Online 3D Acquisition and Model Integration

Tobias Jaeggli[1] Thomas P. Koninckx[1] Luc Van Gool[1,2]

[1]Katholieke Universiteit Leuven, ESAT / VISICS, Leuven, Belgium

[2]Swiss Federal Institute of Technology (ETH) D-ITET / BIWI, Zürich, Switzerland

{*tjaeggli,tkoninck,vangool*}*@esat.kuleuven.ac.be*

Abstract

This paper presents a system which yields complete 3D models in a fast and inexpensive way. The major building blocks are a high-speed structured light range scanner and a registration module. The former generates 'raw' range data whereas the latter performs a fast registration (ICP) and renders the partially integrated model as a preview of the final result.

As the scanner uses only a single image to make a reconstruction, it is possible to scan while the object is moved. The use of an adaptive projection pattern gives a more robust behaviour. This allows the system to deal more easily with complicated geometry and texture than most other systems. During the scanning process the model is built up incrementally and rendered on the screen. This real-time visual feedback allows the user to check the current state of the 3D model. Holes or other flaws can be detected during the acquisition process itself. This speeds up the model building process, and solves indirectly the problem of view-planning. The whole real-time pipeline - comprising acquisition, merging and visualization - only uses off-the-shelf hardware. A regular desktop PC connected to a camera and LCD projector is turned into a high-speed scanner and modeler. The current implementation has a throughput of approx. 5 fps.

1. Introduction

Creating 3D models of real objects is a task that continuously gains importance in different commercial and scientific areas. Nevertheless, the technology needed to produce such models is not yet easily accessible. Available systems typically involve high costs and long acquisition times. In addition, a substantial amount of manual processing by experts is often unavoidable.

Most range scanners only deliver data from one side of the scanned object at a time. For many applications, however, complete models of complex objects are needed.

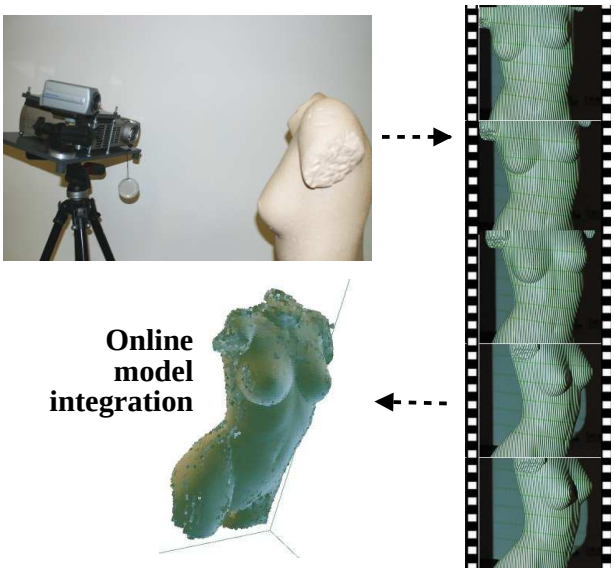


Figure 1: This paper gives an overview of a high-speed acquisition and modeling pipeline. Top left: the range scanning setup. Right: input video sequence of moving object. Bottom left: online model integration.

Therefore a multitude of individual range images from different views have to be 'stitched' together. This requires a complete 3D modeling process, which includes *view planning*, *acquisition* of range data, *registration* to position each scan relative to the other scans, *integration* to produce a single new surface and *validation*. This last step is usually done visually and requires *rendering*.

It is exactly for this set of closely related tasks that we propose an integrated solution. Range acquisition, merging and rendering is done on the fly. This allows for a close collaboration between these modules, and in addition it gives direct feedback to the user who can judge the partial result.

Relatively little work has focused on on-line 3D modeling systems so far. Commercial systems have traditionally

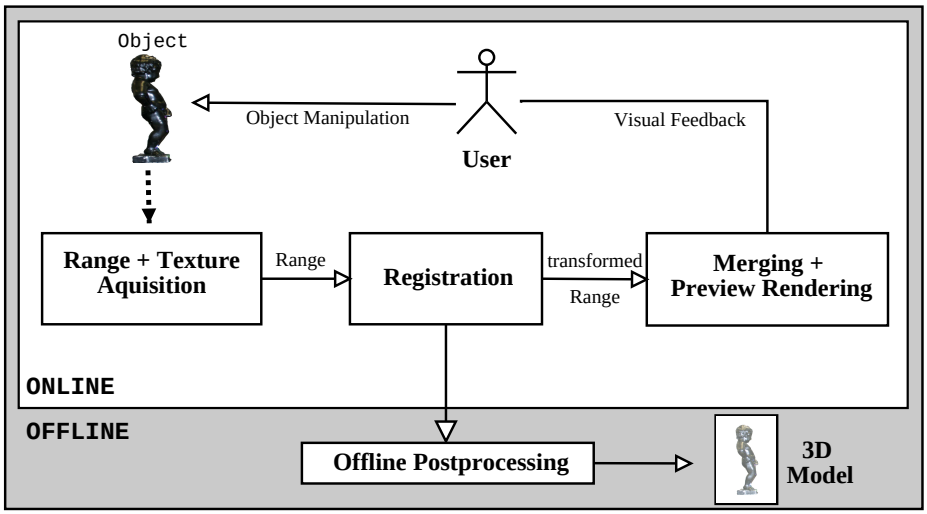


Figure 2: *The 3D-Model Acquisition Pipeline, comprised of an on-line and an off-line block. On-line a crude model is built up and displayed in order to inform the user about the current status of the scanning process. Offline a high quality model is constructed.*

used high-accuracy laser-scanning techniques for range acquisition and semi-automatic tools to build complete models. They however often suffer from drawbacks mentioned before. This is in most cases due to the lack of a quality estimation at the moment of the acquisition.

Recently, commercial structured light scanners have become available [4, 6]. These systems can be used in a more flexible way than laser scanners. However, they still can't give immediate and constantly updated visual feedback.

The feasibility of real-time scanning has recently been demonstrated by Rusinkiewicz et al.[12], at Stanford University. Range data are obtained by structured-light triangulation, using a time-coded projection pattern, that allows for slow movement of the scanned object. As a first, main difference, our scanner can reconstruct shape from a *single* frame (one-shot) and is therefore less sensitive to fast movements. Secondly, the projected pattern is dynamically adapted to the observed scenery. This makes this system more robust in cases with complicated geometry and texture. A third extension when compared is the acquisition of the original texture (based on images taken without the structured light). Fourthly, a multiview refinement makes the registration more reliable and allows for long acquisition sequences with minimal drift. Another strength of our system is the fact that it runs on a standard PC and does not need any special hardware like high-speed cameras or special or adapted projectors.

In summary, although closely related to the scanner developed at Stanford, this work presents a series of new functionalities which result in an overall performance improvement. To the best of our knowledge this is the first implementation which presents an *on-line single-shot scanning and integration pipeline.*

The rest of the paper is organised as follows: In section 2 we give an overview of the functionality of the system. Section 3 describes the general architecture and covers the major building blocks. Preliminary results are presented in section 4. The paper is concluded in section 5.

2. System Description

One of the main difficulties when capturing range data is to know how many images to take and from which viewing directions. In the presented system, this is solved in an intuitive way by giving the user the possibility to examine the partial integration of the object at any time. Information about the quality of the scans and their registration is available *during the acquisition process*. Spotted holes can be filled in on-the-fly. Manipulation of the object in front of the scanner yields scan data of all surface parts. This manipulation is rather uncontrolled and can be often abrupt. One-shot acquisition is important to deal with this dynamic behaviour.

Together with the user who moves the object (or alternatively the scanner around the object), the online stages of the pipeline form a feedback loop.

The setup is based on commodity hardware which results in a user-friendly and inexpensive device. The range scanner uses a common video source such as a DV-camera or a IEEE-1394 camera and a LCD-projector, all other parts are

implemented in software only.

3. An Integrated Scanning and Registration Pipeline

The online model acquisition pipeline we propose consists of a set of basic building blocks (see also figure 2):

- *Range and texture acquisition:* Computation of raw range-data for each incoming frame. Only parts in which structured light illumination could be identified yield surface patches. A second shot without any special illumination and taken in short succession acquires the corresponding texture.
- *Registration of partial surfaces*: The outcome is a transformation from the scanner-oriented incoming patch data to the model-oriented reference frame.
- *Merging and rendering*: The model consisting of all currently accumulated scans is shown on the screen.

Finally the data is postprocessed and integrated into a single, high-quality model.

The integration of these building blocks in a single pipeline has not only the advantage of an immediate (partial) result, but also gives the registration and rendering modules access to all data which are normally internal to the scanner. This shared memory first of all avoids expensive data transfers between modules. This applies mainly to the scan data. It also avoids to execute similar tasks multiple times. A typical example is the generation of the surface normals. This is rather inexpensive when done while generating the surface. The normals are needed to compute the registration efficiently and for the rendering. Also the knowledge of both camera and projector calibration, can be used to speed up the registration. A preliminary attempt to exchange a quality estimation for each data point is already implemented. In future versions, we plan to include additional feedback between the modules.

3.1 Range and Texture Acquisition

The 'Range and Texture Acquisition' unit delivers range data and texture images. For details about the actual range acquisition and the calibration of the system, we refer to earlier work ([7, 8]). The 'base pattern' used with this scanner is a set of equidistant vertical black and white stripes. In the transversal direction an identification code underlies this first pattern. This allows to solve the correspondence problem. Geometrical and color properties of both the base pattern and code pattern are adapted on-line to the content of the scene, and to the current camera-projector configuration. This adaptive behaviour boosts the robustness of the overall setup.

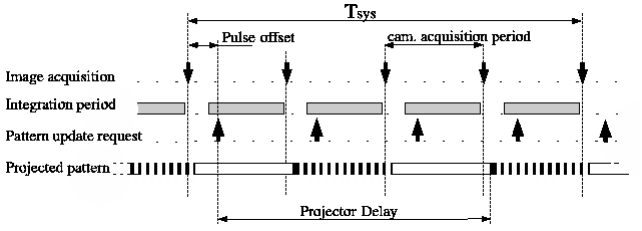


Figure 3: *The synchronization problem: the change of the projected pattern should occur exactly between the acquisition of the previous frame and the start of the integration period of the next frame.*

For the texture acquisition a second camera shot without the structured light illumination is needed. Without noticeable delays between the command to the projector and the image acquired by the camera this would be a trivial task. In reality however, tight synchronization between camera and projector is required. If this constraint isn't satisfied texture images are partially exposed to the structured-light pattern, and the range images are degraded .
A second reason for synchronization is situated within the scanner itself. The projected pattern varies over time, based on the current content of the scene. To make it possible to distinguish between consecutive projected patterns, the camera and projector should be synchronized with the processing and adaptation algorithm. (see [8])

3.1.1 Camera-projector synchronization

First of all an estimation of the delay between camera and projector is made. To this end at time t_0 white light is projected. A short pause guarantees that the output of the projector is switched to this uniform illumination. In a next step, at time t_1, the projection is switched off and the camera is started. The acquisition is stopped two frames after the detection of an abrupt intensity change in the acquired images. Within this set of images, we look for the first without a horizontal intensity transition. This is done in order to discard images which are only partially exposed. The timestamp t_2, corresponding to the acquisition of this image gives a rough delay estimate $t_2 - t_1$. This delay is only accurate up to the resolution of the measurement device, being the framerate of the camera. It will be referred to as the 'System Delay' T_{sys}. For common camera-projector pairs T_{sys} lies typically between 70 and 200 msec, which corresponds to the interval of 3 to 8 camera frames.

Low frame rate, high quality textures

The most intuitive synchronization is achieved by waiting for a period T_{sys} after every pattern update. Schematically:

grab a frame → switch off structured light → wait T_{sys} for the projector to stabilize the output → grab a frame → turn on illumination → wait T_{sys} → ...

Although perfect textures result in this way, this approach has a rather limited throughput. The waiting in between the acquisitions limit the frame rate to approx. 10 Hz. or 5 dual shots (texture and range) per second. Of course, while waiting only the acquisition thread needs to be suspended and the CPU is free to process the data ,so no time is 'lost'. (see also section 3.1.2)

Feedforward synchronization
Instead of waiting for the projector to stabilize its output during each cycle, it is also possible to apply a modification to the pattern, which will only be observed a couple of frames later.

We refer to the time between a request for a pattern change and the actual update of the pattern as 'projector delay' [1] (see figure 3). This time interval is shorter (less than one camera acquisition period) than the measured T_{sys}. Given this unknown but fixed projector delay we start to offset the 'pattern update request' with a small amount of time. For a small interval of values this 'pulse offset' will cause the pattern transition to be located in the desired time interval.
This system works for true 'single shot cameras', which start their integration on demand. As long as frame requests to the camera occur at a fixed frequency all timings stay stable and synchronization is preserved.
For cameras with fixed internal frame-rate (e.g. a DV handy-cam) we however have no control over the actual moment the integration starts. We only notice when a new frame is ready. In case of frame-drops, or frame requests at an odd division of the internal frame-rate (which also will cause lost frames) T_{sys} is no longer completely fixed. The projector delay however remains unchanged and soon the camera-projector synchronization will be lost.
This problem can be fixed to some extent. We save in a FIFO-buffer with length x a timestamp t_i each time a new pattern update is requested. x is the number of frames corresponding to T_{sys}. Before grabbing the next image, we check the timestamp available at the first position of the buffer, which now corresponds to t_{i-x}. The moment when the next image is acquired is delayed with $(T_{sys} - PulseOffset) - (t_i - t_{i-x}) - t'$. This is the difference between the measured time and the desired fixed T_{sys} time interval. The subtraction of 'PulseOffset' compensates for the time difference between the moment of this time measurement and the moment that the new time stamp will be added to the FIFO. If this value is negative, we need to discard this image in order to avoid swapping the content of the texture and image buffer. The next projector pulse, which has an offset against the camera acquisition is also automatically readjusted. The camera itself is insensitive to noise on the timing of the grabbing as it will transmit the last integrated image anyway. The FIFO assures that we readjust to the moment when the upcoming pattern modification was requested instead of only adjusting to the interval of one period. The result is that in most cases the lock between camera and projector can be retained. Experiments gave very good results up to approx 7.5 Hz dual-shot acquisition, which corresponds to an actual synchronization at 15 Hz. Acceptable results were still possible up to 20 Hz. *Higher camera frame rates would allow for higher synchronization speeds, because of a smaller penalty in case of frame drops.*
If synchronization is needed at higher frame rates than possible with this software-only implementation, one might use the vertical sync of the VGA-signal to trigger the camera. The major limitation of the software implementation is the limited accuracy of the timestamps on a non real-time OS.

[1] caused by finite response-time of the device and an internal image buffer

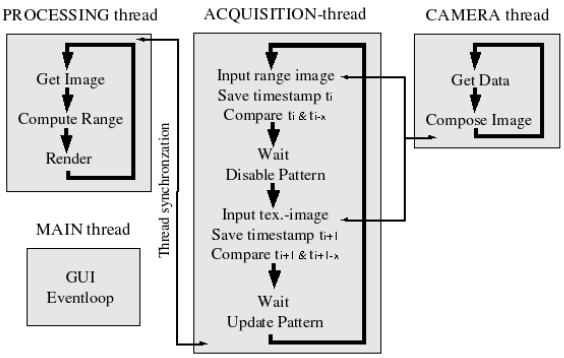


Figure 4: *Implementation: a separate acquisition-thread allows to keep projector-camera timings fixed, processing is synchronised to the acquisition by thread synchronisation. In case a camera with fixed internal frame-rate is used, this device acts as a sync-generator for the rest of the pipeline. The main eventloop introduces one additional thread.*

3.1.2 Implementation aspects

The camera-projector synchronization is implemented as shown in figure 4. The processing time needed for a single input frame is dependent on the complexity of the scene and the noise level in the input image. This makes it necessary to decouple processing and acquisition, as the latter requires strict timing as is explained in the

previous section. The acquisition thread grabs the image and updates the pattern, and informs the processing thread that a new image is available. The processing thread then computes the range map based on the image. The slowest of both steps determines the execution speed of the system. This is done to avoid 'spinning'. We shouldn't grab more images than can be processed, and every image should be processed only once.
In case a camera with fixed frame-rate is used, this device will act as a sync-generator for the rest of the pipeline. The event loop and GUI takes an additional thread.
The use of a dual-head graphical board allows us to configure two displays. One steers the structured light pattern to the projector, the other is needed for the display.

3.2 Registration

The range scanner described in the previous section provides 3D and texture data at a high frame rate. To produce complete 3D models the object has to be moved in order to present it from all sides to the camera. This yields a high number of surface patches that one has to bring into a common coordinate system by applying a rigid transformation to each of them. The first patch is arbitrarily chosen as the reference. All following 3D scans are transformed in order to fit with this first scan.

Most commercial systems use calibrated movements of the object or the scanner, often realized with a robot arm or a rotation device. As this specialized hardware makes the setup expensive and bulky, and thus seriously limits the applicability, we prefer doing without. Therefore we chose a software-only solution that uses the geometry of the reconstructed surfaces.

Because of the high framerate of the range scanner, it is reasonable to assume that the movement of the object between successive scans is small. Therefore the transformation of the previous scan presents a good initial estimate for the transformation of a new scan. In other words, the problem of crude registration is already solved, which makes it possible to perform geometry-based fine registration using the Iterative Closest Point (ICP) algorithm described in the next section.

3.2.1 Pairwise Registration

ICP [1, 2] is a well known algorithm for aligning two 3D surfaces in the form of point sets $P = \{p_i, i = 1..N\}$ and $Q = \{q_j, j = 1..M\}$. Iteratively the following steps are repeated until a convergence criterion is satisfied:

- Build a set of point pairs (p_i, q_j) consisting of 'closest' points (cfr.inf.) of the overlapping area of the two surfaces P and Q.
- Compute a rigid transformation T that minimizes an error measure, usually the sum of squared distances $\sum dist(Tp_i, q_j)^2$ between corresponding points p_i and q_j.
- Apply Transformation T to P.

A number of variations of ICP with different qualitative and quantitative performance have been proposed (see [10] for a comparison). Our implementation focuses on computational efficiency and robustness to noise rather than highest possible accuracy or insensitivity to degenerate cases.

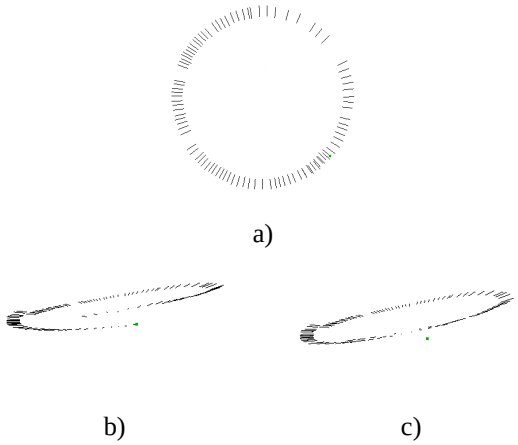
The distance measure $dist(Tp_i, q_j)$ to be minimized can be either the euclidean distance of the points or the distance from p_i to the tangent plane to Q in q_j [2]. For the first case a closed form solution to compute T has been proposed by Horn [5], whereas for the point-to-plane distance measure a solution for T can be found either iteratively or by linearizing the problem assuming small rotation angles. ICP with point-to-plane minimisation is less sensitive to aliasing caused by the discretisation of the surfaces, and its convergence rates are an order of magnitude higher than with the point-to-point approach. Unless the scans are very noisy they can be aligned in 3 to 6 iterations.

The most time-consuming operation is the search for closest points; the naive approach to find for each p_i the closest point q_j in 3-space requires NM comparisons. This process can be speeded up by taking advantage of the fact that the 3D points of each scan originate from 2D images. By projecting p_i into the range image that corresponds to Q using the current estimate of the relative transformation T (and the camera-calibration), a corresponding point q_j can be found in constant time. Geometrically this corresponds to a search for corresponding points along the viewline of the camera of Q and has the convenient side-effect of implicitly excluding those points from the set of point-pairs, which are in the non-overlapping part of P and Q.

Subsampling the point set P results in a further speed-up of the whole registration without affecting its quality, if done reasonably. In our experiments, subsampling factors of up to 10 yield good results, whereas a further decrease of the number of point pairs (less than 5-10%) might lead to misalignments.

Noisy data can be (partly) dealt with by eliminating outliers from the set of point pairs. The possible criteria[2] to classify a pair as in- or outlier include *distance of points* in a pair, *normal compatibility* and rejection of points on *mesh boundaries*. We reject point pairs that have a bigger distance than some *dynamic threshold*, which is decreased when the two surfaces move closer to each other. It is initialized to ∞ (no rejection), and is lowered after each iteration to twice the average distance of all the point matches, but never smaller than twice the sample density (average

[2]Based on the same criteria a weight can be assigned to the point-pairs.



a) b) c)

Figure 5: Rotation sequence: The camera centers form a circle. a) top view, b) accumulation of errors, c) after multiview refinement

spacing between reconstructed 3D points). This process allows for surfaces that are initially relatively far from each other and rejects points which belong to wrongly reconstructed surface parts when surfaces are close.

The residual error (distance measure after registration) gives a rough estimate about the general quality of the registration. The same distance measure *including the outliers* can be used as an indication about the amount of noise or errors in a scan, although we don't know which of the surfaces is incorrect.

Errors in scans can be detected as long as they are in the overlapping part of the surfaces, but it is difficult to localize the wrongly reconstructed part of a surface. Therefore we adopt the strategy to reject scans entirely, if they are likely to contain errors. This can be justified by the high 3D scan acquisition rate. Once a first (reference) scan has been chosen, each new 3D scan is aligned with the previous scan. Based on the residual registration error and the error *including outliers* a decision is made whether the new frame has to be accepted or rejected.

3.2.2 Multiview Refinement

When aligning a big number of surface patches (typically several hundreds) as described in the previous section, errors accumulate, even if the error of an individual pairwise registration is small. The error is best visible when the scanned object is rotated 360 degrees around its own axis (see also fig. 5). The first and the last patch of such a sequence should be perfectly aligned, however due to accumulated errors this is generally not the case. By performing a multiview refinement, the geometry of the model and thus the visual appearance can be dramatically improved.

Initially, the position of each scan S_i relative to the reference scan S_0 is determined by two factors: First, the position of its preceding scan S_{i-1} and second the transformation $T_{i,i-1}$ to align S_i with S_{i-1}. As said, this leads to accumulated errors, and doesn't scale to long capturing sessions with a big number of scans. Additional pairwise registrations of non-successive scans yield more constraints for the position of each scan and thus decrease global drift. Ideally each scan could be aligned with all overlapping scans.

In a first step, pairs of scans that could potentially be aligned, have to be found. Informally spoken, we have to look for scans that have 'similar' transformations. Unfortunately, no distance measure for rigid transformations is commonly known. Instead we have successfully experimented with a simple measure that combines the euclidean distance of the cameras of two scans and the angle between their viewing directions.

In a second step, pairwise registration of such candidate pairs is attempted. If it succeeds, the resulting transformation can be used as an additional constraint for the position of the corresponding scans.

In the third step, the positions of all the scans can be recomputed based on all the constraints resulting from pairwise registrations.

We use a technique similar to [9], where the positions of all patches are refined iteratively taking into account the pairwise constraints. The algorithm is summarised below. For a detailed description we refer to the original paper. Initially `pend_queue` contains all the scans that are connected to multiple other scans via pairwise constraints.

Algorithm 1 Iterative refinement of all transformations

```
pend_queue ← all the scans with multiple connections
while pend_queue≠ ø do
    S ← pop(pend_queue)
    T_improvement ← adjust position of S using the constraints from all its pairwise registrations.
    if T_improvement > threshold then
        merge all scans that are connected to S into pend_queue
    end if
end while
```

3.3 Merging and Preview Rendering

One of the key features of our system is the ability to present the user a visual feedback about the current state of the scanning process in real-time. The task of this stage of the pipeline is to bring the quickly growing amount of partly redundant range data into a form that can be kept in memory, to incrementally update them when new data are available, and to render them on the screen.

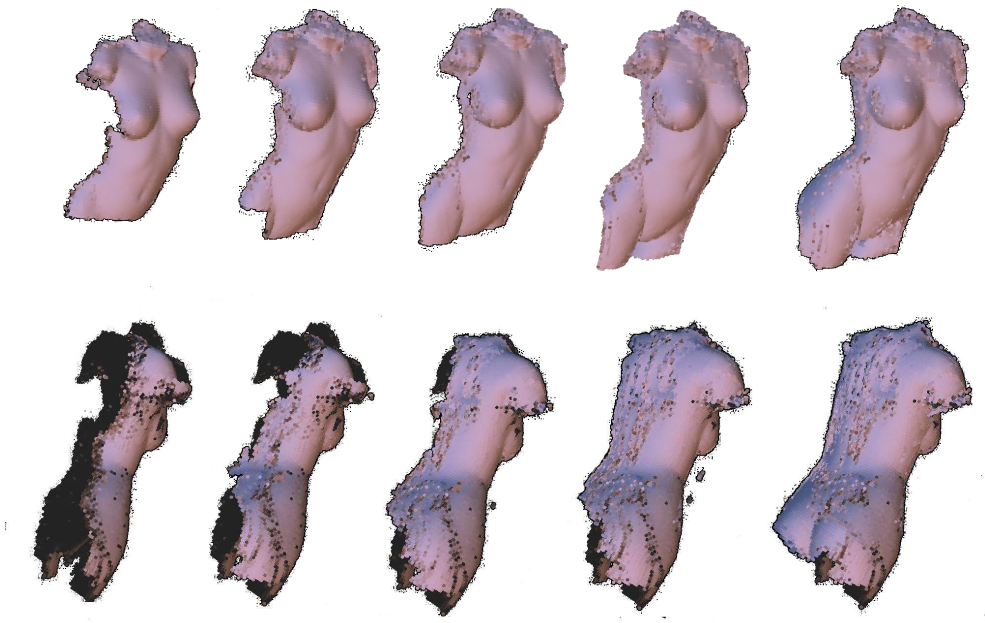


Figure 6: *Preview rendering of the front and (noisy) back of the model during the online scanning and registration process.*

Existing integration algorithms that produce a high-quality surface out of a number of input surfaces are computationally expensive [3, 13]. In our system, such an integration is only done in an offline postprocessing step.

Instead we simply merge the 3D points into the cells of a voxel-grid and render them directly rather than as a polygon mesh, similar to [12]. Only one point per voxel-cell is kept and a corresponding normal vector is computed as the average normal of all the 3D points that fall into the same voxel cell. For every occupied cell a circle is rendered with its color/greyscale determined by the average normal of the cell and the position of the virtual light sources. If each circle is scaled such that it overlaps with its neighbours, the illusion of a surface is created. This technique is referred to as *splatting* and is often used for high-resolution 3D models [11]. The rendering quality is sufficient as a preview of the model to determine visually if all the important parts of the object have been scanned.

The resolution of the voxel grid is set roughly equal to the sample density of 3D points. When the grid resolution is too high, points that actually belong to the same surface are likely to fall into different cells (due to misalignments or noise) and build multiple layers. Too low grid resolutions result in a loss of detail and general visual quality.

3.4 Offline Postprocessing

An online capturing session provides us with a number of range and texture scans as well as corresponding transformations to bring them into a common coordinate system. The voxel grid has been created for visual feedback during capturing only and is discarded. The remaining last step to integrate the data into a 3D polygonal mesh model can now be performed without user intervention. Optionally a high-accuracy registration algorithm can be run prior to the integration.

The volumetric integration method by Curless et al.[3] is well suited for our needs. The final result is a single triangular surface.

4. Results

Figure 6 shows screenshots of the 3D model of a mannequin being built up in realtime. This example sequence consists of approx. 380 scans. Part of the input data has been (automatically) discarded. Finally 246 scans have been used for the resulting model, each containing between 20000 and 30000 points. The online acquisition process took 60 s.

In figure 7 a preview rendering of the finished model and the final offline integration are shown. For the latter, a resolution of approx 40000 triangles has been chosen. No multiview refinement was performed, an error which is due to accumulated misalignments is visible on the shoulder of the mannequin.

Currently our system runs on a Pentium4 2.26 GHz at ca. 5 frames per second. There is potential for further speed up, as no optimisation toward the highest possible throughput has been done yet.

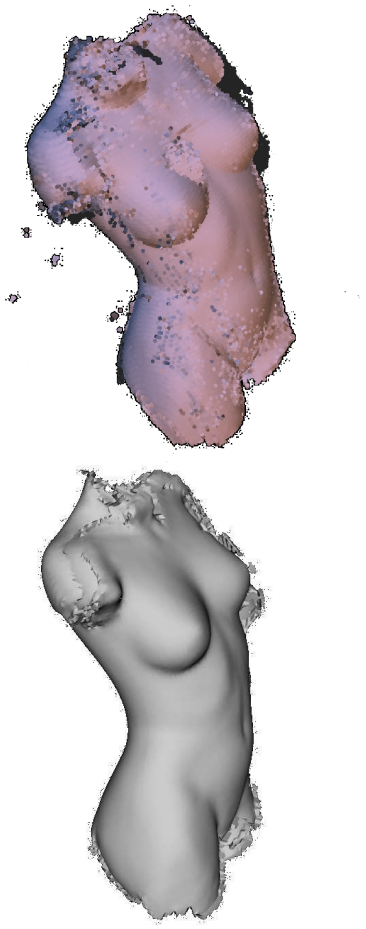


Figure 7: *Preview rendering and final integration of the whole model. On the shoulder of the mannequin the accumulated errors are visible.*

5. Conclusion

A system to create 3D models of real-world objects was presented. During the range acquisition process a preview of the integrated model is shown on the screen. This visual feedback allows for intuitive checking of the quality and completeness of the scanned data. The output of the online system are textured range images and corresponding registration information. The final integration is done in an offline postprocessing step. Preliminary results have been presented.

Future work will include texture in the registration and integration modules. The ability to recover when the system loses track of the registration will considerably improve the robustness of the online scanning. We will also look for methods which can more easily deal with the hands of the user who is manipulating the object.

Acknowledgments

The authors gratefully acknowledge support by the Un. Leuven via the project GOA 'VHS+'. Thomas Koninckx gratefully acknowledges support by a grant of the Flemish Institute for the Advancement of Science in Industry 'IWT'. Thanks to Joris Vanden Wyngaerd for his implementation of the volumetric integration.

References

[1] P.J Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Patt. Anal. Machine Intell.*, 14(2):239-256, February 1992.

[2] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*10(3):145-155, April 1992.

[3] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. SIGGRAPH '96, *Computer Graphics Proceedings* 1996.

[4] Eyetronics, *www.eyetronics.com*

[5] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*,4(4):629-642, April 1987.

[6] InSpeck Inc., *www.inspeck.com*

[7] T.P.Koninckx and L. Van Gool, High-speed active 3D acquisition based in a pattern-specific mesh, proc. 'SPIE: EI Photometrics', pp.26-37 Januari 2003 USA.

[8] T.P.Koninckx, A. Griesser and L. Van Gool, Real-time Range Scanning of Deformable Surfaces by Adaptively Coded Structured Light, 3DIM03, October 2003 Canada.

[9] K. Pulli, Multiview Registration for Large Data Sets, Int.Conf. on 3D Digital Imaging and Modeling, Ottawa, pp.160-168, 1999.

[10] S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm, Proc. ICCV 2001.

[11] S. Rusinkiewicz and M.Levoy. QSplat: A Multiresolution Point Rendering System for Large Meshes. SIGGRAPH 2000.

[12] S. Rusinkiewicz, O. Hall-Holt and M. Levoy. Real-Time 3D Model Acquisition, SIGGRAPH 2002.

[13] G. Turk and M.Levoy. Zippered Polygon Meshes from Range Images. In SIGGRAPH '94, pages 311-318, July 1994.

[14] J. Vanden Wyngaerd and L. Van Gool. Automatic Crude Patch Registration: Toward Automatic 3D Model Building. *Computer Vision and Understanding* 87, 8-26 (2002).