
| 8 | 724721 |
| 9 | 726105 |
| 10 | 726497 |
| | |
| | |
| | |
| | |



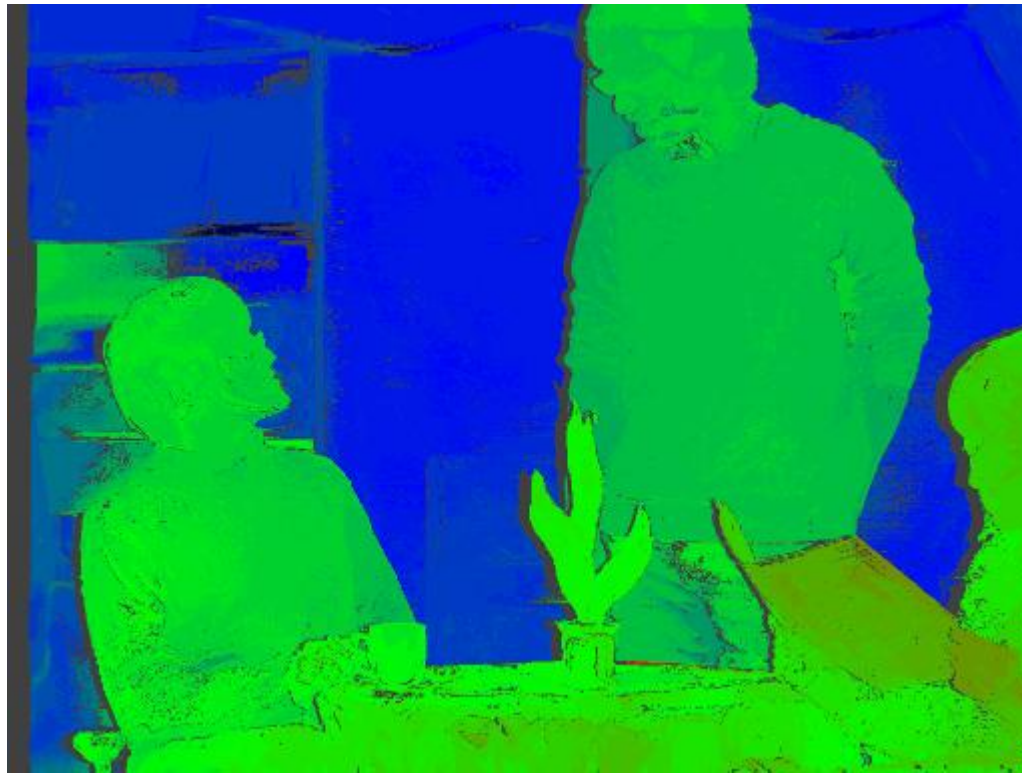
Example Iterative Post-Processing



| Iteration | Matches |
|-----------|---------|
| 1 | 673409 |
| 2 | 702785 |
| 3 | 715166 |
| 4 | 718839 |
| 5 | 721323 |
| 6 | 722950 |
| 7 | 724232 |
| 8 | 724721 |
| 9 | 726105 |
| 10 | 726497 |
| 11 | 726898 |
| | |
| | |
| | |



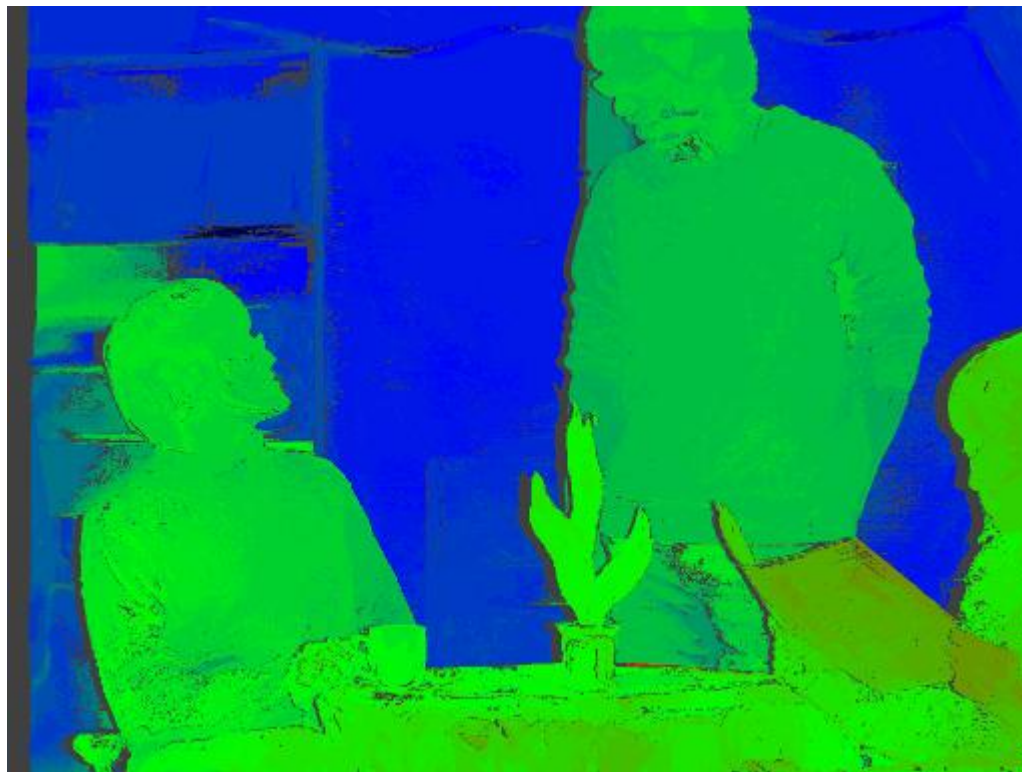
Example Iterative Post-Processing



| Iteration | Matches |
|-----------|---------|
| 1 | 673409 |
| 2 | 702785 |
| 3 | 715166 |
| 4 | 718839 |
| 5 | 721323 |
| 6 | 722950 |
| 7 | 724232 |
| 8 | 724721 |
| 9 | 726105 |
| 10 | 726497 |
| 11 | 726898 |
| 12 | 726776 |
| | |
| | |



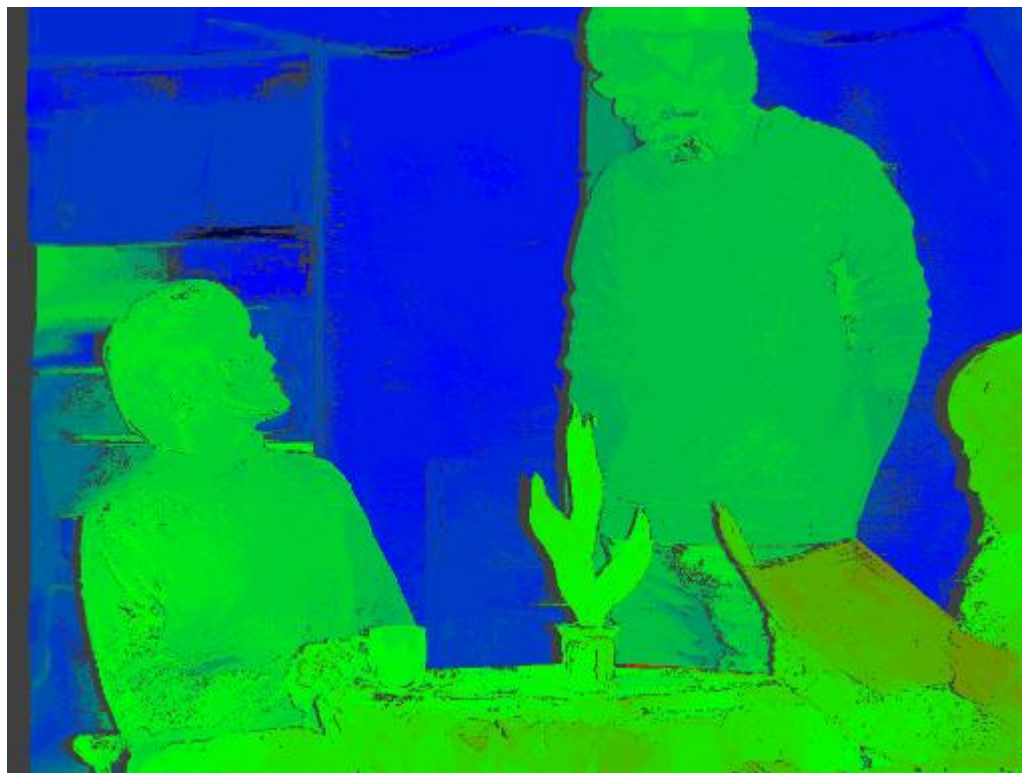
Example Iterative Post-Processing



| Iteration | Matches |
|-----------|---------|
| 1 | 673409 |
| 2 | 702785 |
| 3 | 715166 |
| 4 | 718839 |
| 5 | 721323 |
| 6 | 722950 |
| 7 | 724232 |
| 8 | 724721 |
| 9 | 726105 |
| 10 | 726497 |
| 11 | 726898 |
| 12 | 726776 |
| 13 | 727296 |
| | |



Example Iterative Post-Processing

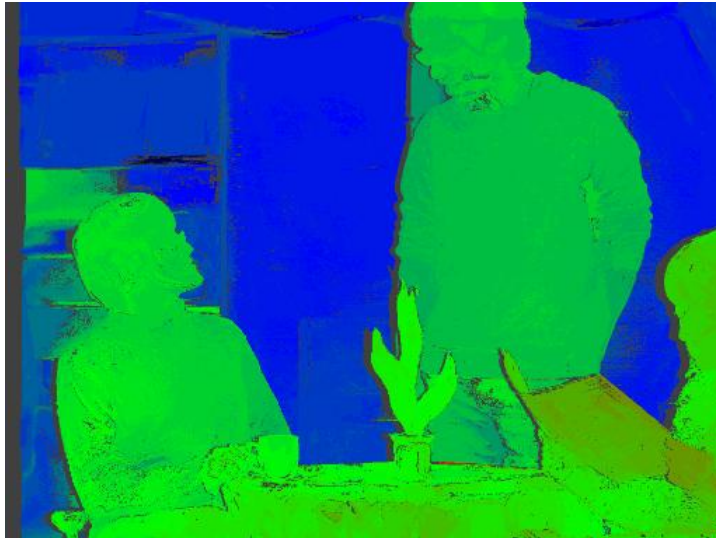


| Iteration | Matches |
|-----------|---------|
| 1 | 673409 |
| 2 | 702785 |
| 3 | 715166 |
| 4 | 718839 |
| 5 | 721323 |
| 6 | 722950 |
| 7 | 724232 |
| 8 | 724721 |
| 9 | 726105 |
| 10 | 726497 |
| 11 | 726898 |
| 12 | 726776 |
| 13 | 727296 |
| 14 | 727537 |

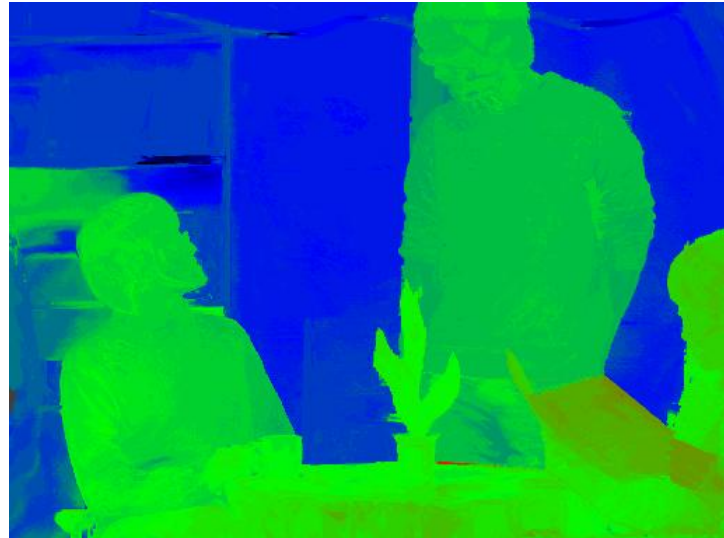


Example Iterative Post-Processing

Iteration 14



Final Result

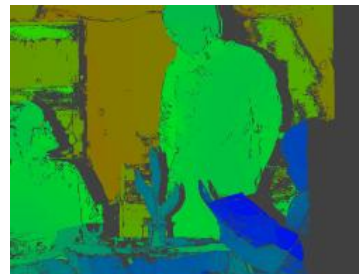
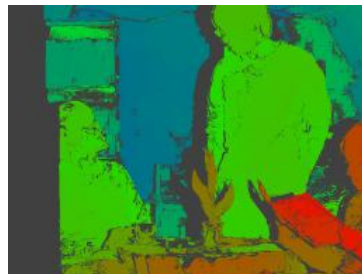


Example Iterative Post-Processing

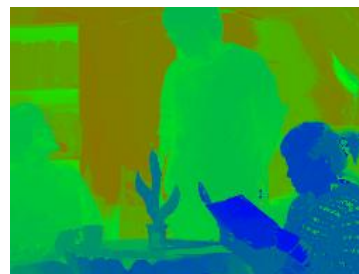
- Left and Right View



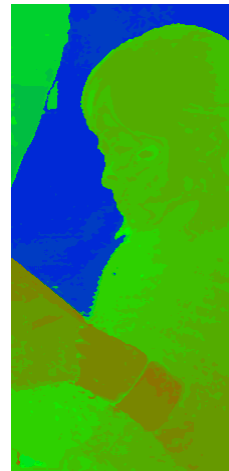
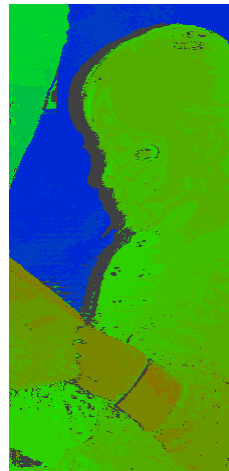
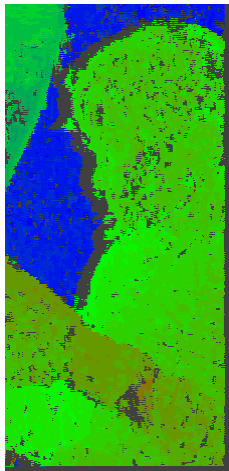
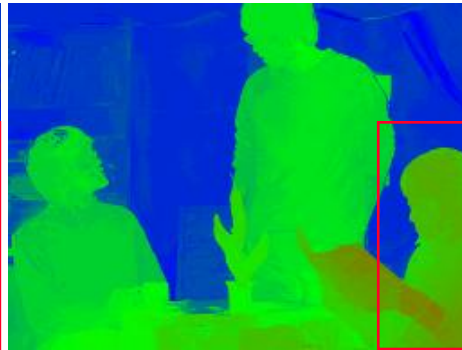
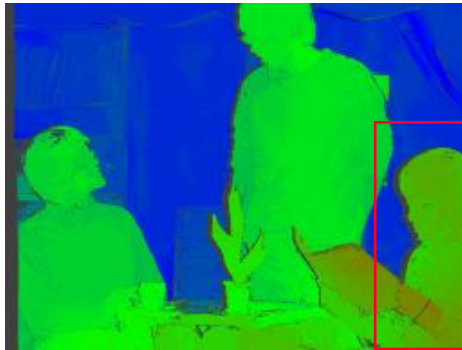
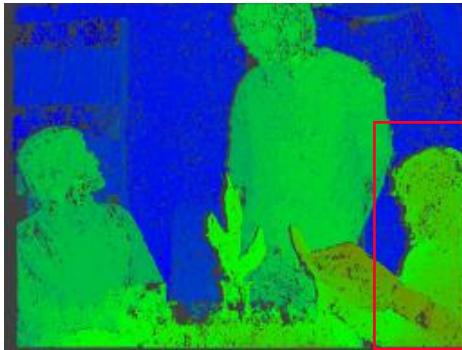
- Consistent Disparities



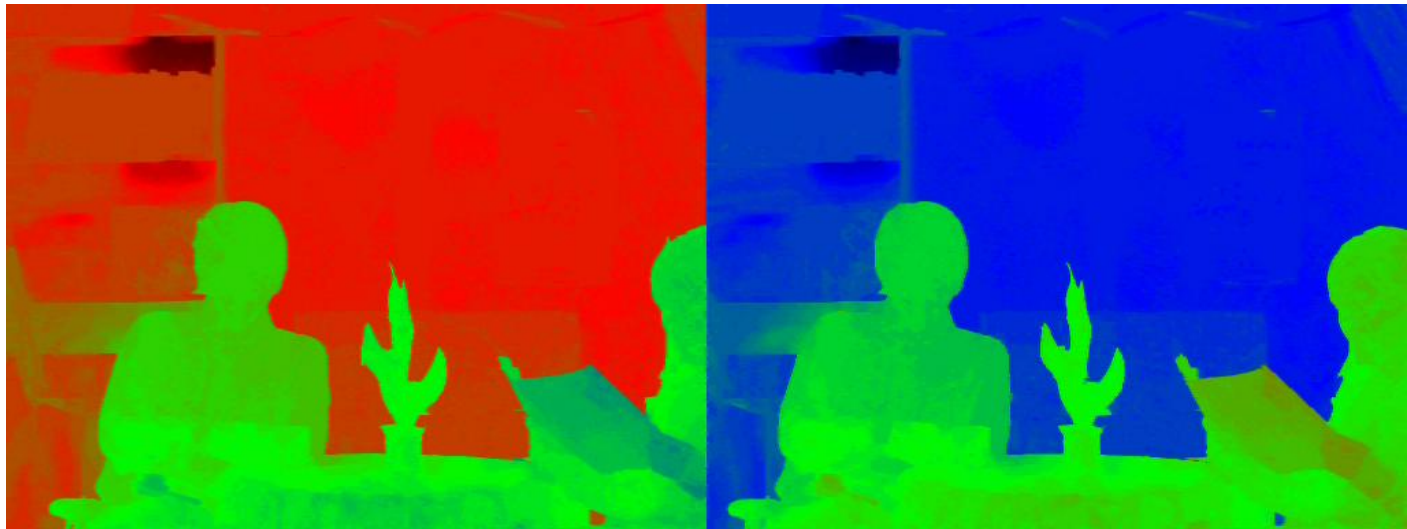
- Final Result



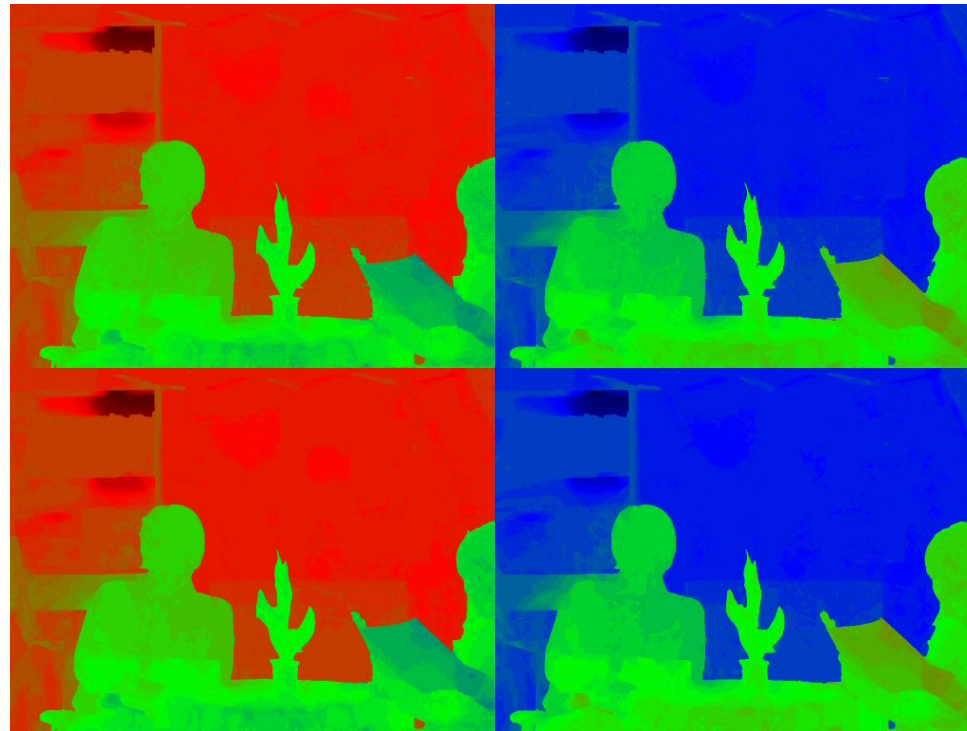
Example Iterative Post-Processing



Example Iterative Post-Processing



Example Iterative Post-Processing



Example Rendering: Parallax



Example Rendering: 2 Outer 3 Inner View

Left and right original view



Interpolated views at
0.25, 0.5, 0.75 between
the original views



Extrapolated views at
-0.5 and 1.5



More than two cameras

- Depending on the number of cameras
 - Select stereo pairs
 - Perform stereo matching like described
 - Fuse depth maps
 - Postprocessing

Example: 3 cameras

Camera1



Camera2

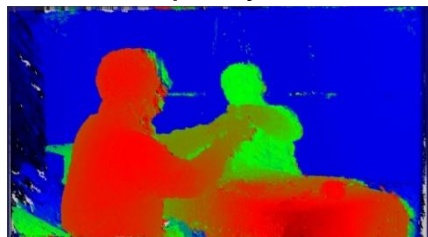


Camera3

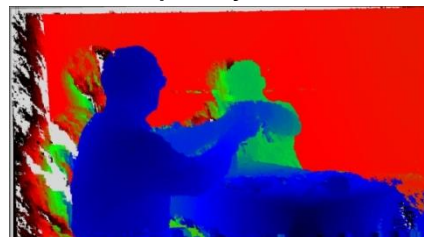


- Disparity estimation Camera 2 to Camera 1 -> Disparity_21
- Disparity estimation Camera 2 to Camera 3 -> Disparity_23

Disparity_21

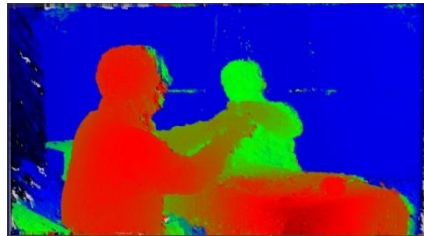


Disparity_23

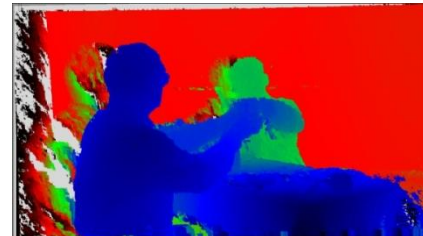


Example: 3 cameras

Disparity_21



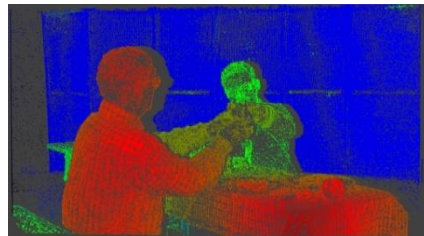
Disparity_23



Left-Right-Consistency Check Disparity_21 -> Disparity_lr_cons_21

Left-Right-Consistency Check Disparity_23 -> Disparity_lr_cons_23

Disparity_lr_cons_12

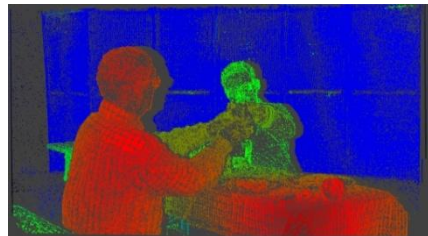


Disparity_lr_cons_23



Example: 3 cameras

Disparity_Ir_cons_21

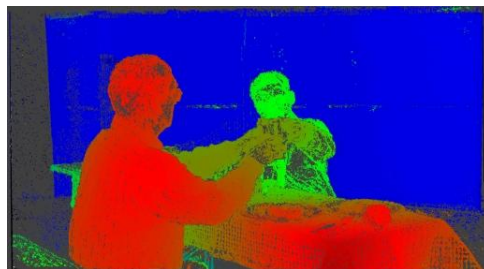


Disparity_Ir_cons_23



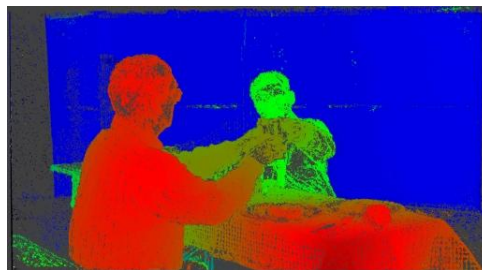
Fusion of Ir-consistent disparity maps

Disparity_fused_cam2



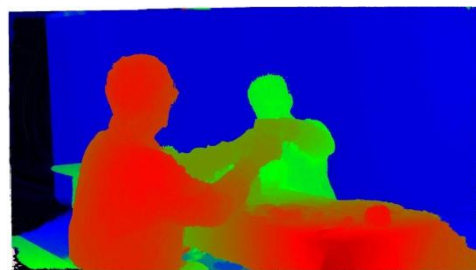
Example: 3 cameras

Disparity_fused_cam2



Cross-bilateral-median-filtering of fused disparity map

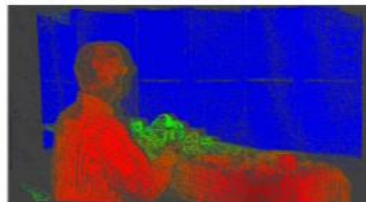
Final filtered disparity map



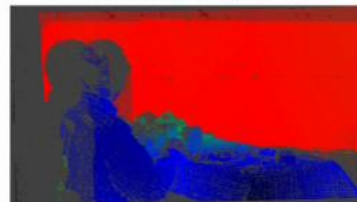
Example: 3 cameras



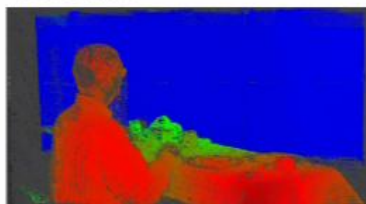
LR-Consistent cam2-cam1



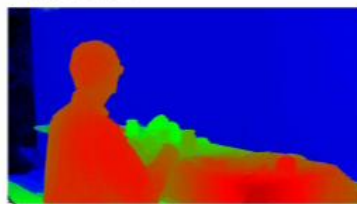
LR-Consistent cam2-cam3



Fused cam2



Final cam2



Example: 4 Cameras

Camera 1



Camera 2



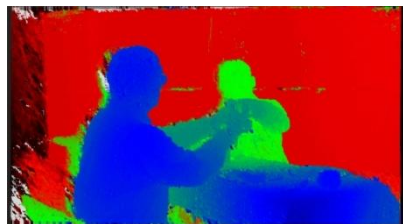
Camera 3



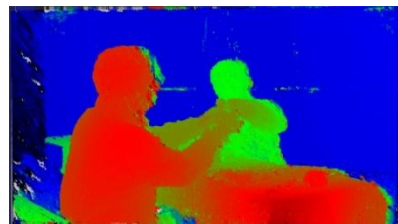
Camera 4



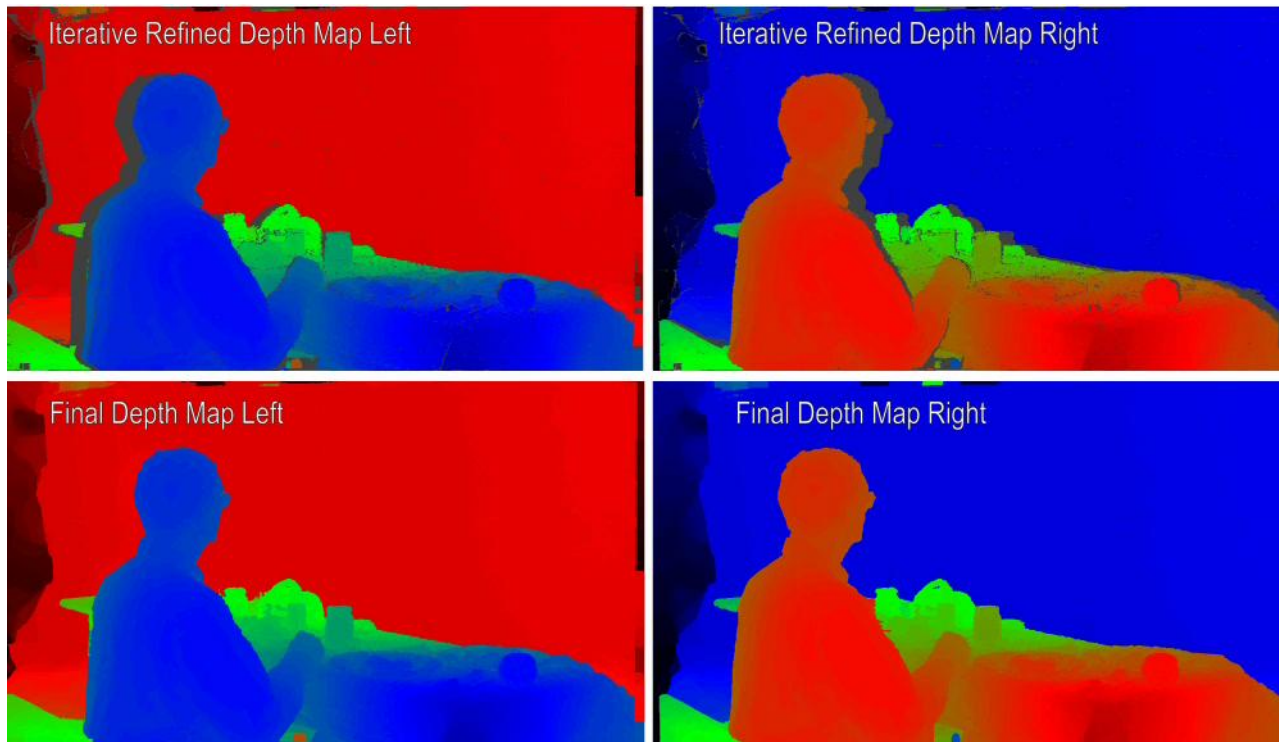
Depth Camera 2



Depth Camera 3



Example: 4 cameras



Example: 4 cameras

Left Original View



Right Original View



Interpolated 0.25



Interpolated 0.5



Interpolated 0.75



Future Work

- Improve temporal consistency
- Hierarchical approach to improve results in homogeneous areas and speed up the iterative process
- Occlusion Layer estimation

END

THANK YOU

