Konsta Hölttä

Non-static Object Capture Using Multi-view Stereo Video

School of Electrical Engineering

Seminar report
Espoo 18.4.2014

Seminar report supervisor:

Prof. Ville Kyrki

Aalto University
School of Electrical
Engineering

Contents

1 Introduction

Computer vision is a mature field; the steps of acquiring three-dimensional structure of a real-life scene are well known. Multiple view geometry (or multi-view stereo) is a method to interpret depth in camera images and thus to acquire 3D scans. [1–4] For detailed scanning dealing arbitrary reconstruction types, there exist a wide range of algorithms, each suited for different setups. [5].

Stereo vision and the related motion capture are well studied fields. 3D scanning and reconstruction has been successfully done on single objects and bigger scenes accurately with the help of ever increasing computing power [6–8], and mocap in coarser form is an ubiquitous tool in the film industry [9]. Microsoft Photosynth [10], for example, is able to take an arbitrary collection of photos from the internet and reconstruct three-dimensional models of publicly photographed targets. Users can also upload their own collections. Another web-based tool is Autodesk 123D Catch [11], which is popular among hobbyists. It reconstructs three-dimensional models from photographs with no configuration, but with a restricted accuracy.

Using similar principles as the human eyes to calculate point disparity and depth of individual pixels from photos, 3D point clouds can be constructed, given only a set of two-dimensional images taken of a same target from different poses and and a few camera parameters. [1] This technique, also called photogrammetry, has applications in many fields ranging from mapping of larger scenes to scanning of individual objects. The principles are mature; current state of computing power and quality cameras has introduced lots of progress in automatic software tools.

Three-dimensional structure can be scanned with other means too, such as laser range finders [12] or structured light [13]; in this seminar work, the focus is kept on stereo-based computer vision.

When 3D scanning is extended to take into account temporal changes in geometry or appearance (color), more complex hardware and calculations are needed in order to cope with the changing data. Non-static cases need a larger set of cameras set up so that the captured target can move while the geometry is imaged from several different directions. Applications include for example performance capture for video games or movies [14], cloth deformation capture [15] and aerial heritage mapping [16]. A related field in robotics is simultaneous localization and mapping (SLAM), where the surroundings are mostly static but the camera moves in an unknown environment. [17]

For there are several methods on the topic, choices must be made when implementing the reconstruction setup. The length of this seminar work cannot cover all the details; the common initial steps are described, sometimes referring to more detailed papers for details. The bottom-up structure of this work is divided as follows: the first part presents the basic geometric principles behind most implementations on stereo vision, starting from grabbing images of real-life scenes, finally extending to multi-view stereo. Issues in the dynamic case are discussed when applicable. Then, the next chapter focuses on what should be done when tracking the dynamic actions of a scanned target. Finally, the current state of software tools and applications are presented. The last chapter summarizes the methods.

2 Imaging

Digital stereo vision in the end analyses digital images; this section introduces the basic image acquisition steps and concerns that affect reconstruction quality. Cameras are never ideal, and practical algorithms take lens imperfections into account (e.g. [18]). Images are commonly taken with digital cameras that project a three-dimensional view from a single viewpoint to an image plane, and finally to a discrete grid of numerical values that describe light intensities.

Practical details, such as depth of field, sharpness, aberrations and others are not considered, as they vary greatly depending on the used hardware and are out of scope of this work. It suffices to say that in a practical system the choice of good optics is a key to good quality reconstruction. Accuracy and errors depend on not only decidable physical parameters of a stereo imaging rig, such as camera positioning, but also on e.g. physical construction errors, lens imperfections, camera sensor noise, image compression and algorithmic accuracy. [19–21].

2.1 Pinhole camera

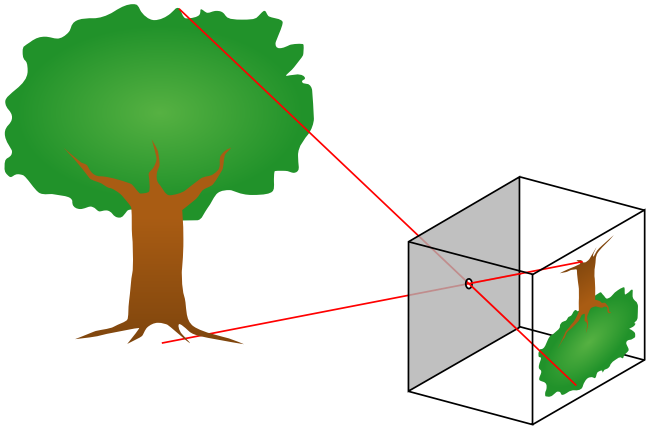

Figure 1: Pinhole camera principle. The box represents a camera; image seen through the small hole is formed to the plane on its back, rotated upside down.

A physical camera is in its simplest form modeled as a pinhole camera; an ideal device that projects an image upside down on its film through a small aperture. Illustration given in image 1. In computer vision, this projection is given as a 3×4 matrix, when homogeneous coordinates are used. Homogeneous coordinates add an extra dimension to the interpretation of coordinates and each point becomes a line that crosses the origin in a dimension one higher than the original. In addition, several vector operations become more convenient to manipulate. [1, 4]

The pinhole model (or, perspective projection) states that the world point (x, y, z)

is projected to the image plane (f units away from the origin) at (u, v):

$$\begin{pmatrix} u \\ v \end{pmatrix} = -\frac{f}{z} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{1}$$

Light rays travel through the pinhole camera's aperture to the image plane that is f units behind the pinhole. The result can be derived from similar triangles with a common vertex at the aperture. Sometimes the sign is inverted, which results in a plane between the pinhole (i.e. camera origin) and the actual point, where the image is not rotated; this can be more convenient to analyse. [1]

Setting the camera to origin and using homogeneous coordinates, the mapping is given with a camera matrix as

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z/f \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2}$$

The camera position and rotation in a global coordinate frame can be encoded in a matrix so that the point (x, y, z) in global coordinate frame is first transformed relative to the camera's origin; section 3.1 discusses this in more detail.

2.2 Optics

In practice, no actual camera works ideally; imperfections in the lenses project points to positions that differ from those predicted by straight lines in this linear case. Lens distortions deviate the rays, and no system is in perfect focus, so that one light ray spreads out as a circle. In reconstructing, methods that estimate the points and minimize errors are used, as no model predicts the camera perfectly.

Construction of optical systems is well studied. [22] Actual camera lenses consist of not only a single glass element but many, especially in the case of zoom lenses. In this work, the inner workings of these systems are ignored and equations assume a simple projective model, which is a safe assumption when the image is in focus.

The following equation applies for a thin lens while capturing sharp images:

$$\frac{1}{a} + \frac{1}{b} = \frac{1}{f} \tag{3}$$

where f is the focal length of the lens, a is the distance between the lens and the film, and b is the distance between the lens and the imaged source.

All practical optical systems (lenses) introduce some non-linear distortion that affects the performance of the ideal pinhole model. Common distortions are the purely radial so-called barrel and pincushion distortions, where the magnification is a nonlinear function of image ray distance from the center of the lens. Fisheye lenses are commonly known to have this kind of effect. Tangential distortion is less common, particularly in great magnitudes, and is often ignored. Its cause is small

misalignments in separate elements in a single optical system; lenses being offset from each other and not parallel to the image plane. [22]

Wilson [23] discusses optical systems' relation to depth of field, focus and distortions.

It should be noted that the nonlinear optical distortions are different from the inevitable perspective projection distortion that happens when projecting a 3D scene to a 2D plane, which is taken into account in the reconstruction.

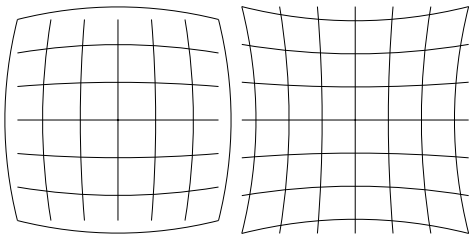

Figure 2: Barrel (left) and pincushion distortions that would show up in an image of a grid of straight lines. For a lens with no distortion, the lines would not be curved.

Distortion should be corrected in software, as the following stereo algorithms assume that the images are free of nonlinear errors, i.e. straight lines in the world should remain straight in 2D images after the projective transformation.

The radial correction used by the OpenCV library to create a new image of the original pixel values at new positions [18] is

$$x_{corr} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{4}$$
$$y_{corr} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{5}$$

Trucco and Verri [3] use only the two first coefficients. For tangential distortion:

$$x_{corr} = x + (2p_1 xy + p_2(r^2 + 2x^2)) \tag{6}$$
$$y_{corr} = y + (2p_2 xy + p_1(r^2 + 2y^2)) \tag{7}$$

x and y are the original coordinates in the distorted image, x_{corr} and y_{corr} are the corrected ones, k_1, k_2, k_3, p_1 and p_2 are coefficients specific to the distortion, and r equals to the distance to image center located at $(x_c,\ y_c)$:

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2} \tag{8}$$

2.3 Shutter

In dynamic (i.e. moving) environments, the process of acquiring several images consecutively is an important thing to consider. When a group of cameras are capturing

the same target, they should operate synchronously and grab images at same infinitesimally small points in time. In reality, there is some error: the cameras do not work in perfect sync, and their sensors take some time to acquire an image.

Moreover, a CCD sensor should be used in place of a chaper CMOS sensor; CCDs incorporate a "global shutter", i.e. the whole sensor images the scene at once. A phenomenon called "rolling shutter" is common in CMOS sensors: the reading happens linearly, and moving objects get distorted when several pixels see the same object occurring at the same time. This should be avoided, but it can be also taken into account and fixed. [24, 25] Uniform lightning is also usually assumed: when comparing pixel intensities, some figures might not look the same in different brightnesses, which is one of many reasons to use cameras that do as little automatic improvements as possible.

2.4 Video

Humans perceive motion when a previously seen object is seen again in a nearby location or similar position. Current digital video technology encodes motion in a sequence of still images, usually displayed in constant rate. Three dimensional motion is usually no different: it is encoded as discrete poses in sequence. In order to do object capture in stereo, video material from two or more of cameras is used to initially capture a sequence of still photos.

When scanning a scene continuously, a camera grabs frames using the same principles as in photos, but does it in sequence, at a speed that is called frame rate. Another notable point from the capture viewpoint is the shutter speed; in movies, the shutter is often open deliberately so long that fast motion is blurred, because it is considered aesthetically pleasing to human eye; even though the motion is blurred, more information about the scene movement is encoded per frame than when grabbing infinitesimally short times. [23] For sharp images that are preferred in reconstruction, this is to be avoided.

Synchronizing all the cameras that shoot the same scene might not a trivial issue in practice, depending on the gear used. Professional grade cameras can be synced to a single clock generator, so that they all operate on the same frequency and phase. The same method is used when shooting with machine vision cameras that have external trigger input. This still leaves a small phase difference caused by unequal transmission paths from the clock generator. Synchronizable camcorders are very expensive, though, and consumer-grade hardware usually lacks all possibilities to properly sync frequency or phase, not to mention frequency drift or frame jitter. Clapperboard is a ubiquitous and simple way to sync video and audio, but it still leaves a maximum of half a frame lag between the camera sequences; this is illustrated in figure 3. When cameras open their shutter in a different time, they effectively shoot a different scene, breaking one of the most fundamental assumptions of stereo vision: that the images encode geometrically same objects. This can be compensated to some degree with optical flow [25].

Faster frame rate encodes information more often, which is preferable as longer distances of pixels of same objects are more difficult to match when tracking objects;

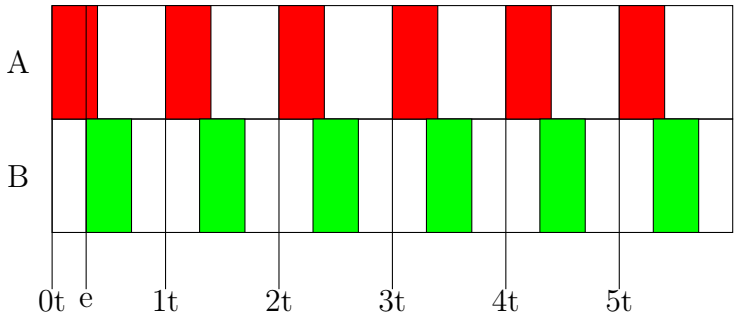

Figure 3: Video phase difference. Red rectangles illustrate exposure times of camera A, green rectangles the same for camera B. Frame period is t, and the cameras have a constant exposure time offset of an arbitrary e.

faster shutter speeds help to reduce motion blur. Fast shutter (i.e. short exposure) obviously needs to be compensated by using more sensitive sensors or more light to get equivalently bright images. Noise vs. motion blur is a common tradeoff that has to be made when building a stereo vision system.

Video recording and motion tracking are best considered orthogonal issues; while a single static case can be scanned in three dimensions, so can be also each frame of a sequence, separately. While this sounds tempting, it might not be computationally feasible, though, because the reconstruction must be started all over again for each frame and the topology is uncorrelated between the frames.

3 Stereo vision

Three-dimensional dynamic capture means in general the way of recording a sequence of movements of a real-life target or scene. This section introduces the methods for seeing three-dimensional structures from stereo setups, consisting of two or more cameras. The configuration of a proper setup with two is described; it can be extended to the multi-view domain. Most of the sophisticated algorithms able to recover a dense reconstruction base their work on the same principles, as follows.

3.1 Coordinate systems and transforms

The previous section described how to record a view of a scene with a camera. From now on, the term camera refers to a particular camera configuration, which can be a single physical camera moved to different locations.

The camera is a projective object located somewhere in the imaged scene. Its *intrinsic parameters* model the properties of projection, but do not take into account the camera location in any global coordinate system. The *extrinsic parameters* contain the camera location and rotation in another global coordinate system, structured as a matrix. This is especially advantageous when there are more than one cameras and their coordinates must be related. [1, 4] This part quickly reviews basic transforms whose results are needed in the later reconstruction steps.

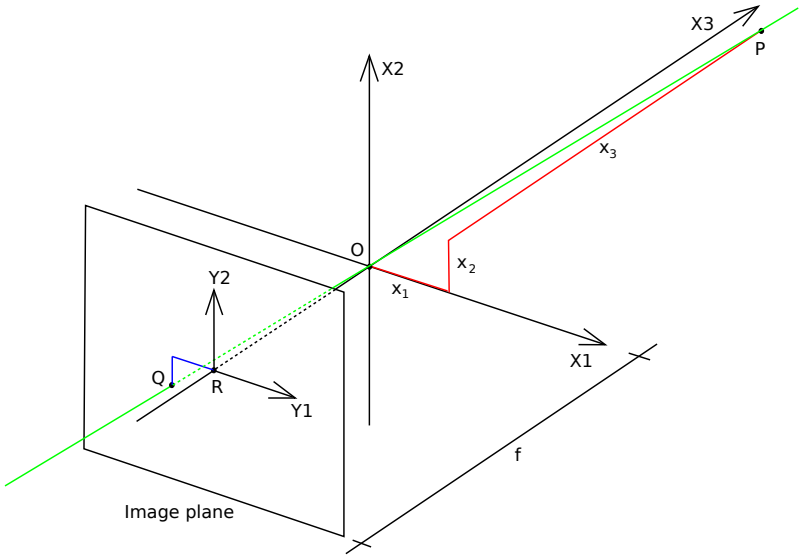

Figure 4: Pinhole camera geometry. Camera coordinate system origin at O, axis X3 points towards the optical axis, Y1 and Y2 point to image plane axes and R is the principal point, at the image center. The point P projects to Q, as well as everything else on the line joining them. The image plane is f units away from camera origin; f is called the focal length.

In computer graphics and vision, points and directions are usually described

in homogeneous coordinates. Translation, perspective projection, rotation, and other operations are conveniently described as matrices by using an additional dimension for points, of which usually the last element is 1: $(x, y, z, 1)$. All points (xw, yw, zw, w) map to the same point (x, y, z). [1, 26]

The imaging process essentially captures a projection to a flat two-dimensional plane of the camera's view, as described in section 2. When relating points between different cameras that view the same scene, the cameras' relational positions and rotations must be known. One of the cameras is often conveniently chosen as the origin of a global coordinate frame, so that its extrinsic parameters become unity transforms (programming libraries often assume this, see e.g. [18]). Each three-dimensional point in the world is transformed to the small sensor or film inside the camera, which is then digitized to a discrete two-dimensional grid of pixels. The size of this pixel array (i.e. image) is referred to as the camera's resolution.

The transformation chain is encoded as follows, given a homogeneous point (4-dimensional vector) X representing a 3D location described in physical (e.g. metric) coordinates:

$$\begin{aligned} x &= PX && (9)\\ &= M_i X_s && (10)\\ &= MTRX && (11)\\ &= M_i M_p TRX && (12)\\ & && (13) \end{aligned}$$

x is a 2d pixel in a discrete image, X_s exists on the sensor. R, T encode the camera rotation and translation (extrinsics); M_p projects the world coordinates to the camera sensor (film) - still in world coordinates (intrinsics!), and finally the affine M_i transforms the points from the sensor to pixel coordinates on the digital discretized image.

The whole projection $P = M_i M_p TR$ can be used as-is without decomposing it to separate matrices, unless the individual parameters are needed. As the chain consists of several matrices, some of them are defined only up to scale; the coordinate systems' units can be chosen freely. Software packages usually do not decompose the chain, because it is not needed and unique parameters cannot be found because of scaling.

The internal parameters, intrinsics, encode how the image is formed on the sensor: they consist of focal length, sensor size and principal point: (last column left out, as it's full of zeroes in 3x4)

$$M = \begin{pmatrix} m_x & \gamma & u_0 \\ 0 & m_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{14}$$

For simplicity, it is often denoted $\alpha_x = m_x f$, $\alpha_y = m_y f$. $R = (u_0, v_0)$ is the image center (or principal point). For square pixels, $m_x = m_y$, and for a non-skewed sensor, $\gamma = 0$, which is often the case. [1, 2, 4]

3.2 Camera calibration

Reconstruction algorithms need to relate points between images; the camera properties are needed. Calibrating a camera means to measure its intrinsics and extrinsics in order to map its data to a known coordinate frame. Calibration has always to be done, but it does not necessary need to be a manual step before scanning; self-calibration attempts to target this convenience problem. [1, 8]

Automatic calibration tools rely on an amount of feature pairs of which the best matches are found, or a known pattern, such as a planar checkerboard pattern [chuan; zhang] whose features are also distinguished with a similar algorithm but a priori knowledge of the object structure is used for precise calibration. These usually need several pictures taken with the same camera from different poses.

The checkerboard calibration step can also measure optical distortion at the same time. [18, 27]

One possible way is direct linear transform (DLT) [1]: the whole matrix P is solved from $x_i = PX_i$ by constructing a system of equations from the projections of some known points i, and minimizing an error metric, as the case is usually overconditioned.

3.3 Binocular disparity

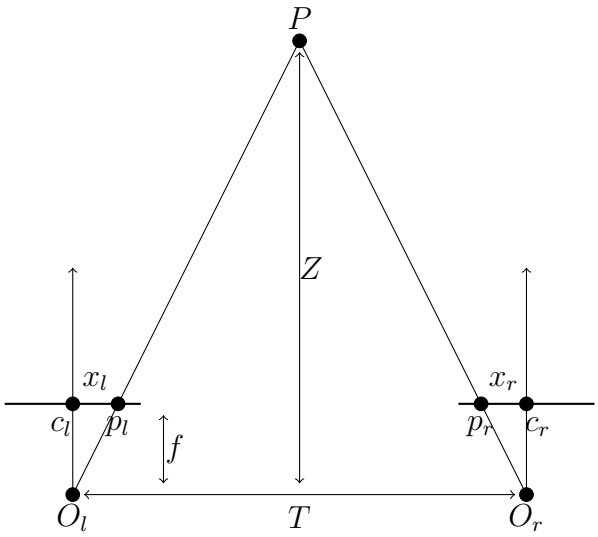

Figure 5: A very simple stereo setup, picture from above. The image planes (thick lines) are actually imaginary, as a real film in a camera would exist behind the principal point and project the image upside down, as described earlier in 2. The coordinates exist in the world coordinate units. The symbols O are the camera origins (T units between each other); c the principal points; x the image plane coordinates of p w.r.t. the principal points; and f is the focal length. The unknown is Z, depth of point P.

Assuming known calibration with identical cameras (same focal length and sensor) in a setup described above, visualized in figure 5, points can be triangulated as follows:

From similar triangles with a common vertex at P, we get (note that $x_r < 0$ as it's to the left, towards to the negative axis, from the corresponding plane's origin)

$$\frac{Z}{T} = \frac{Z - f}{T - x_l + x_r} \tag{15}$$

$$= \frac{Z - f}{T - d} \tag{16}$$

$$ZT - Zd = ZT - fT \tag{17}$$

$$Z = \frac{fT}{d} \tag{18}$$

The disparity d is the difference of the points in their image planes, $d = x_r - x_l$. If the image planes would be fixed as being physically correct, in the back side of the camera origins, the focal length should be negated to keep the correct interpretation and sign because the projected physical image is mirrored in both axes. Image processing between the sensor and a picture file usually inverts this.

As the equation 18 shows, depth is directly inversely proportional to disparity in this simple case. To map the depth to correct units, only focal length f and the baseline T are needed additionally; when using pixel coordinates instead of physical in d, also the pixel size should be taken into account. All of these are encoded in the camera parameters. Algorithms such as those in OpenCV [18] can compute point clouds from disparity images.

3.4 Epipolar geometry

Triangulation or reconstruction of the scene structure given by image pair(s) is usually done on the base of a known relationship between the cameras. Such relationship, known as calibrating the cameras, can be automatically determined, given by corresponding points that can be distinguished in each image and matched. [1, 3]

In stereo vision, the same scene of interest is seen by two or more cameras at the same time. The cameras are rarely aligned perfectly such as in the disparity setup described above, however. Epipolar geometry encodes the relations between arbitrarily positioned cameras in a standard way so that coordinates of a 3D point seen in several images can be calculated with the same triangulation.

A point seen by camera C_l at 3D point A could be anywhere on the line between C_l's origin and P, because a certain line passing through the principal point always projects to a point. This line is seen as a single point A_l. From another viewpoint in camera C_r, this line equals to some line on B's image plane. The real point must be on that line. The inverse applies for any point on C_r and a line on C_l. The lines on the image planes are called epipolar lines.

Essential matrix defines how the camera poses differ by the something something points seen by both. When A_l, A_r encode the points in figure 6 by the corresponding

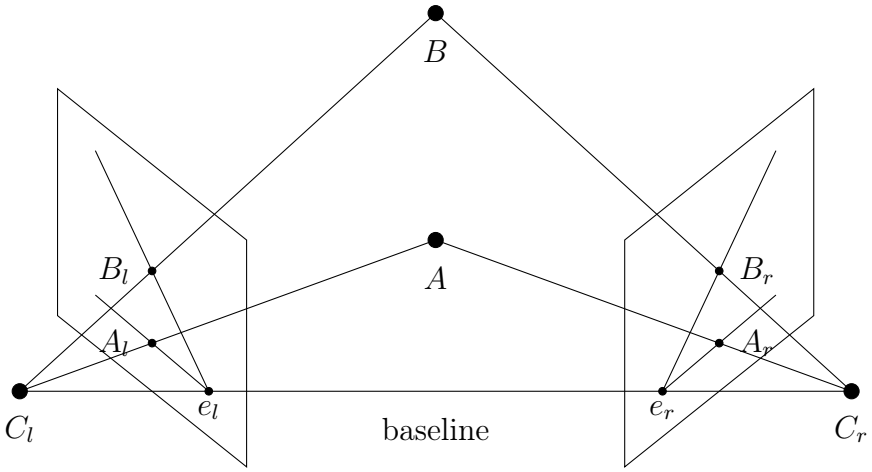

Figure 6: Two camera views on same scene. World points A, B project to planes of different views imaged from C_l and C_r on the left (A_l and B_l), and to the right (A_r, B_r). When A_l is known, its corresponding point A_r (not initially known in practice) is found on the epipolar line joining e_r and A_r in the right image. All epipolar lines in a view join in the same point (e_l and e_r).

camera coordinates, and the baseline difference (vector from C_l to C_r) is marked as t, it holds that $(A_l - C_l) \cdot t \times (A_r - C_r) = 0$, as all the vectors are coplanar; the cross product yields a normal to the plane, which is perpendicular to all the vectors, thus the dot product equals 0. [1]

Essential matrix is a matrix form of this relation; it includes the relative rotation and translation of the two cameras.

Fundamental matrix relates the corresponding points in stereo images; it has the same meaning as the essential matrix, but it works in the pixel coordinates of the cameras, which are obtained after the projective transform that takes the intrinsics into account. Inverting the matrix M_i (3.1) in sensor-to-pixel coordinate transform and using it on pixel coordinates, world coordinates seen by the camera can be obtained.

The fundamental matrix relates the pixels and epipolar lines, and as such it is useful in image processing where the images are described as pixels in a color array (image) and not colored physical coordinates.

3.5 Point matching

Previously, basics for reconstructing three-dimensional location for a point pair were introduced, assuming known positions for the same point in different images. To reconstruct a whole scene from a full image, all pairwise points must be matched, i.e. found that what pixel in one view represents the same object as one in other view.

Matching is often also called correspondence searching: Given a pixel in one image, what is the corresponding pixel in another image taken from the same scene?

Pixels correspond to each other if they represent the same physical point.

To describe the pixel's characteristics, its surrounding environment is encoded as a *feature*, a easily recognizable, unique property vector. When discussing about features, not every pixel's neighbourhoods are used; *good* features are those that have strongly distinguishable properties, such as edges and corners.

Edges or corners are essentially high-frequency information in the image that can be interpreted as a 2D discrete function; thus, they can be detected by a discrete high-pass or band-pass filter, zeroing all but those pixels where a high difference is found [28]

Neighbourhood of a pixel where features are detected is often called a window or a patch.

Matching can be done in sparse or dense mode; sparse matching finds a set of features from each image, and tries to match them. Dense matching runs through each pixel of one image and tries to find the same from another one with e.g. template matching [29]; from the coordinate differences in image space between the two images, a disparity map is built. The disparities are then directly transformed to depth values, yielding a point cloud.

Scale-invariant feature transform (SIFT) [30] is a commonly used algorithm for local feature detection. A GPU implementation is also available [31]. Invariance to scaling, translation and rotation makes SIFT useful in describing features that can be matched between unaligned images. Other similar commonly used means are Speede Up Robust Features (SURF) [32] Harris corner detector [33].

3.6 Correspondence and rectification

In order to triangulate a real point from two or more photographs, the location of the point in all images must be known. Rectification is a process that simplifies this search problem by restricting the search to a single dimension. By aligning the cameras such that their images are coplanar, the search only has to be performed on a line that is parallel to the line connecting the camera centers. After rectification, the corresponding lines are axis-aligned (horizontal or vertical) in both images. [1]

3.7 N-view stereo

The case of more cameras than a single pair uses the same principles in epipolar geometry. It brings more restrictions and dimensions; in three dimensions, for example, the fundamental matrix becomes three-dimensional, the trifocal tensor. [1] It is also more computationally heavy, as more data must be processed; if no information about camera parameters is available, pairwise checks between the images may become expensive. [34]

Multiple baseline stereo is a simple special case for many cameras. When all the cameras lie on the same baseline, calibration is easier and points can be selected by using a minimized sum of errors. [35]

The cameras that are used in capturing a scene can be fixed or positioned arbitrarily; in fact, the structure from motion technique [36, 37] enables to use just one

camera that is moved around. Accurate and fast reconstructions are still traditionally done with stereo camera pairs, though.

Another common way is to use pairwise cameras for individual reconstructions to build a point cloud for every pair, and then register them together. [14]

3.8 Reprojection errors

The quality of the reconstruction is measured by reprojecting the 3D points back to the cameras with the estimated parameters, and calculating the distance between the projected and original point. [1] Bundle adjustment [38] seeks to optimize all camera parameters at once with good performance.

A common way to handle feature errors is Random Sample Consensus (RANSAC). Random subsets of the sample space is iterated, and samples that do not fit well to a model that is constructed of a smaller set are ignored. The iteration that matches most samples is selected. [1]

4 Object tracking

Analogous to the case of traditional 2D video consisting of separate discrete frames of pixels, a dynamic stream of 3D data is, in a simple case, individual "frames" of point sets.

The dynamic case requires tracking of individual points or objects in order to usefully handle the moved object(s). Non-static human motion capture cases in video gaming or movies where post-processing time is available rely on manual work to perfect the quality. A realtime case would need to e.g. automatically register each frame between the previous one to use the geometry's dynamic properties.

The simplest tracking step is to leave tracking out completely: depending on the application, tracking might not be needed if the work done on the three-dimensional data does not need e.g. topological continuity in the time domain, but recomputes its work on each new point cloud.

Registration and tracking of point clouds or meshes is a large topic on its own; this section reviewes only some of the most common methods presented in the literature. Some work with the reconstructed 3D structure; others use the source images and do additional work.

Disorganised nature of the result data topology in frame-per-frame capture make it challenging to track individual points. In three dimensions, a template model is often scanned beforehand that is then morphed to match the target object to keep the topology (i.e. vertex neighborhood connectivity) constant. This technique adapts to both static and rigid cases. [39, 40]

Common method also for the entertainment industry is to use a pre-determined model of the scanned target or a completely separate character, and deform it on each frame based on the current state of the object, fitting the model's vertices to the scanned set.

Facial animation does not necessarily even need multi-view stereo for tracking and driving a predesigned character [41, 42].

4.1 Registration

When reconstructing an object all over again in consecutive frames, the points might not be fully aligned, and more importantly, they do not share the same model topology, i.e. the vertices have no other relation to the vertices in previous frames than being spatially near each other, accompanied with noise. [43]

Registration aligns disoriented models (e.g. point clouds, surfaces or triangular meshes, depending on the application) together with a rigid transformation between them so that one becomes the other from a reference coordinate frame's viewpoint. Aligning two geometrically *same* objects may not hold if the object deforms in a non-rigid way; when tracking the motion/deformation of a surface, the point sets describe a different geometry and there will always be some error.

A common method is Iterative Closest Point (ICP): iterative refinement approach to move the meshes closer to each other a small step at a time. When ICP is used locally, it can be applied to non-rigid cases too. [44]

4.2 2D features

In two dimensions, the full reconstruction step can be skipped when using only features in image space, providing real-time performance. [45] When assuming that a precise object stays in the cameras, features can be detected and tracked locally with no reprojection, while the tracked object is assumed to keep the same topology as a previously scanned model.

4.3 Optical flow

Motion in an image shows as changes in pixel color; it can be perceived as similar pixels flowing to the direction of movement in a pattern. A motion vector can be calculated for each pixel, resulting in a vector field; following and interpolating the movement is another way to detect three-dimensional movement. [46–48]

By combining the movement tracking to a mesh whose dynamics are modeled physically, motion can be estimated from a single image. [49]

5 Software tools

Now that the theoretical background has been described, this section reviews the current state of available tools on the reconstruction problem. There is a wide collection of libraries, open-source tools and commercial packages for both automatic reconstruction and generic mesh editing for post-processing the data.

5.1 Libraries

There exist several generic computer vision and geometry processing libraries, most common of them being probably OpenCV [18], Point Cloud Library (PCL) [50] and Computational Geometry Algorithms Library (CGAL) [51]. These are written in the C or C++ languages and they have bindings to several scripting languages too.

OpenCV contains a large set of tools for 2D image processing and extends also to camera calibration and 3D reprojection. It is probably the most commonly known and most used computer vision algorithm package.

PCL contains a set of algorithms for filtering, segmentation, registration, visualization, and more for working on data sets that consist of points that may share attributes such as colors and normals. CGAL's scope is in the same field, concentrating on a little more advanced topics.

5.2 Free software

While libraries can be used to write new tools, many pieces of software are readily available that perform the algorithms presented with some additional magic and are distributed in binary executables.

Camera Calibration Toolbox for Matlab, also included in OpenCV as a C port, is a more or less standard tool to computation of undistortion maps and intrinsic and extrinsic parameters with checkerboard images using non-linear optimization and homographies. [27]

Bundler is a bundle adjustment and structure-from-motion system for computing camera poses and sparse point clouds. [36]

SiftGPU, a SIFT implementation for graphics processing units. [31] It is built on Sift++ [52]; both are based on difference of gaussians, a method for edge detection [28]

Multicore (parallel) bundle adjustment PBA computes something something in a way that exploits the modern parallel nature of computer processors that have several computing cores. [38]

Patch-based, clustering multi-view stereopsis (PMVS/CMVS) starts from a set of matching keypoints (features) and expands them to a dense patch set iteratively. [53, 54]

VisualSFM [34] is a common and free but closed-source integration tool simplifying the workflow using external programs.

Meshlab is a portable editor of point clouds and meshes [55]. It can be used interactively and scripted to do the same steps automatically. In a reconstruction

pipeline, it is one of the last processing steps, used for removing outliers or fitting a surface on a point cloud, and finally projecting the textures to the generated mesh when given the camera parameters in relation to the point cloud pose. It can also perform registration between point clouds.

Screened poisson surface reconstruction, or PoissonRecon, is another alternative for surface fitting. [56]

Cmpmvs is a multi-view reconstruction software for transforming a set of calibrated image data to a full textured mesh, used in a similar way as VisualSFM but using internally a different method based on graph cuts and weakly-supported surfaces. [57]

Python Photogrammetry Toolbox is another pipeline combinator for full 3D reconstruction, popular in archaeological fields. It combines Bundler and PMVS and others. [58]

VisualSFM is probably the most common tool in the open source community. It integrates into a few clicks the pipeline from images to 3D point cloud, using SiftGPU for features, PBA for camera estimation, and PMVS/CMVS for dense matching. A common post step is to use Meshlab to filter outliers away from the data and to build a triangular textured mesh of it with the input points and normals.

5.3 Commercial

There is a large selection of commercial solutions available. Some companies are devoted to building software on certain applications or constructing whole scanning rigs. Examples include:

Autodesk 123D Catch is a free web-based application for automatic structureless reconstruction.

CaptiveMotion provides a facial capture and retargeting system that can be used with full-body mocap.

MotionScan is the technology behind the video game L.A. Noire. [59] It uses tens of cameras to recover detailed structure.

Faceshift encodes a markerless face captuer in a feature space that describes virtual muscle and bone movements.

3DF Zephyr Pro is another automatic photo reconstruction system.

Mova Contour Reality Capture does high-performance surface capture with a large array of cameras.

Pendulum Studio provides a capture system that integrates with game engines.

Pix4dMapper converts aerial images to geometric surface models.

Acute3d is targeted for large-scale photogrammetry and cultural heritage digitization.

6 Applications

The application domain for reconstructing three-dimensional fields is wide, and expands constantly with the ever increasing computing power. The popularity of handheld computers - smartphones - equipped with relatively good quality cameras is currently an attractive target for 3d scanning software, while the actual reconstruction works in the cloud. Kinect, the active infrared-based scanner for the xbox console, has reached a phase where it can be considered ubiquitous in gaming.

The Matrix (1999) [60], famous of the bullet-time scenes, used heavy optical flow processing to "slow time down"; it is a good example on what is needed to capture data that changes over time in non-controllable and non-repeatable ways. The Matrix Reloaded [61] used a technique called Universal Capture by Borshukov et al. [62] to encode and simulate accurate facial movements in 3D.

Facial surface motion capture (mocap) is currently a standard tool in the movie and video game entertainment industry among the more mature mocap for whole body movement to record performances for replaying them later. Mocap records movements of a human body or another object that is to be recorded so that the movements can be replayed or analyzed. Recent movies and video games (such as [59]) have shown that by capturing real motion of facial expressions instead of simulating them produces impressive results. Literature on capture of human faces and skin surface motions is large. [42] Professional quality capture still needs a laboratory environment with a large number of cameras [63, 64].

Human performance capture is traditionally done on special easily distinguishable markers, typically small retroreflective patterns, that are mapped to a model for playback. Playback then interpolates between the recorded positions. Advances in camera resolution make it possible to use only texture features without separate markers.

Cloth motion is another highly nonrigid application that can be scanned for acquisition of model parameters. [15]

Large scale static 3D scanning and texturing of archeologic sceneries is done with combining laser scanning and photographs [65].

Small objects have been scanned successfully with a turntable [37], and structure from motion techniques have recently been presented that impressively recover a structure from pictures taken all over an outdoor location with no a priori information about the camera configurations. [6, 7].

Static applications are more developed; while dynamic scanning has lots of interest in scanning faces and human motion, also static cases have received attention in medical applications, landscape/architecture engineering and mapping, and crime scene investigation.

7 Summary

Scanning real-life objects and scenes into three-dimensional computer models is an old area of research that is receiving increasing attention while computational processing power increases. Especially surface capture of nonrigid facial expressions or human actions is a popular topic; majority of current literature on dynamic 3D scanning deals with human performance.

This report introduced the basics of a multi-view stereo vision reconstruction pipeline for static and dynamic scans. The steps and concerns involved from taking pictures with plain cameras to reconstructing a three-dimensional geometry and appearance of objects were presented. Most reconstruction methods share the same geometric principles on multi-view geometry, but each have their own details in the final steps where a point cloud or topological mesh is produced; to study those in more detail, the cited work presents history and state-of-the-art methods. Care should be taken in each step to minimize reconstruction errors; good lighting and high-quality optics in well positioned cameras guarantees good input data to be processed by clever algorithms.

Dynamic scenes present challenges in every step. Methods for tracking, registration and remodeling of dynamic, non-rigid objects vary, using techniques from optical flow to iterative locally adaptive registration.

Different popular algorithms for reconstructing and tracking scenes or morphing objects were presented. One should consider the level of generality needed when constructing a 3D scanning rig; the more constraints and assumptions are set, better quality data can be accomplished. A static, fixed array of cameras is able to produce sub-millimeter data, while also a freely moving camera with no prior 3D scanning intentions can be used to recover geometric structure and colors. A number of currently available software tools were presented, and some hot applications were shown; the topic can be applied in several fields, and it is increasingly popular in not only experimental fields but also among consumers and hobby enthusiasts.

References

[1] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[2] Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2010.

[3] Emanuele Trucco and Alessandro Verri. *Introductory techniques for 3-D computer vision*, volume 201. Prentice Hall Englewood Cliffs, 1998.

[4] Anders Heyden and Marc Pollefeys. Multiple view geometry. *Emerging Topics in Computer Vision*, pages 45–107, 2005.

[5] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 519–528. IEEE, 2006.

[6] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. Multi-view stereo for community photo collections. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[7] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Towards internet-scale multi-view stereo. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1434–1441. IEEE, 2010.

[8] Marc Pollefeys, Reinhard Koch, Maarten Vergauwen, and Luc Van Gool. Handheld acquisition of 3d models with a video camera. In *3-D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on*, pages 14–23. IEEE, 1999.

[9] Thomas B Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006.

[10] Microsoft Corporation. Photosynth. http://photosynth.net. Accessed 2014-04-18.

[11] Autodesk Inc. 123d catch. http://www.123dapp.com/catch. Accessed 2014-04-18.

[12] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, et al. The digital michelangelo project: 3d scanning of large statues. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 131–144. ACM Press/Addison-Wesley Publishing Co., 2000.

[13] CMPPC Rocchini, Paulo Cignoni, Claudio Montani, Paolo Pingi, and Roberto Scopigno. A low cost 3d scanner based on structured light. In *Computer Graphics Forum*, volume 20, pages 299–308. Wiley Online Library, 2001.

[14] Derek Bradley, Wolfgang Heidrich, Tiberiu Popa, and Alla Sheffer. High resolution passive facial performance capture. *ACM Transactions on Graphics (TOG)*, 29(4):41, 2010.

[15] David Pritchard and Wolfgang Heidrich. Cloth motion capture. In *Computer Graphics Forum*, volume 22, pages 263–271. Wiley Online Library, 2003.

[16] Fabio Remondino. Heritage recording and 3d modeling with photogrammetry and 3d scanning. *Remote Sensing*, 3(6):1104–1138, 2011.

[17] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.

[18] Willow Garage and Itseez. Opencv. http://opencv.org. Accessed 2014-04-18.

[19] Fredrik Hollsten. The effect of image quality on the reconstruction of 3d geometry from photographs. 2013.

[20] Mikko Kytö, Mikko Nuutinen, and Pirkko Oittinen. Method for measuring stereo camera depth accuracy based on stereoscopic vision. In *IS&T/SPIE Electronic Imaging*, pages 78640I–78640I. International Society for Optics and Photonics, 2011.

[21] D Rieke-Zapp, W Tecklenburg, J Peipe, H Hastedt, and Claudia Haig. Evaluation of the geometric stability and the accuracy potential of digital cameras—comparing mechanical stabilisation versus parameterisation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(3):248–258, 2009.

[22] Rudolf Kingslake. *A history of the photographic lens*. Elsevier, 1989.

[23] Anton Wilson. *Anton Wilson's cinema workshop*. American Cinematographer, 2004.

[24] Omar Ait-Aider, Nicolas Andreff, Jean Marc Lavest, and Philippe Martinet. Simultaneous object pose and velocity computation using a single view from a rolling shutter camera. In *Computer Vision–ECCV 2006*, pages 56–68. Springer, 2006.

[25] Derek Bradley, Bradley Atcheson, Ivo Ihrke, and Wolfgang Heidrich. Synchronization and rolling shutter compensation for consumer video camera arrays. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 1–8. IEEE, 2009.

[26] Elan Dubrofsky. *Homography estimation*. PhD thesis, UNIVERSITY OF BRITISH COLUMBIA, 2009.

[27] Jean-Yves Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/. Accessed 2014-04-18.

[28] David Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[29] Richard O Duda, Peter E Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[30] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[31] Wu Changchang. Siftgpu: a gpu implementation of scale invariant feature transform (sift).

[32] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[33] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[34] Changchang Wu. Towards linear-time incremental structure from motion. In *3DTV-Conference, 2013 International Conference on*, pages 127–134. IEEE, 2013.

[35] Masatoshi Okutomi and Takeo Kanade. A multiple-baseline stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(4):353–363, 1993.

[36] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM transactions on graphics (TOG)*, 25(3):835–846, 2006.

[37] Andrew W Fitzgibbon, Geoff Cross, and Andrew Zisserman. Automatic 3d model construction for turn-table sequences. In *3D Structure from Multiple Images of Large-Scale Environments*, pages 155–170. Springer, 1998.

[38] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M Seitz. Multicore bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3057–3064. IEEE, 2011.

[39] Morten Bojsen-Hansen, Hao Li, and Chris Wojtan. Tracking surfaces with evolving topology. *ACM Trans. Graph.*, 31(4):53, 2012.

[40] Hao Li, Bart Adams, Leonidas J Guibas, and Mark Pauly. Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics (TOG)*, 28(5):175, 2009.

[41] Erika Chuang and Chris Bregler. Performance driven facial animation using blendshape interpolation. *Computer Science Technical Report, Stanford University*, 2(2):3, 2002.

[42] Zhigang Deng and Junyong Noh. Computer facial animation: A survey. In *Data-Driven 3D Facial Animation*, pages 1–28. Springer, 2007.

[43] Wenyi Zhao, David Nister, and Steve Hsu. Alignment of continuous video onto 3d point clouds. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1305–1318, 2005.

[44] Benedict J Brown and Szymon Rusinkiewicz. Global non-rigid alignment of 3-d scans. In *ACM Transactions on Graphics (TOG)*, volume 26, page 21. ACM, 2007.

[45] Julien Pilet, Vincent Lepetit, and Pascal Fua. Real-time nonrigid surface detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 822–828. IEEE, 2005.

[46] James J Gibson. The perception of the visual world. 1950.

[47] Berthold K Horn and Brian G Schunck. Determining optical flow. In *1981 Technical Symposium East*, pages 319–331. International Society for Optics and Photonics, 1981.

[48] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM Computing Surveys (CSUR)*, 27(3):433–466, 1995.

[49] Douglas DeCarlo and Dimitris Metaxas. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 231–238. IEEE, 1996.

[50] Willow Garage and Open Perception. Point cloud library. http://pointclouds.org. Accessed 2014-04-18.

[51] CGAL project. Computational geometry algorithms library. http://www.cgal.org. Accessed 2014-04-18.

[52] Andrea Vedaldi. Sift++ a lightweight c++ implementation of sift, 2011.

[53] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(8):1362–1376, 2010.

[54] Yasutaka Furukawa and Jean Ponce. Patch-based multi-view stereo software (pmvs-version 2). *PMVS2, University of Washington, Department of Computer Science and Engineering. Web. Downloaded from on May*, 14, 2012.

[55] ISTI-CNR. Meshlab. http://meshlab.sourceforge.net/. Accessed 2014-04-18.

[56] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):29, 2013.

[57] Michal Jancosek and Tomás Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3121–3128. IEEE, 2011.

[58] Pierre Moulon, Alessandro Bezzi, et al. Python photogrammetry toolbox: A free solution for three-dimensional documentation. *Python Photogrammetry Toolbox: A free solution for Three-Dimensional Documentation*, 2011.

[59] Team Bondi. *L.A. Noire [video game]*. Rockstar Games, 2011.

[60] Andy Wachowski and Lana Wachowski. *The Matrix [film]*. Warner Bros. Pictures, 1999.

[61] Andy Wachowski and Lana Wachowski. *The Matrix Reloaded [film]*. Warner Bros. Pictures, 2003.

[62] George Borshukov, Dan Piponi, Oystein Larsen, John P Lewis, and Christina Tempelaar-Lietz. Universal capture-image-based facial animation for the matrix reloaded. In *ACM Siggraph 2005 Courses*, page 16. ACM, 2005.

[63] RJ Winder, Tron Andre Darvann, Wesley McKnight, JDM Magee, and P Ramsay-Baggs. Technical validation of the di3d stereophotogrammetry surface imaging system. *British Journal of Oral and Maxillofacial Surgery*, 46(1):33–37, 2008.

[64] Depth Analysis. Motionscan. http://depthanalysis.com/motionscan/capture/. Accessed 2014-04-18.

[65] José Luis Lerma, Santiago Navarro, Miriam Cabrelles, and Valentín Villaverde. Terrestrial laser scanning and close range photogrammetry for 3d archaeological documentation: the upper palaeolithic cave of parpalló as a case study. *Journal of Archaeological Science*, 37(3):499–507, 2010.