Towards On-Line Digital Doubles

Andreas Griesser¹ Nico Cornelis² Luc Van Gool¹,²

¹Swiss Federal Institute of Technology (ETH), D-ITET / Computer Vision Laboratory, Zürich, Switzerland
{griesser,vangool}@vision.ee.ethz.ch

²Katholieke Universiteit Leuven, ESAT / VISICS, Leuven, Belgium
{nico.cornelis,luc.vangool}@esat.kuleuven.ac.be

Abstract

We present a modular system for real-time 3D-scanning of human bodies under motion. The high-resolution shape and colour appearance is captured by several scanning units positioned around the object of interest. Each of these units performs a foreground-background segmentation and computes a valid depth-range for the spatially neighbouring units. Multiple depth-ranges are combined in a visual hull representation, which limits the search-range for the 3D-reconstruction. Depth-estimation is based on a hierarchical mult-view-stereo plane sweep approach. Robustness and accuracy is increased by incorporating imperceptible infrared illumination as well as adding local pixel gradient information. All parts of the processing pipeline, involving camera color conversions, segmentation, depth-range computation, visual-hull generation, lossless image compression, network transfer of the infrared and colour images, and the plane sweep algorithm, are implemented on the GPU and highly optimized for speed, allowing scanning times of less than 40ms per frame. Experimental results demonstrate the applicability of our system to the creation of high-density on-line digital doubles.

1. Introduction

Creating digital doubles of human bodies has enjoyed great popularity over the past decades. For several target applications - motion capture and film production are cases in point - real-time processing is crucial. Quite many systems capturing dense three-dimensional shape have been developed so far. However, most of the approaches focusing on one specific reconstruction method suffer from limited quality or lack real-time constraints. Among the most popular approaches are shape-from-stereo and shape-from-visual hulls. While silhouette-based visual hull methods require a considerable number of cameras positioned around the object of interest, the resulting 3D-models can be generated rather quickly. Despite the fact that multi-view stereo methods are computationally more expensive, their reconstructions are more accurate and visually more attractive. Combining both strategies seems to be promising and thus we follow this trend. The majority of proposed scanning systems only capture the images with high speed while reconstruction remains an off-line process.

Hasenfratz et al. [3] build a volumetric representation of the human body from 4 cameras based on voxel carving. The resulting models are relatively coarse and the system is not easily extensible. A different method is presented by Matusik et al. [7], who create a Polyhedral Visual Hull. This technique is enhanced by Li et al. [5] who utilize graphics hardware in order to speed up the scanning process. Nowadays GPU’s easily outperform general purpose CPU’s. Appropriately adapted stereo algorithms generate dense depth maps in real-time [10, 11, 14, 12]. Closely related to our work is the system of Li et al. [6]. They reconstruct a Polyhedral Visual Hull and supply the resulting depth range to a stereo algorithm. This technique eliminates false correspondences found in background areas but still lacks performance in low-textured surfaces. Sinha et al. [8] propose an elegant technique to combine stereo and visual hull data by formulating the problem based on a graph-cut. In the work of Ypsilos et al. [13] a random-dot infrared pattern is projected onto the human face, simplifying the search for pixel correspondences. Hilton et al. [4] use the same technique for off-line scanning people from multiple views.

In this paper we present a novel system capable of producing dense 3D-shapes from multiple views in parallel. First a partial Visual Hull is created from a subset of cameras, resulting in a per-pixel depth range for each input image. Then follows a plane sweep stereo matching algorithm [1], generating a subpixel-accurate depth map. Our system consists of several scanning units, each containing a computer, a camera and a light projector. Additional infrared
illumination, invisible for the observer, boosts robustness on low-textured surfaces, such as human skin or unicolor clothes. The overall system is modular and can easily be extended without adapting the core algorithms. Frame rates of up to 25fps can currently be reached with meshes delivered by a single unit containing approx. 60,000 3D-points.

The remainder of this paper is organized as follows: Section 2 describes the system architecture. The processing pipeline is explained in section 3. Section 4 is devoted to the core algorithms, in particular to color conversions, image compression, Visual Hull computation, and depth estimation. Section 5 proposes results and finally section 6 concludes the work.

2. System Architecture

The proposed system for scanning human bodies consists of several cameras and projectors positioned around the object of interest. 3D reconstruction is based on a combination of shape-from-stereo and shape-from-visual hull, whereby most processing is done on graphics hardware. The object is illuminated by LCD projectors and the appearance is captured by color cameras. Stereo correspondences on homogeneously textured surfaces can hardly be found and thus we apply near-infrared illumination by projecting a random-dot pattern onto the scene. This pattern is captured by dedicated greyscale-cameras, whereby their optical sensors are sensitive to the near-infrared frequency spectrum.

Color (CL) and infrared (IR) devices are equipped with optical filter glasses with an optical separation wavelength of 700nm. Industrial cameras are usually shipped with a built-in IR-cutoff filter, which can easily be replaced by glass plates with a similar refraction index. The color cameras we use have a cutoff at 650nm and the LCD-projectors cannot exceed the 700nm wavelength, dependent on the lamp source and the type of optical prism. Although conventional slide-projectors, as they are used in [13], provide high intensities in the infrared spectrum, the pattern cannot be changed and/or adapted to the scene. Therefore we use DLP-devices in our system with removed infrared filters in front of the light source. It is important to mention that the type of lamp source affects the amount of emitted IR light: Ultra High Purity (UHP) gas discharge lamps are among the most suitable ones for our purposes. Additionally we removed the internal color wheel in order to increase the IR intensity by a factor of 2. External IR filters in front of the DLP-lenses block all light below 700nm.

In order to obtain modularity and extensibility of our system, each pair of CL- and IR-cameras is connected to one computer and forms a scanning unit, generating a depth map w.r.t. to the CL-camera’s image plane. Dependent on the spatial placement of the units, IR- and CL-projectors are attached such that the scanned person is illuminated from all sides. Figure 1 depicts a part of our setup consisting of 5 scanning units. Each unit communicates with its neighbours through a switched Gigabit Ethernet network, allowing for bidirectional neto datarates of up to 100 Mbytes per second. A dedicated Master computer controls system behaviour and triggers the cameras, resulting in a perfectly synchronized image capture.

Calibration of the cameras is based on the method of Svoboda [9]: a small white LED, which can be seen by both CL- and IR-cameras, is swept through the scanning volume. Since the cameras are synchronized, point correspondences in all images are used to compute full projection matrices w.r.t. a common World Coordinate System. Reprojection errors of less than 0.4 pixels can be obtained.

![Figure 1. Modular System Setup: several scanning units are positioned around the scanned object. A depth map on the reference unit is computed using visual hull information of the 2 left and 2 right neighbours followed a pixel correspondence search in the stereo neighbour’s images.](image)

Optimal timing performance is gained by utilizing parallelism on graphics hardware. While the GPU is performing the majority of core algorithms, the CPU is reserved for image capture, network transmission, and image buffering.

3. Processing Pipeline

Initiated by a camera trigger, the scanning unit’s processing pipeline, starting with the image capture and ending with the depth map computation, is depicted in figure 2. The CL camera we use has a physical resolution of $1280 \times 1024$ px., color-coded by a Bayer Color Filter Array. GPU-based De-Bayering is performed, resulting in $640 \times 512$ px. RGB images (CL-RGB), which are required by the Master computer for texturing the final integrated 3D-model as well as by the local stereo-algorithm. The IR images from the greyscale cameras are downsized to the same resolution (IR-Grey).

A foreground-background segmentation procedure generates a smooth and compact binary change mask (Seg-Img).
Figure 2. Processing pipeline for each scanning unit. The grey-shaded parts mark network units as well as compressed data-structures.

w.r.t. a pre-recorded static background, which is required by the visual hull computation. Massive speedup is gained by running the following algorithms on the foreground pixels only and thus discarding irrelevant background areas. In typical situations, where the foreground covers only between 10% and 20% of the image area a theoretical speedup of factor 5 could be reached. We adapt a technique known as Early-Z Culling: drawn geometric fragments can be discarded even before they reach the fragment shader if they pass a depth-buffer test. This requires to setup the depth-buffer once on beforehand, based on the segmentation result.

A losslessly compressed version of the original texture (CL-RGB) is transmitted from all scanning units to the Master-PC. A greyscale-converted version of this image (CL-Grey) as well as the downscaled IR-image (IR-Grey) is sent to the stereo-neighbour, from which the corresponding image pair is received and decompressed.

The partial Polyhedral Visual Hull is computed by intersecting the silhouette cones of 5 scanning units: 2 left neighbours, 2 right neighbours, and the local machine. This would usually require to distribute all segmentation masks over the network, which turns out to be rather slow. Instead we split the creation of the Visual Hull into two parts: the Epipolar-Depth Range (EDR), and the Projective-Depth Range (PDR). In the first part we find intersections of the local segmentation mask with the epipolar lines in the current CL-image plane w.r.t. the corresponding neighbour. The list of intersections is converted into a geometric representation (Remote EDR-Buffer) and sent to the appropriate unit. In the PDR-step the received EDR-Buffers from the 4 neighbours are merged into a local depth-range image (depth range), storing per-pixel based minimum- and maximum depth.

The so-created depth range limits the search-range for correspondences in the stereo algorithm. The hierarchical plane sweep method we use [1] is capable of handling several input images and computes a depth map by finding the best pixel matches based on a color similarity measure (SSD or SAD). In our experiments we observed that the cameras, although of equal type and equal parameter settings, behave unequal. We reduce this problem of differing intensities in the input images by using local gradient information within a $3 \times 3$ pixel neighbourhood.

4. Core Algorithms

The following subsections are devoted to the different procedures of the processing pipeline. Although most of the parts are implemented on the GPU, little CPU intervention is required to setup and activate the vertex and fragment shader programs. The software framework is built on a multi-threaded basis, allowing the operating system’s scheduler to switch tasks as soon as blocking commands, such as network-transmission, CPU-GPU data transfer, and image grabbing, are executed.

4.1. Image Conversions

Color images from the locally connected CL-camera are organized in a Bayer-arrangement: each $2 \times 2$ pixel block contains one red, one blue, and two green color values. The generation of RGB images with full physical resolution would require color interpolation and thus increase the memory consumption by a factor of 3. Clearly, on a physical red pixel, the missing green and blue channels are formed by averaging the surrounding green and blue pixels, whereby the red value itself remains untouched. Notice that image noise, caused by low exposure times, is reduced on interpolated pixels. Another method downsamples the Bayer image by forming a RGB pixel from each $2 \times 2$ pixel block and thus interpolating only the two green channels. The resulting image size is half of the physical size and allows for faster processing.
We combine both methods by performing an upsizing step first, followed by a $2 \times 2$ pixel block averaging. This has the advantage of reduced image size and better noise reduction. Our implementation runs completely on the GPU in less than 1ms on a GeForce 6800GT. In order to have all images, required by the stereo algorithm, of equal size, the IR-data undergoes a downscale process by a factor of 2.

4.2. Foreground-Background Segmentation

Finding changed foreground regions in the CL-images is based on a color-similarity measure between the current frame and a pre-recorded average background frame ([2]). Simple background subtracting methods usually suffer from scattered segmentations and high sensitivity to illumination changes. Especially environments with changing lighting conditions, i.e. by projecting patterns or shadows casted from the scanned person, demand for more sophisticated algorithms. Our method uses an extended colinearity criterion approach and iteratively solves a Markov Random Field for smooth and compact segmentations.

Our GPU-based approach is slightly illumination-invariant and processes image sequences in less than 4ms per frame, resulting in smooth and compact segmentations, even on low-contrasted images.

4.3. Lossless Image Compression

The image transfer of $640 \times 512$ RGB pixels over a Gigabit Ethernet network takes approx. 8-12ms, depending on the network protocol. Based on a graphics card with AGP 8x interface, the transfer between GPU and CPU is still a bottleneck, providing only approx. 250MB/s bandwidth. Thus, the download of the color image from the GPU takes additional 4ms while on the receiver side the upload can be done in approx. 2.5ms. The overall transmission time for an RGB image between the GPU’s on different scanning units is 17.5ms, and thus too high for real-time systems.

We developed a novel compression algorithm that rearranges the input image in 8-pixel wide blocks. Each block is marked as ‘background’ if it comprises only background pixels in the corresponding segmentation mask. Otherwise a ‘foreground’ label is assigned. A Look-Up-Table (LUT) stores the label for each block. Afterwards the original image undergoes a defragmentation procedure, whereby background blocks are replaced by foreground blocks. This strategy reduces the relevant image size and allows for faster GPU-CPU transmission as well as faster network transfer.

In our experiments we measured between 5% and 20% foreground regions, resulting in compression times of less than 1.3ms, including download of the compressed image from the GPU to the CPU. Decompression can be done in less than 1ms including GPU upload while network transmission time is 2.2ms. Thus, we could reduce the overall transmission time from 17.5ms to 4.5ms without loss of information.

4.4. Visual Hull Computation

The search for correspondences in stereo images can be limited by a disparity range or, as it is the case for our plane sweep method, by a depth range. This range is defined by a Visual Hull, based on the segmentation masks of each participating camera. Since a depth range is required by all units reconstructing a depth map, the complete Hull from all cameras in the system would need to be projected into their image planes. However, in most cases it is sufficient to compute a partial Hull from only a small subset of cameras. We propose a rapid method which uses segmentations from 2 left and 2 right neighbouring units as well as from the current machine. The basic principle is similar to the approach presented in [7] but is computationally less expensive.

Figure 3 visualizes the principle of depth range computation for a given pixel with coordinates $u, v_{ref}$ the intersections of the projected epipolar lines in the neighbouring image planes are computed. The valid depth range is between points $D$ and $E$.

![Figure 3. Epipolar geometry for the creation of the Visual Hull from 3 cameras. For a considered pixel at position $u, v_{ref}$ the intersections of the projected epipolar lines in the neighbouring image planes are computed. The valid depth range is between points $D$ and $E$.](image.png)
The described process is split into two parts, the first computes a list of line-segmentation intersection pairs whereas the second combines the lists from the 4 neighbouring units and computes the final depth range. Both parts are explained more in detail in the following subsections.

4.4.1. EDR - Epipolar Depth Range

Consider a current image plane and a given reference image for which the intersections need to be found. Assume the epipole (i.e. the projection of the reference camera center into the current image) lies outside of the image plane, as shown in figure 4, all \( N = \text{width} \cdot \text{height} \) epipolar lines could be constructed. They 'enter' the image at points \( P_1 \) and 'leave' at points \( P_2 \). Considering pixel-accuracy in the segmentation masks, the total number of lines to be tested can be reduced by merging those lines with equal pixel coordinates for \( P_2 \). Clearly, if two epipolar lines have equal integer-rounded exiting points \( P_2 \), they are merged without loss of generality. This reduction leads to a maximum of \( N_0 = \text{width} + \text{height} - 1 \) lines surviving this test for the given example.

![Figure 4](image)

**Figure 4.** The total number of epipolar lines, entering the image plane at points \( P_1 \) and leaving at points \( P_2 \), is reduced by merging all lines with equal coordinates for \( P_2 \). The coordinates of the lines survived are stored in two 32bit look-up-tables (EDR-LUT).

For each new image frame, a GPU fragment shader program finds the intersections with the valid epipolar lines as shown in figure 5. Starting at a point \( P_1 \) a value \( \text{seg} \) is set to 1 within foreground areas and 0 otherwise. Walking along the line until \( P_2 \) is reached, the following quantifiers are computed:

\[
\begin{align*}
\text{dist} &= \text{len}(P_1P_2) \\
\theta &= 0..\text{dist} \\
d_1 &= \text{seg} \cdot d \\
d_2 &= \text{seg} \cdot (d - \text{dist})
\end{align*}
\]

The desired intersection points \( d_{\text{min}} \) and \( d_{\text{max}} \) are found by taking the minimum of \( d_2 \) and the maximum of \( d_1 \). From the implementation point of view, the above calculations are performed within a loop in the fragment shader program. We speed up processing by increasing the loop variable \( d \) with a step size of 5 pixels in a first run and then refine it with a step size of 1 pixel.

![Figure 5](image)

**Figure 5.** Walking along an epipolar line between \( P_1 \) and \( P_2 \) updates the quantifiers \( d_1 \) and \( d_2 \). Intersections with the segmentation are defined as \( P_{\text{min}} \leftarrow (d_{\text{min}} = \text{min}(d_2)) \) and \( P_{\text{max}} \leftarrow (d_{\text{max}} = \text{max}(d_1)) \).

Based on the geometric configuration shown in figure 6 both intersection points \( P_{\text{min}} \) and \( P_{\text{max}} \) are converted into their corresponding angles \( \alpha \) between the current camera center \( C_x \) and the line \( c = C_xC_{\text{ref}} \). Given the \( 3 \times 3 \) matrix \( K_x \) with the intrinsic camera parameters we compute the angle as follows

\[
\begin{align*}
b_{\text{min,max}} &= K_x^{-1} \cdot \begin{bmatrix} P_{\text{min,max} \cdot x} P_{\text{min,max} \cdot y} 1 \end{bmatrix}^T \\
\cos \alpha_{\text{min,max}} &= |b_{\text{min,max}} \cdot c|.
\end{align*}
\]
The values $\cos \alpha_{\text{in}, \text{max}}$ are stored in a 32bit RGBA texture (EDR-buffer), whereby 16bit resolution is used for each component. In our system the input images have 640×512 pixels size and the number of valid epipolar lines $N'$ is typically between 800 and 1400. Thus, the EDR-buffer consumes between 3.2 and 5.6 kBytes, which is much less than load to the CPU.

After each scanning unit discarded, the following processes on the GPU can utilize the Early-Z-test and obtain a markable speedup dependent on the amount of foreground area. Based on the geometric configuration shown in figure 6, the length of the vector $a$ for each received $\cos \alpha_{\text{in}, \text{max}}$ from unit $i$ is computed as follows:

1. Project a considered pixel at position $u, v$ in the reference image into three-space:
   \[
   \mathbf{r} = K_{\text{ref}}^{-1} \cdot [u \ v \ 1]^T.
   \]
2. Fetch the corresponding $\cos \alpha$ value from the received EDR-buffer.
3. Compute $\cos \beta = \mathbf{r} \cdot (-\mathbf{r})$.
4. Compute the length of $\mathbf{r}$:
   \[
   ||\mathbf{r}|| = ||\mathbf{r}|| \cdot \frac{\sin \alpha}{\sin \gamma} = ||\mathbf{r}|| \cdot \frac{1}{\sin \beta / \tan \alpha + \cos \beta}
   \]
   The above sequence is repeated for all $\cos \alpha_{\text{in}, \text{max}}$, and the integrated depth range is defined by
   \[
   \begin{align*}
   \bar{\alpha}_{\text{in}} &= \max_{i=1...n}(\bar{\alpha}_{\text{in},i}) \\
   \bar{\alpha}_{\text{max}} &= \min_{i=1...n}(\bar{\alpha}_{\text{max},i})
   \end{align*}
   \]
   whereby $n$ is the total number of participated Visual Hull neighbours. Processing time for $n = 4$ neighbours and a foreground area of 20% is less than 1.5ms.

### 4.5. Multi-View-Stereo Plane Sweep

Each scanning unit generates a depth map w.r.t. the local CL-camera reference image by searching for corresponding pixels in the CL-IR image pair of the stereo-neighbour. A projective plane parallel to the reference image plane is swept through three-space at a number of discrete depths and projected onto the target images. The method is derived from the GPU-based implementation presented by Cornelis [1] and gains speed and reconstruction quality from its hierarchical nature. Color similarity is measured by a sum of squared difference (SSD) or sum of absolute difference (SAD), whereby the latter behaves slightly better in our setup.

Since correspondences in color images can barely be found on homogenously textured surfaces, we actively project a random-dot pattern in the near-infrared frequency spectrum, invisible for the CL-cameras. Robustness of the scanning process is increased by combining both color-SAD (CL-SAD) and infrared-SAD (IR-SAD). In arbitrary setups the brightness of the camera images could slightly differ from each other, caused by unequal lenses and/or imprecisions in the manufacturing process of the imaging device. Therefore we modify the matching method by using local pixel gradient information within a 3×3 neighbouhood instead of pixel intensities directly. The resulting 8 gradient values for each of the 4 input images are computed on the GPU. The SAD values between the reference image (ref) and the corresponding image (cor) are defined as follows:

\[
\begin{align*}
\text{SAD}_{\text{IR}} &= \sum_{i=1}^{8} |\text{grad}_{\text{IR,cor},i} - \text{grad}_{\text{IR,ref},i}| \\
\text{SAD}_{\text{CL}} &= \sum_{i=1}^{8} |\text{grad}_{\text{CL,cor},i} - \text{grad}_{\text{CL,ref},i}|
\end{align*}
\]

The final matching score for a pixel under inspection is

\[
\text{SAD} = \min(m_{IR}, w_{IR} \cdot \text{SAD}_{IR}) + \min(m_{CL}, w_{CL} \cdot \text{SAD}_{CL})
\]

whereby $m_{IR}$ and $m_{CL}$ are arbitrary thresholds, indicating the maximum radius of influence in gradient space. $w_{IR}$ and $w_{CL}$ are additional weighting factors, currently set to $w_{IR} = 1$ and $w_{CL} = 2$.

During the sweep the related depth value is assigned to a pixel if it scores a better (i.e. lower) SAD value than the one already stored. Fine structures, such as sharp edges on the object, are retrieved by iteratively testing the currently considered pixel with depth values stored in the surrounding pixels. For more insights we refer to [1].

It is important to mention that all computations are necessary on pixels with a valid depth range. Thus, a speedup is obtained by setting up the Early-Z-test with the result...
from the Visual Hull computation. In contrast to the original implementation of the plane sweep algorithm we do not start with the initial sweep at user-defined depths. Instead, since a per-pixel based depth range is given, we start sweeping from the front-face (min. depth) towards the back-face (max. depth). This eliminates false correspondences possibly found in the background area of the stereo neighbour images.

5. Results

In the following examples we arranged 8 scanning units around the object in a distance of approx. 3 metres. 5 of these units are positioned such that a baseline of 1.5m is reached. The other 3 units are required for a more accurate estimation of the Visual Hull. The several processing computers are equipped with a 3GHz Pentium CPU and a NVIDIA GeForce 6800GT graphics card.

Figure 7 shows the computed Visual Hull depth range from 5 views (2 left and 2 right neighbours, 1 reference view). The red and green dots indicate the per-pixel minimum, respectively maximum depth range. Processing time, given the segmentation mask, including network transfer of the 4 EDR-buffers is approx. 6ms.

Figure 8. Partial Visual Hull from 5 cameras: red dots indicate the minimum depth and green dots the maximum depth range per pixel. The yellow lines indicate the line of sight between the camera centers and the world coordinate system origin.

In figure 8 the 3D reconstruction on one of the scanning units is shown. The CL- and IR-input images in the top row are converted into gradient images and the resulting depth map is given in the bottom left picture. As can be seen in the CL-images, even in oversaturated or homogenous surfaces on the shirt, a correct depth value is derived. A more detailed view of the body is given in the bottom right part of the figure. Processing time of the plane sweep for given model is approx. 19ms, whereby a total number of 50,000 points is reconstructed. One can observe that the lower part of the legs are disconnected from the body. This is caused by those cameras, which could not observe these areas and thus no stereo correspondences can be found.

The reconstructed 3D-meshes from 3 scanning units is shown in figure 9, whereby processing time for 70,000 points is less than 23ms.

6. Conclusion and Future Work

We propose a modular system for 3D-scanning of human bodies in real-time. The majority of processing is done on graphics hardware, involving color conversion, foreground-background segmentation, image compression, Visual Hull computation, and multi-view-stereo depth estimation. Pixel correspondences, even in homogeneously textured surfaces, are correctly found by the integration of infrared illumination. Depth and appearance is captured simultaneously on several scanning units positioned around the object of interest. Scanning speed of up to 25fps is obtained by highly optimized implementations and scales well with the processing power of the hardware components. System modularity is obtained in the sense of adding more units without penetrating the processing times. Rearrangement of the scanning units for special purposes, such as face-scanning for exam-
Figure 9. Pointclouds and textured surfaces from 3 different scanning units. Left: 60.000 points (20ms). Middle: 50.000 points (19ms). Right: 70.000 points (23ms)

ple, is straight forward. The combination of Visual Hull and multi-view-stereo together with infrared illumination operating at real-time speed makes the system unique in its own way.

In the near future we plan to adapt the infrared pattern to the scanned object by minimizing the overlapping regions of multiple illuminations and adding temporal constraints. Integration of the final model on a dedicated computer is still off-line but we are working on on-line versions in order to provide a real-time view of the captured sequence.

References