SLStudio: Open-Source Framework for Real-Time Structured Light

Jakob Wilm¹, Oline V. Olesen¹ and Rasmus Larsen¹
¹ Department of Applied Mathematics and Computer Science, Technical University of Denmark, Matematiktorvet, Building 321, DK-2800 Kgs. Lyngby, Denmark
e-mail: jakw@dtu.dk
² Department of Clinical Physiology, Nuclear Medicine & PET, Rigshospitalet, Copenhagen University Hospital, Denmark

Abstract—An open-source framework for real-time structured light is presented. It is called “SLStudio”, and enables real-time capture of metric depth images. The framework is modular, and extensible to support new algorithms for scene encoding/decoding, triangulation, and acquisition hardware. It is the aim that this software makes real-time 3D scene capture more widely accessible and serves as a foundation for new structured light scanners operating in real-time, e.g. 20 depth images per second and more. The use cases for such scanners are plentyful, however due to the computational constraints, all public implementations so far are limited to offline processing. With “SLStudio”, we are making a platform available which enables researchers from many different fields to build application specific real time 3D scanners.

The software is hosted at http://compute.dtu.dk/~jakw/slstudio.

Keywords—Images acquisition systems and information extraction, Image processing tools, Image and video processing, computer vision

I. INTRODUCTION

The advent of inexpensive consumer depth sensors has lead to much interesting research in the fields of human computer interaction, 3D scene understanding, pose estimation and others. These new devices are used for a plethora of applications, ranging from scene capture and understanding to object digitization, motion tracking, human-computer interaction, and many others.

Current devices do however present different trade-offs between spatial and temporal resolution and system geometry, cost, and features. Many new applications call for a flexible device, which can be custom built and optimized for the problem at hand.

Currently, few software packages are publicly available for structured light, and to our knowledge, this is the first open-source software with real-time performance. The hardware components involved are rather inexpensive, such that one can implement a megapixel resolution, 20 Hz, device for under 2000 USD.

The software framework presented here is highly customizable to work with cameras and projectors of different manufacturers, and supports a palette of scene encoding/decoding strategies. The software, referred to as “SLStudio” is available open-source, encouraging active community engagement.

We developed SLStudio for motion tracking and correction in medical imaging [4], but the software has wide applicability, including and extending those applications which currently use off-the-shelf devices such as Microsoft Kinect, ASUS Xtion and others.

In recent years, a number of real time dense scene acquisition methods have emerged. Time of Flight work on the principle of measuring the time delay between emission and receipt of light pulses or oscillations. While these sensors provide fast update rates, they suffer from high noise and bias [2], making them unsuitable for applications that require high accuracy.

Triangulation based methods require two viewpoints of a scene. In the passive case, these would constitute two cameras, and depth calculation is based on estimating disparity between the views. These methods most often require long processing time, however real-time implementations do exist, see e.g. [7] and [9]. The accuracy of these methods is limited, and especially weak in homogenous image regions.

Active triangulation methods employ laser light or projected patterns in order to add texture to the scene, and improve the accuracy and robustness of triangulation. The structured light concept is especially versatile in that different projected pattern sequences provide trade-offs between accuracy, robustness and update frequency. Single-shot methods employ a static pattern, and rely on neighborhood information for decoding, hence limiting their resolution. For direct coding of depth calculation is based on estimating disparity between the views. These methods most often require long processing time, however real-time implementations do exist, see e.g. [7] and [9]. The accuracy of these methods is limited, and especially weak in homogenous image regions.

II. STRUCTURED LIGHT PRINCIPLE

The fundamental principle underlying structured light methods is triangulation of corresponding points for a projector-camera pair. This is shown schematically in figure 1. A single camera is sufficient for scene acquisition, in that the projector acts as a second viewpoint. The distinct advantage over two-camera stereo is that texture is added by means of the projector light, which aids in the determination of correspondences.
Having control over the projected output pattern allows for direct coding of projector coordinates into a series of camera frames. Using an appropriate decoding method, projector coordinates can then be determined in the camera’s view. Finally, the triangulation principle is employed to extract spatial coordinates, yielding 2.5D depth maps, possibly with texture information.

A calibration is usually performed prior to scanning, as this relates pixel coordinates to real world distances, and allows for metric reconstructions of the scene. In the calibration procedure, the optical properties of the projector and camera are determined, usually aided by a calibration target such as a black/white checkerboard with known spatial dimensions.

The steps involved in obtaining a single depth map are the following:

1) Scene encoding in which one or both projector coordinates are encoded into a sequence of patterns, which are then projected onto the scene in succession. Pattern strategies include Phase Shifting Profilometry (PSP), binary Gray codes, color coding, and many others. A review of such methods is given in [1].

2) Frame capture. One camera frame is acquired for each projected pattern. We refer to one set of camera frames as a frame sequence.

3) Decoding the frame sequence by means of an algorithm that matches and reverses the action of the pattern encoder. This determines for every camera pixel the corresponding projector coordinates. From this step, an intensity or shading image can usually be calculated, which provides color or gray-scale texture for point cloud rendering.

4) Triangulation, which is the process of turning the decoded projector coordinates into a metric 2.5D surface represented by points. Geometrically, this step corresponds to triangulation of points by intersecting corresponding rays originating from camera and projector. If only one projector coordinate was coded into the frame sequence, the intersection may be found as the solution to a linear set of three equations. With two coordinates encoded, the system is overdetermined, and can be solved using different triangulation algorithms. This reconstruction of points may be followed by further processing, i.e. surface reconstruction, in which a mesh representation is computed.

III. IMPLEMENTATION

“SLStudio” is written in C++ and makes use of OpenCV and Point Cloud Library for most of its computations, while the Qt framework is used for the graphical user interface, and multi-core processing. Figure 2 shows screenshots from within the running program. The main program window contains a 3D point cloud display and a histogram, to aid in avoiding over- or underexposure, and additionally, the decoded up map and texture information.

The software is comprised of the following modules, all implemented as abstract C++ classes, with a number of concrete implementations:

**Calibrator** is a generic interface to calibration methods. A number of patterns are provided, and the corresponding camera frames are then used to determine the intrinsics of the camera and projector lenses, and the geometric relation between the two. Point cloud are expressed in the camera’s coordinate system after calibration. We provide one calibration method, which is highly accurate and very convenient. It captures frame sequences of a planar checker board using PSP and uses radial basis functions to translate world point coordinates into projector space. Projector lens distortion is determined and corrected for. Details are found in [6].

**Projector** models the interface for pattern projection. It features a direct projection mode in which the pattern is provided directly as a block of memory, and a deferred mode, in which patterns may be buffered for later display. The latter is convenient in the default OpenGL implementation, which allows patterns to be uploaded to GPU texture memory for fast display with vertical synchronization. This implementation has custom, compile-time determined versions for Linux X.org,
Windows and Mac OS X, to create a second fullscreen context, that does not interfere with the main screen, which displays the control GUI.

The **Camera** class implements communication with a camera. Two distinct modes are supported; software trigger and hardware trigger. In many industrial cameras, hardware triggering enables fast frame acquisition by avoiding the overhead of initiating exposure through a software API, and having projector and camera exposure synchronized perfectly. Attention must be paid to exposure times – for instance, in DLP (Digital Light Projection, Texas Instruments Inc.), projected grayscale values will only be faithfully represented if the camera exposure is a multiple of the single pattern exposure time. For hardware triggering, a short electrical trigger pulse is sent to the camera at the start of each projection. For commercial projectors, a compatible trigger source is the analogue VSYNC signal found in VGA and DVI cables. The class is implemented as an object “factory”, to support multiple cameras through the same interface. Concrete implementations exist for the camera APIs of IDS Imaging, Point Grey Research Inc., XIMEA, and the libdc1394 library for most industrial Firewire cameras.

A **Codec** is comprised of matching **Encoder** and **Decoder** classes, which have concrete implementations for different pattern strategies. A distinction is made between coding in one or two directions. While one is sufficient for triangulation, two directions are used where high accuracy is mandatory. The output of any decoding operation is termed \( u_p \) or \( v_p \) map, as it represent for every camera pixel coordinate the corresponding horizontal or vertical coordinate in the projector output. Implementations include Phase Shifting Profilometry (PSP) with and without phase unwrapping and binary Gray coding. The highest point cloud update frequency is achieved with 3-step PSP, which requires only three patterns for scene encoding.

The **Triangulator** is the class involved in triangulation of points using predetermined camera and projector parameters and the \( u_p \) and/or \( v_p \) map obtained from decoding a frame sequence. We have implemented fast triangulation based on \( u_p \) or \( v_p \) using a pre-computed determinant-tensor\[^5\], and algebraic triangulation. SLStudio represents point clouds internally in the PCL format, which can be written to disk or used for further processing.

Our software is modular and easily extensible to fit the specific requirements for a depth measurement device. The preferences dialog, shown in figure 3 shows the configurable options currently available.

The program employs task-based multithreading to share work among the CPU cores available in a machine. The threading structure is illustrated in figure 4. A natural division of work is employed, which lets one thread handle the GUI and graphical display, one thread the camera/projector calls, while a third is responsible for frame sequence decoding and a fourth for point cloud reconstruction. Practically, work is performed
on heap-allocated arrays, and thread-communication consists of passing smart pointers to these to other threads for further processing.

IV. CONCLUSION

In conclusion, we have introduced a feature-rich software framework for real-time structured light. In our use case, it enables dense scene acquisition with over 300,000 points at 20 Hz, and allows other users to build custom low cost depth cameras with real-time processing ability. It also provides a platform for research in structured light by allowing users to implement new pattern coding strategies. The software is available open-source, and it is our hope that the community finds it useful, and participates in the continued development of it.

REFERENCES