Konsta Hölttä

Multi-view stereo reconstruction for computer graphics content capture

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 18.5.2015

Thesis supervisor:

Prof. Ville Kyrki

Thesis advisor:

Prof. Jaakko Lehtinen

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Konsta Hölttä

Title: Multi-view stereo reconstruction for computer graphics content capture

Date: 18.5.2015 Language: English Number of pages:9+114

Department of Electrical Engineering and Automation

Professorship: Automation technology Code: AS-84

Supervisor: Prof. Ville Kyrki

Advisor: Prof. Jaakko Lehtinen

Rendering of photorealistic models is becoming increasingly feasible and important in computer graphics. Due to the high amount of work in creating such models by hand, required by the need of high level of detail in geometry, colors, and animation, it is desirable to automate the model creation. This task is realised by recovering content from photographs that describe the modeled subject from multiple viewpoints. In this thesis, elementary theory on image acquisition and photograph-based reconstruction of geometry and colors is studied and recent research and software for graphics content reconstruction are reviewed. Based on the studied background, a rig is built as a grid of nine off-the-shelf digital cameras, a custom remote shutter trigger, and supporting software. The purpose of the rig is to capture high-quality photographs and video in a reliable and practical manner, to be processed in multi-view reconstruction programs.
The camera rig was configured to shoot small static subjects for experiments. The resulting photos and video were then processed directly for the subject's geometry and color, with little or no image enhancements done. Two typical reconstruction software pipelines were described and tested. The rig was found to perform well for several subjects with little subject-specific tuning; issues in the setup were analyzed and further improvements suggested based on the captured test models. Image quality of the cameras was found to be excellent for the task, and most problems arose from uneven lighting and practical issues. The developed rig was found to produce sub-millimeter scale accuracy in geometry and texture of subjects such as human faces. Future work was suggested for lighting, video synchronization and study of state-of-the-art image processing and reconstruction algorithms.

Keywords: multi-view stereo, reconstruction, 3D scanning, computer graphics, imaging

AALTO-YLIOPISTO SÄHKOTEKNIIKAN KORKEAKOULU	DIPLOMITYÖN TIIVISTELMÄ

Tekijä: Konsta Hölttä

Työn nimi: Monen näkymän stereon rekonstruktio tietokonegrafiikan sisällön taltiointiin

Päivämäärä: 18.5.2015 Kieli: Englanti Sivumäärä:9+114

Sähkötekniikan ja automaation laitos

Professuuri: Automaatiotekniikka Koodi: AS-84

Valvoja: Prof. Ville Kyrki

Ohjaaja: Prof. Jaakko Lehtinen

Fotorealististen mallien renderöinti on yhä tärkeämpää ja mahdollisempaa tietokonegrafiikassa. Näiden mallien luominen käsityönä on työlästä vaaditun korkean tarkkuuden takia geometrian, värien ja animaation osalta, ja onkin haluttavaa korvata käsityö automatiikalla. Automatisointi voidaan suorittaa taltioimalla sisältö valokuvista, jotka kuvastavat samaa kohdetta useammasta näkymästä. Tässä diplomityössä tarkastellaan kuvantamista ja valokuvapohjaisen geometrian rekonstruktiota geometrian ja värien kannalta ja katsastetaan viimeaikainen tutkimus ja ohjelmistot grafiikkasisällön rekonstruointiin. Taustan pohjalta rakennetaan laitteisto, joka koostuu yhdeksästä valmiina saatavilla olevasta digikamerasta aseteltuna hilaksi, itse tehdystä etälaukaisimesta sekä ohjelmistoista. Laitteiston tarkoituksena on taltioida luotettavalla ja käytännöllisellä tavalla korkealaatuisia valokuvia ja videokuvaa, joita voi käsitellä monen näkymän stereon rekonstruktion tietokonesovelluksissa.

Laitteisto säädettiin kuvaamaan pieniä staattisia kohteita kokeita varten. Tuloksena saadusta kuvamateriaalista laskettiin kohteen geometria ja värit ilman mainittavaa kuvanparannuksen käyttöä. Käytiin läpi kaksi tyypillistä rekonstruktio-ohjelmistoa testiksi ja laitteiston havaittiin soveltuvan hyvin useisiin kohteisiin ilman erityistä säätämistä. Kameroiden kuvanlaatu todettiin tehtävään erinomaiseksi ja useimmat haasteet johtuivat epätasaisesta valaistuksesta ja käytännön pulmista. Laitteiston todettiin tuottavan alle millimetriskaalan geometriaa ja pintavärikuvaa ihmiskasvojen kaltaisista kohteista. Jatkotyötä ehdotettiin valaistukseen, videon synkronointiin ja viimeisimpien kuvankäsittely- ja rekonstruktioalgoritmien tutkimiseen.

Avainsanat: monen näkymän stereo, rekonstruktio, 3D-skannaus, tietokonegrafiikka, kuvantaminen

Preface

This thesis is the first step in an adventure that started from a need and interest for an end-to-end machinery for acquiring realistic 3D content for computer graphics research and content production in Aalto University. The work was conducted in the Department of Media Technology at Aalto University, with help from the Automation Technology laboratory. My personal field of study and greatest interest is Automation and Systems Technology, where similar applications of computer vision can be found in robotics.

I would like to thank my advisor Jaakko Lehtinen for his guidance regarding this work and for the glorious opportunity to work on such an interesting subject, and my supervisor Ville Kyrki for valuable and supportive feedback on this thesis. Thanks too to my co-workers at the time, Markus Kettunen, Miika Aittala and Ari Silvennoinen, for inspiring discussions about computer graphics during this work.

Last but not least, thanks also to my family and friends for all the help and support. Of them, special greetings to Arto Rusanen and Samuli Vuorinen for sharing their deep knowledge in imaging technology and lending me photography hardware. Arto also kindly worked as a test subject.

Otaniemi, 18.5.2015

Konsta "sooda" K. Hölttä

Contents

Symbols and abbreviations

Symbols

In general, bold capital letters (e.g., $\boldsymbol{A}$) represent matrices, plain capital letters represent 3D points and corresponding lowercase letters matching 2D points (e.g., $\vec{P}$, $\vec{p}$). The arrow denotes a vector. Plain letters without the arrow represent scalars.

$\vec{C}$	camera position
f	focal length
κ_i	ith radial distortion coefficient
$\boldsymbol{K}$	camera intrinsic matrix
$\boldsymbol{M}$	projection matrix
$\boldsymbol{O}$	coordinate system origin
$\vec{p} = (p_x, p_y)$	pixel coordinates matching $\vec{P}$
$\vec{P} = (P_x, P_y, P_z)$	3D world coordinates
ρ_i	ith tangential distortion coefficient
r	radius
$\boldsymbol{R}$	rotation matrix
t	time
$\boldsymbol{T}$	translation matrix
$\vec{T}$	translation vector

Abbreviations

2D	two-dimensional
3D	three-dimensional
API	Application programming interface
APS-C	Advanced Photo System type-C
b	bit
B	byte
BRDF	Bidirectional reflectance distribution function
CCD	Charge-coupled device
CG	Computer graphics
CMOS	Complementary metal-oxide semiconductor
COC	Circle of confusion
CPU	Central processing unit
DOF	Depth of field
DSLR	Digital SLR
Exif	Exchangeable image file format
FPS	Frames per second
GCC	Gnu compiler collection
GPU	Graphics processing unit
I/O	Input/Output
IC	Integrated circuit
LED	Light-emitting diode
MCU	Microcontroller unit
MILC	Mirrorless interchangeable-lens camera
ML	Magic Lantern
Mocap	Motion capture
MP	Megapixels
MTF	Modulation transfer function
MV	Machine vision
MVS	Multi-view stereo
OpenGL	Open Graphics Library
PCB	Printed circuit board
PCL	Point Cloud Library
px	pixel
RAM	Random-access memory
RGB	Red-green-blue
SD	Secure Digital
SDK	Software development kit
SfM	Structure from motion
SIFT	Scale-invariant feature transform
SLR	Single-lens reflex camera (system camera)
UI	User interface
USB	Universal Serial Bus

1 Introduction

1.1 Background and motivation

The real world where we live in consists of visually three-dimensional (3D) objects. Photographs and drawings of it, on the other hand, are flat, only two-dimensional (2D) *projections*. Specifically, a camera takes a projection of a scene by capturing the light traveling around in the scene, originated from light sources, ending up to the flat film or digital image sensor. The result is a distorted view of the 3D dimensions and colors, depending on illumination and camera pose.

In *computer graphics*, a great interest is the digital *synthesis* of visual 3D content [1]. A geometric 3D model of a scene is given, and the light transport in it is synthesized to produce a flat projection from a single view, as a camera would show the scene. 3D computer graphics has numerous applications in video games, motion pictures, medical sciences, architectural engineering, archaeology, and more. Among most applications, content of high photorealistic detail is of interest and increasingly needed. The constantly increasing and already high level of processing power in computer hardware makes it possible to meet this requirement. Constructing realistic three-dimensional structures by hand is difficult and time-consuming, though. The goal of this thesis is to address this problem by using *3D reconstruction*: current state of digital cameras enables to create content directly by scanning the shape and color of real 3D objects using standard digital photography and computer vision. Scanning objects and scenes in real life for automatically building 3D content is becoming increasingly popular. In this thesis, such scanning methods are studied and a rig for performing photographic 3D scans is built.

The structure of 3D content is universally described as surfaces consisting of connected polygons, mapped with pictures to add high-detailed color. While computer graphics synthesizes images by drawing millions of polygons to produce a realistic image, *computer vision* tries to attain the inverse: to study images of 3D surroundings computationally for understanding the geometric structure in them. Among the applications of computer vision, reconstructing such 3D structure from two-dimensional images is an important one. 3D reconstruction naturally unites computer vision and computer graphics.

Multi-view stereo (MVS) [2] is a method for extracting the visible 3D geometry and color information of a subject, given several images of the subject taken from different angles (*views*). MVS can be applied to an unordered collection of pictures or to the imagery of a more strict set of calibrated cameras. A MVS scanning rig is a specially constructed machinery, consisting of a number of cameras and software, for shooting photos from different views and processing the photos using computer vision.

Plain *stereo vision*, MVS based on fixed two-camera views, has been long used in many applications varying from the robotics industry to 3D movie recording. Stereo vision compares the views of two cameras for depth cues, using similar principles as the human binocular vision. Multi-view stereo incorporates a multitude of cameras to acquire a more comprehensive and accurate view at the same time. The practice of acquiring multi-view stereo images and processing them for original 3D structure is called *3D scanning*, *visual 3D capture*, *3D reconstruction* or *photogrammetry.*

The need for *dynamic* content divides this particular application of 3D scanning in two types. *Static* subjects, i.e., stationary rigid bodies, present a simpler problem. Dynamic subjects, such as human limbs, skin, and clothing, move or deform over time. Static capture is simpler to accomplish, as the time when the pictures are taken does not affect the outcome; even a single moving camera can be used.

Human *motion capture* is one application of dynamic 3D reconstruction, with a goal to capture whole-body motion. Traditional ways to do motion capture use a set of bright markers attached on a special suit to recover only the motion of limbs over time, often further encoded as parameters of skeletal joint angles. The motion information is then reinterpreted for a 3D model to move its parts virtually. *Surface capture* refers to a more detailed scan of the motion of a complete surface, such as a human face or a cloth, possibly augmented on a skeletal motion capture data for a complete body. Capturing the movement in high resolution to scan the whole surface deformation and colors is a recent interest, requiring much more computational power than traditional motion capture. Advances in camera resolution and speed make it possible to use only surface texture details for tracking motion without separate markers. Recent movies and video games have shown that by capturing real motion of facial expressions instead of simulating them produces realistic high-quality results.

1.2 Goal and scope of the thesis

The objective of this thesis is to construct a complete scanning rig to be used as a part of an end-to-end pipeline for scanned 3D content generation, using multi-view stereo methods. The thesis is divided in two parts. First, the relevant theoretical background on photographic digital image acquisition and multi-view stereo reconstruction is surveyed. After that, a more practical part follows: the constructed system is described and test scans for both shape and texture information are presented. It should be noted that this work deals with well-known theory only and the theory is approached as is relevant to the goal of this work. Deeply studying state-of-the-art and developing new theory is only suggested for future work.

The research problem of "scanning an object visually in 3D" is formulated in such a way that proper hardware can be selected, and the quality and feasibility of the built system can be addressed. Performance of the reconstruction algorithms depends on the quality of their input. How an image is formed with a digital camera, what cameras are available in the market, what are the typical issues in the reconstruction process, and how can a good 2D input for a 3D reconstruction algorithm be achieved? To further assess and understand the resulting quality of the reconstruction, the fundamentals of the reconstruction algorithms are presented, and state-of-the-art MVS research is quickly surveyed for quality expectations.

A digital camera based 3D scanning rig is constructed, supported by software tools for acquiring both static image data and video files. The key hardware for such a rig consists of properly placed cameras, light sources, and a computer. Putting together all the parts for a general-purpose system needs studying of the end product as a whole. The camera synchronization and image acquisition are automated such that the output of the scanner can be fed directly to a reconstruction software without manual processing. The final result produced by the rig is a set of images or video, ready to be processed by reconstruction algorithms for producing 3D geometry with detailed colors, possibly animated over time. Feasible readily available programs for reconstructing 3D geometry and texture given the captured image data are surveyed and listed, and two popular workflows are tested. Basic background techniques for taking also motion into account are presented, but the use of state-of-the-art algorithms is left for future implementations.

The aim of the system is in hardware robustness, ease of practical use and general extendability. Because there are no detailed plans on all further use of the machinery, it is built to support many kinds of studies in the future without requiring the user to know all implementation issues in detail, and documenting the process in the form of this thesis. A special case subject in this work is human face: an interesting and challenging target because of its complex surface material and diverse ability to produce many different expressions.

The thesis is organised in seven sections, of which this introduction is the first. Section 2 presents the general theoretical background on the elementary methods in image acquisition using a digital camera and the issues in video recording. Relevant 3D reconstruction theory is described in Section 3, with focus on the basics that recent state-of-the-art builds on. Section 4 briefly extends the 3D topic by introducing issues on motion capture and reconstructing dynamic subjects. In Section 5, the implementation of the 3D scanning rig is described. Section 6 shows test experiments on the rig and discusses their results. Section 7 concludes the thesis with proposals on future work.

2 Image acquisition

This section describes the background theory for basic image acquisition steps in a camera and for main concerns that affect reconstruction quality. The major steps in an image acquisition pipeline inside a single camera are depicted in Figure 1 and are studied in this section at the necessary level of detail for the application. Quality of the outcome in a reconstruction pipeline depends on many factors of the used stereo imaging rig, such as camera positioning, lens imperfections, lens focusing, sensor noise, image compression, and algorithmic accuracy [3–5].

2.1 Computer images

In the digital world, with nearly no exceptions, images are stored and represented as *arrays of numbers*, i.e., in discrete form as opposed to the continuous data in the real world. The way of encoding images as discrete two-dimensional arrays is used throughout this thesis. A rectangular *raster image* consists of a regular grid of picture elements, or *pixels*. Each number describes the intensity value for the particular pixel, as depicted in Figure 2. [7, ch. 2.2]

Color images are described as several *color channels*: the intensities for different colors are given separately. The most common model is the *RGB* or *red-green-blue* model, where the primary colors are the same as in human eye; each pixel color is given as a triplet of these three colors. The intensity range may be arbitrary, but is typically a floating-point value or a small integer with zero indicating black and the maximum value indicating full color intensity. [8]

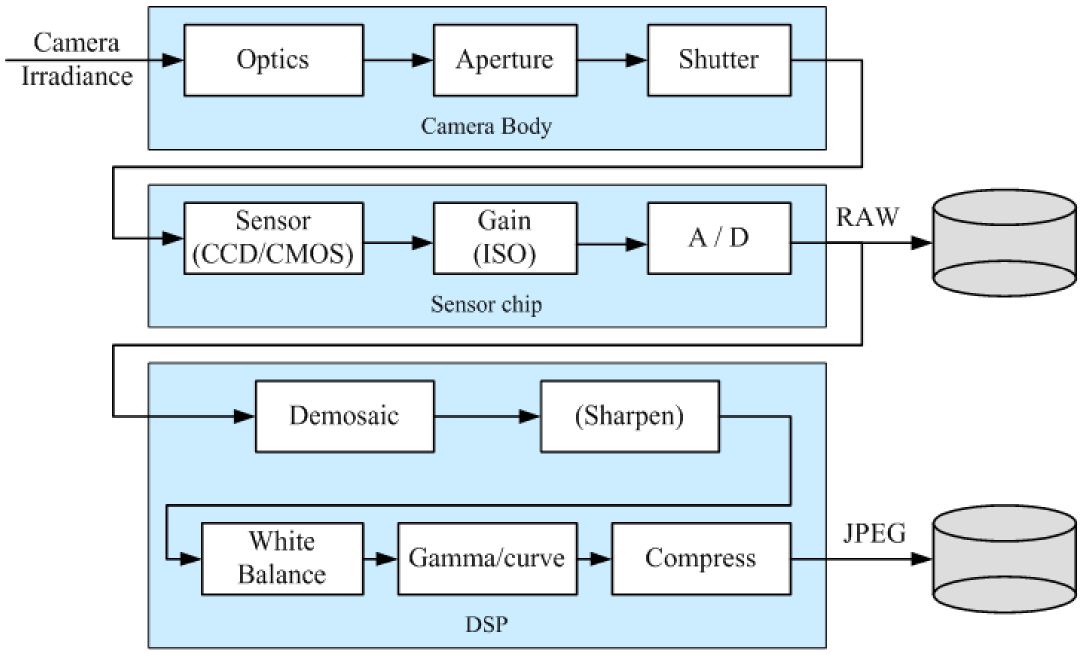


Figure 1: Image processing pipeline in a typical modern camera consists of several optical and digital steps with additional post-processing. Image from [6, p. 74].

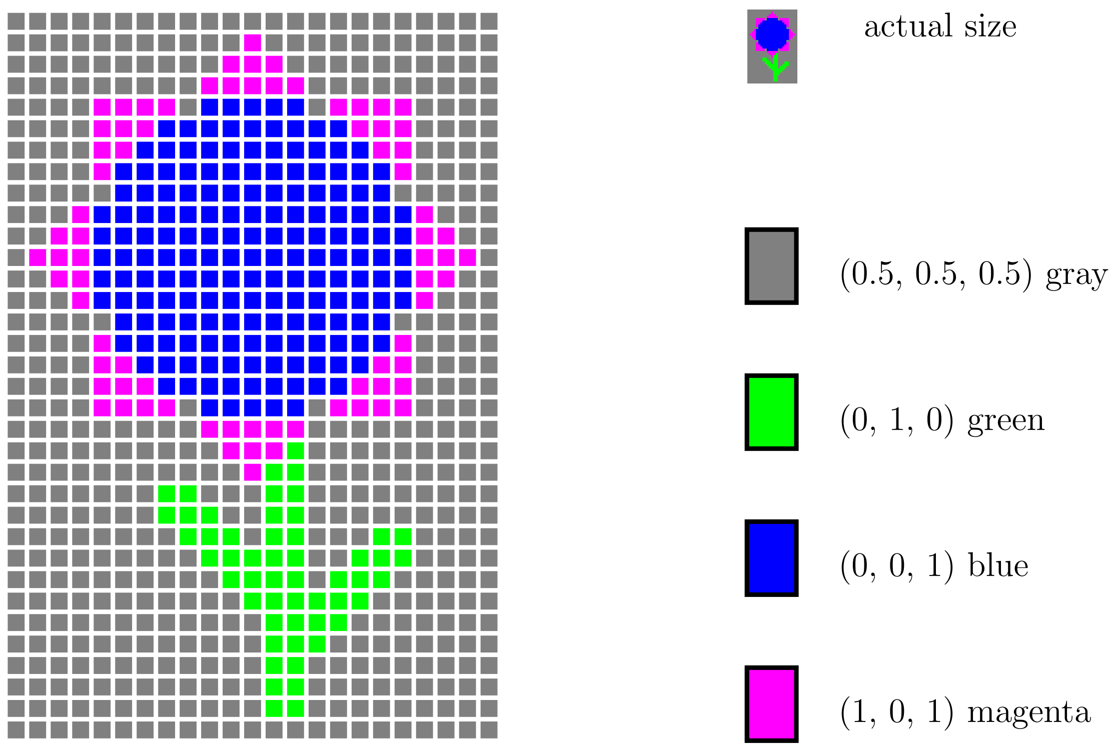


Figure 2: A small raster color image of a flower, with individual pixels depicted as squares. The size of this image is 23 by 34 pixels. Each pixel has a specific color shown as a red-green-blue triplet; thus, the image is completely defined by $23 \times 34 \times 3 = 2346$ numbers.

2.2 Imaging

From the perspective of this thesis, images are taken with digital cameras that project a three-dimensional view from a single viewpoint to an image plane and store the resulting data in digital form. This section describes the process of capturing a still photograph. A digital camera has the same optical working principles as an analog film camera; the main differences are in image storage as a discrete image, as opposed to a photosensitive film.

Reconstruction algorithms assume a *pinhole camera* model on the camera optics, and the images must be captured such that the model is accurate. However, as will be described in this section, cameras in reality seldom match the simplified model thoroughly, and care must be taken when configuring the cameras and post-processing the images later.

2.2.1 Pinhole camera

A physical camera is in its simplest form modeled as a *pinhole camera*; an ideal device that *projects* an image on its film through an infinitesimal aperture, rotated 180 degrees around the *optical axis*, i.e., the line that runs through the pinhole and is perpendicular to the image plane [9, 10]. Illustration of the process is given in

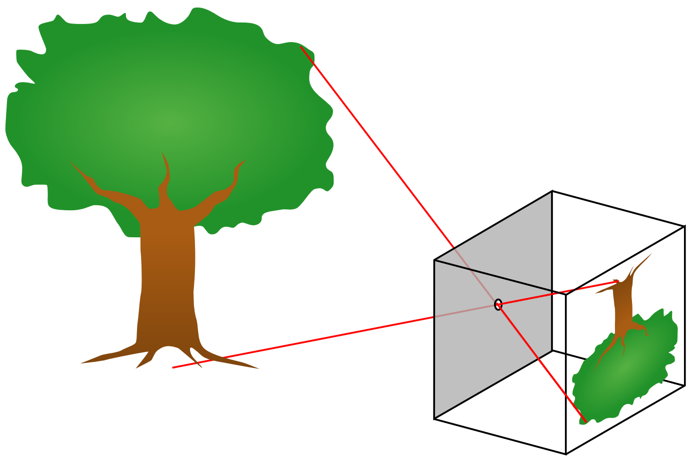


Figure 3: Pinhole camera principle. The box represents a camera. Image projected through the small hole is formed to the plane on the back of the box, rotated 180 degrees, as rays of light travel from the subject through the hole.

Figure 3; for example, light rays from the top right of the subject travel through the pinhole to the bottom left on the film.

The term *projection* means the reduction of three-dimensional objects onto a flat surface, seen from a single view such as the human eye or a camera lens. The *perspective projection* effect makes objects further away from the observer look smaller in the projected image. A cube projected on a plane is shown in Figure 4.

In the scope of this thesis, the only actual camera type considered is one with a *rectilinear lens*, i.e., an optic system that preserves straight lines, as a perspective projection does in a pinhole camera [9]. More exotic lenses that have a completely different projection are not considered, such as "fisheye" lenses, although their projections can be corrected in software.

Mathematically, a point in a three-dimensional world can be encoded in any coordinate system; in this work, a standard 3D cartesian coordinate system is used. In cartesian coordinates, a point is described as a location relative to a selected origin reference, as a triplet of distances along three perpendicular X, Y and Z axes. In a traditional right-handed system, X increases right, Y upwards and Z towards the viewer.

2.2.2 Optics

In practice, no actual camera works ideally; a pinhole-resembling camera with a tiny hole could be built, but it would have poor light gathering power and low resolution limited by diffraction effects [9, p. 29]. Light sensitivity is increased by using a larger hole, and the light rays are directed using carefully manufactured lens systems. Refraction properties of light rays and manufacturing defects introduce additional distortions that must be taken into account.

Construction of optical systems is well studied [9, 10]. Actual camera "lenses" (*camera objectives*) consist of not only a single glass element (optical lens) but many,

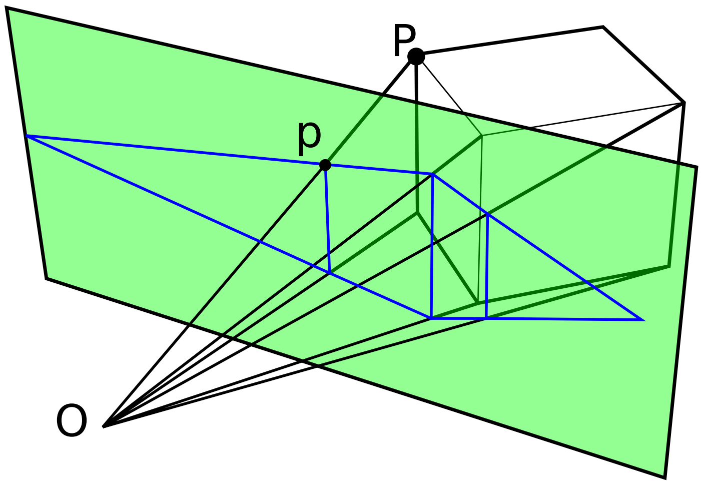


Figure 4: Perspective projection: a cube seen from a camera at origin $\vec{O}$. The 3D point $\vec{P}$ projects on to $\vec{p}$ on the green plane. In a camera, the rays of light travel through a common point to the film that is behind the origin; a virtual projection film between the origin and the object is often used for simplicity. Parallel lines in 3D connect in *vanishing points*, two drawn in the left and right.

especially in the case of zoom lenses. As an example, a mechanically simple fixed focal length Canon lens diagram is shown in Figure 5. There are several lenses, aligned so that their centers lie on the the optical axis parallel to it. Most objectives can be reduced to a *thin lens* model (depicted in Figure 6), an ideal optical model of a single infinitesimally thin glass element [9, p. 21] [6, p. 61].

Focusing A point on a subject is said to be *in focus* when the light rays originating from it converge at the same point on the camera's film. When the camera objective is adjusted for the object to be in focus, its parameters are changed slightly and a different thin lens is formed. *Focal length* is a fundamental property of a lens and determines its amount of magnification [6, p. 61] [9, p. 13]. Illustrated in Figure 6, the thin lens model and its focal length f encode a relation between an object at a distance a and its *focused* image at a distance b on the other side, as shown in Equation 1. A camera system typically has a *magnification* of much less than one, i.e., the image on its film is rendered smaller than the imaged subject, as $a \gg b$. For objects at distances approaching infinity ($a = \infty$), b becomes equal to f, and the thin lens resembles a pinhole with film distance f.

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

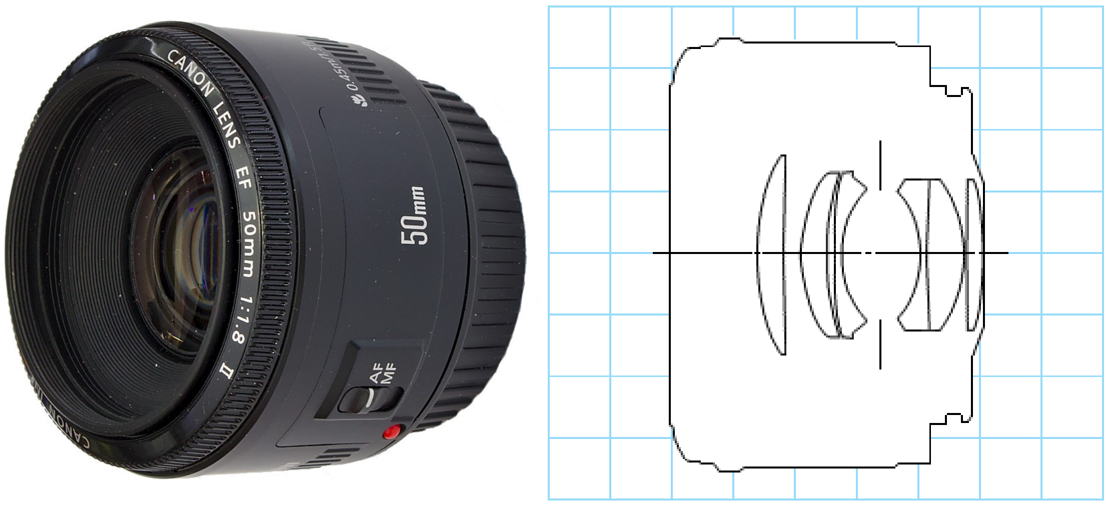


Figure 5: A fixed focal length Canon EF 50mm f/1.8 II camera objective. Left: the device, not attached to a camera. Right: block diagram of the lens construction, showing the six optical elements. Diagram by Canon Camera Museum [11].

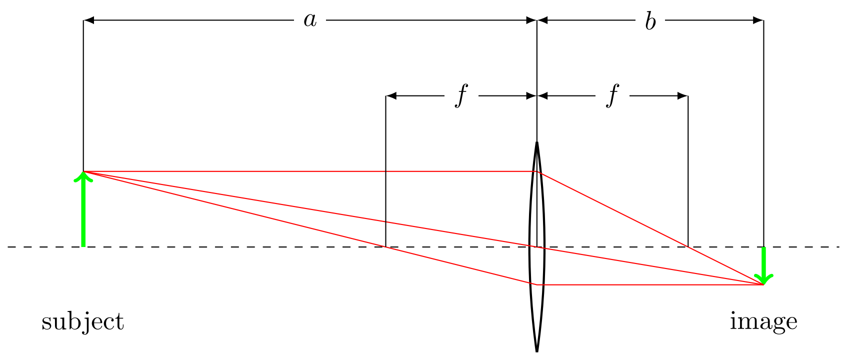


Figure 6: thin lens sample: $a = 6$, $b = 3$, $f = 2$. Actual subjects would be usually much further away; image depicts a macro setup for clarity. A subject at a distance a is *focused* on a film at a distance b on the other side, as light rays (red) travel via the lens.

Aperture The size of the hole where light effectively goes through is called *aperture*; the hole itself is called *aperture stop* and it is located inside the the objective in between its lenses. The diameter of a thin lens corresponds to the aperture diameter and to the stop's *entrance pupil*, the aperture stop seen from the front of the objective. Size of the aperture also directly determines the amount of light entering the image plane, and thus the relative exposure time for a required amount of light energy: doubling the aperture area halves the required exposure time. [9, ch. 13]

As in any practical case, the aperture is not infinitesimally small, and there will be some area of the scene that is imaged as sharp and others in front and behind of this area will be blurred. This effect is called *depth of field (DOF)* and is illustrated in Figure 7. In artistic photos, a *shallow* depth of field can be preferred, having the subject in focus to separate it from the background by blurring. In 3D

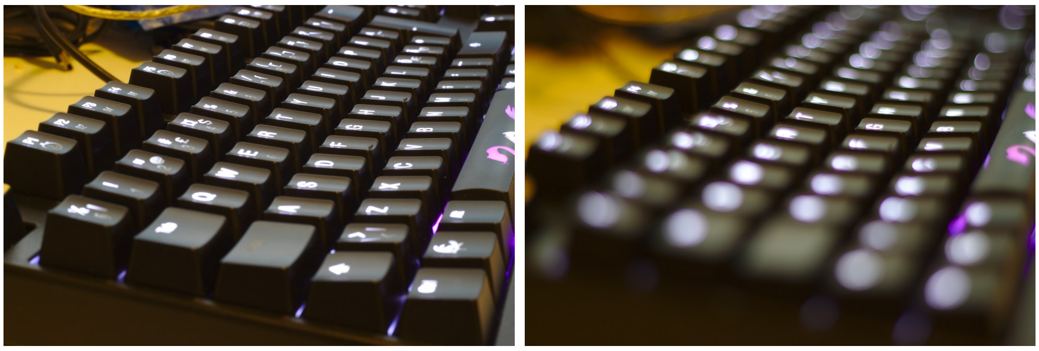


Figure 7: Deep and shallow depth of field in close-up photography using a 50 mm lens. Left: f/16 aperture, 15 second exposure, showing only slight blur in the nearest keys. Right: f/1.8, 0.2 s exposure, with only a small sharp area. The image blurs and the distinctively brighter small lamps on the keys show as increasingly larger bright spots as the circle of confusion increases.

reconstruction, exactly the opposite is preferred: deep focus, i.e., objects at every depth from the camera should be sharp. The aperture size is presented as relative to the focal length and is commonly called the *F number,* N [9, p. 23], specified via the aperture diameter d:

$$N = \frac{f}{d} \tag{2}$$

Clearly, doubling the area of the aperture equals to an increase of the diameter d by a factor of $\sqrt{2} \approx 1.4$. The aperture size is typically denoted literally as f/N, such as $f/1$ or $f/1.4$. This makes the light energy reaching the film universal among all focal lengths. Full multiples of $\sqrt{2}$ are called *stops*, and the perceived f number changes as a function of subject distance [9, p. 23], which is left out here for simplicity.

Circle of confusion As the blur happens gradually, the "level of sharpness" at each point is often defined via a *circle of confusion* (CoC) [9, p. 25]. The distance from the lens along the optical axis where perfect focus is attained is called the *focal plane.* Every point nearer or further away shows up on the film as a circle (for circular apertures), instead of a point; when this circle is significantly larger than the resolving power of the observer, the image of this point is perceived as blurred as the image is the integral of all of these circles.

Depth of field is defined as difference between the shortest and farthest object distances that still generate an acceptable CoC, found geometrically. When f is the focal length, N the aperture f number, s the distance to the subject, and c the maximum acceptable CoC, the depth of field D is (Greenleaf [9, p. 25]):

$$D = \frac{2sf^2Nc(s-f)}{f^4 - N^2c^2(s-f)^2} \tag{3}$$

In Figure 8 that depicts the formation of the circle of confusion, Z' and Z are the maximum and minimum acceptable distances near the subject, and $Z' - Z = D$.

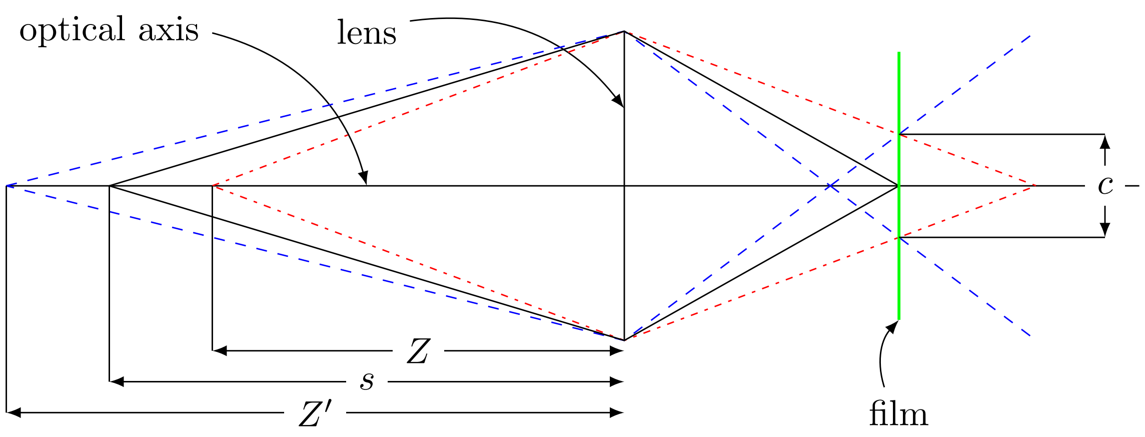


Figure 8: Depth of field and circle of confusion with a thin lens, seen from the side (not drawn to scale). Subjects at distance s project perfectly at the film. Distances between Z and Z' project to a circle smaller or equal to c. It is easily seen that for subjects at longer distances, the depth of field increases.

Sufficient depth of field is important in near-field 3D scans, which translates to a small aperture, and consequently, relatively bright lights in order to maintain a short exposure time. The subject should fill whole image frame to use the pixel density effectively. Increasing the distance to the subject would increase depth of field, but enlarging the subject in the frame again would need a higher focal length, which decreases depth of field.

Acceptable size of the CoC depends on the resolving power of the viewer; when a photograph is looked at, the print size and distance of the viewer determines the circle [9, p. 26]. A standard-considered formula in photography determines the maximum acceptable CoC by dividing the image size by 1500, originating from human vision performance [12, p. 88, 92]. As an example, for the standard "full-frame" 35 mm film size with an image size of 36 x 24 mm (approx. 43 mm diagonal), the CoC yields $c = 43\text{ mm}/1500 \approx 0.03$ mm. For a normal 50 mm lens with the subject at a distance of 1 m, a small aperture of f/11 and using a circle of confusion of 0.03 mm, Equation 3 gives depth of field of $D \approx 25$cm.

In a computer vision system, this purely perceptual guideline is typically ignored and the circle of confusion is determined by the sensor resolution or pixel size. The disk should not span too many pixels, so that accurate detail would be resolvable from the digital image. The pixel size is typically in the order of micrometers, with which a much smaller CoC is required. The sensor of Canon EOS 1D X, for example, has a pixel size of approximately 7×7 micrometers [13] which is considered large. When properties such as type of the subject texture and the quality of the optical path to the sensor influence the resolving power, the circle of confusion is typically used as a guideline instead of a strict rule.

Diffraction The required amount of light is not the only factor that limits the minimum aperture size. *Diffraction* is an effect which reduces image sharpness with

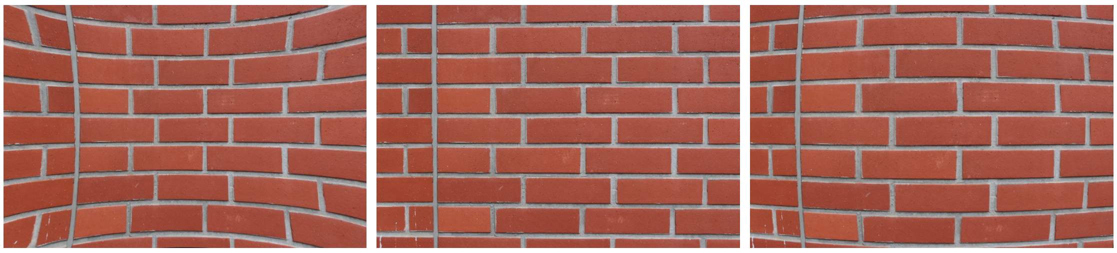


Figure 9: Simulated pincushion (left) and barrel (right) distortions by applying undistortion on an undistorted image (center) using OpenCV.

small apertures. When a light ray diverges on a small aperture due to the wavelike nature of light, the different distances traveled make the rays interfere each other and produce a two-dimensional diffraction pattern, called an *airy disk* [9, ch. 4]. The disk blurs the image in a similar fashion as an object that is out of focus, as each light ray produces its own particular disk. Size of the airy disk is relative to the size of the aperture and the light wavelength. Thus, the aperture size must be chosen as a compromise between the unwanted effects of DOF and diffraction. The disk size can be approximated [9, p. 29] by

$$c = 1.22\frac{\lambda f}{d} = 1.22\lambda N \tag{4}$$

which, for example, for red light having a wavelength λ of 700 nanometers, with an aperture of f/11, results in ≈ 0.01 mm (compare to above CoC of 0.03 mm). Again, this should be compared to the image sensor's pixel size, and already an aperture of f/11 would approach the diffraction limit.

Geometric distortion While most rectilinear lenses are close to a perspective projection, practical optical systems introduce some non-linear *radial* and *tangential distortion* that affect the accuracy of the ideal pinhole model as light rays travel through the pieces of glass [14, 15]. Common distortions are the purely radial so-called *barrel* and *pincushion* distortions, where the lens magnification is a nonlinear function of image ray distance from the center of the lens, caused by the shape of the lenses [14, 15]. In addition to radial, tangential distortion is less significant, and it is often ignored. Its cause is small misalignments in separate elements in a single optical system, lenses being offset from each other and not being parallel to the image plane [10, 14].

Because stereo vision assumes that the images are free of nonlinearities, i.e., straight lines in the world should remain straight in 2D images after the projective transformation, it is important to correct the geometric distortions beforehand [6]. The radially symmetric radial error model and correction proposed first by Brown [14, 15] and used directly or with slight alternations in programming libraries, e.g., OpenCV [16], is

$$\begin{aligned} x_{corr} &= x(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) \\ y_{corr} &= y(1 + \rho_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6). \end{aligned} \tag{5}$$

where x and y are the original coordinates in the distorted image on the film or image sensor, x_{corr} and y_{corr} are the corrected ones, κ_1, κ_2, κ_3 are coefficients specific to the radial distortion and r equals to the distance to optical origin, located at $(x_c,\ y_c)$:

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2} \tag{6}$$

Trucco and Verri [7] present only the two first coefficients, and it is not rare to use only the first κ_1; for lenses with low distortion, this is often enough. A complete derivation of the model involves detailed trigonometric analysis of thin prism models on lens elements [14, 15], which results simply in Equation 5, a simple polynomial. For tangential-only distortion, the corresponding correction has the following form:

$$\begin{aligned} x_{corr} &= x + 2\rho_1 xy + \rho_2(r^2 + 2x^2) \\ y_{corr} &= y + 2\rho_2 xy + \rho_1(r^2 + 2y^2) \end{aligned} \tag{7}$$

where x, y, x_{corr}, y_{corr} and r are as in Equation 5 and ρ_1 and ρ_2 are new coefficients specific to tangential distortion.

The two independent types of distortion are combined by summing them both to the original coordinate, combining Equations 5 and 7:

$$\begin{aligned} x_{corr} &= x(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) + 2\rho_1 xy + \rho_2(r^2 + 2x^2) \\ y_{corr} &= y(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) + 2\rho_2 xy + \rho_1(r^2 + 2y^2) \end{aligned} \tag{8}$$

The effects of different parameters are depicted in Figure 10. The image dimensions are usually normalized such that the origin lies at the image center, and the distance r from the center has a maximum value of 1, making the distortion parameters universal for all image resolutions. A typical image sensor is not a square but a rectangle, and clips the distortion circle accordingly, such that the maximum coordinates along the different axes are different.

The Brown's model describes the distortion characteristics for a lens system on an otherwise perfect image, but typically the inverse transform is required, because a distorted image is given as noted by e.g. Villiers et al. [17]. For each pixel in the undistorted destination image, the pixel value is necessary to find from the distorted source, but the mapping described by the model works to the other way. However, Villiers et al. note that as plain barrel and pincushion distortion can be considered as inverse transforms to each other, the same model can actually be used for the inverse direction if the more complex tangential part is left out [17].

Because digital images consist of a discrete pixel array, the pixel coordinates given by the undistortion do not necessarily lie on exact pixel positions; the sampling should then interpolate between neighboring pixels, considering the image as a continuous but discretely sampled 2D function [18]. Throughout this thesis, interpolation is assumed to be performed as necessary when sampling an image.

2.2.3 Shutter

A static scene is imaged by exposing the photosensitive film (or digital sensor) to the light passing through the optics for a short time, keeping the camera and subject

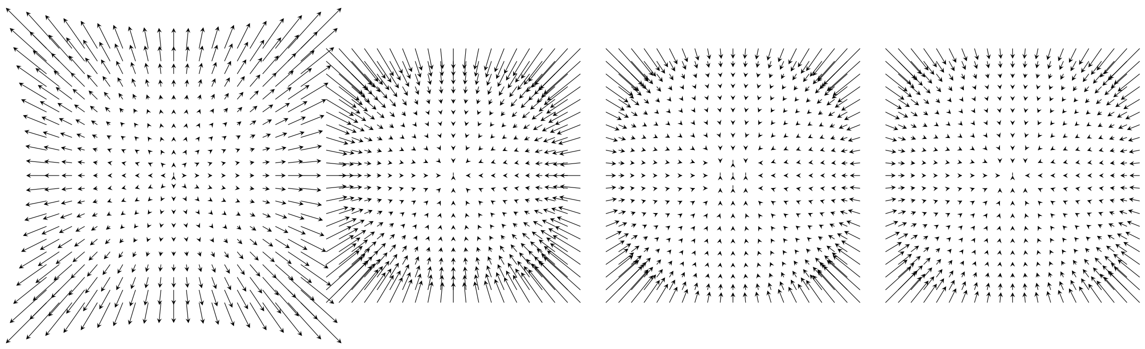


Figure 10: Purely radial distortion correction with different values of two parameters. The arrows point from original coordinates (x, y) to corrected ones (x_{corr}, y_{corr}). From left to right: $\kappa_1 = 0.16$, $\kappa_1 = -0.16$, $\kappa_2 = -0.08$, $\kappa_1 = -0.04, \kappa_2 = -0.04$. The different shapes of second and fourth order polynomials are easily visible: the higher-order terms are visible have most effect in the farthest sides of the image. The leftmost correction is for a barrel distortion, and the others are for pincushion distortion.

steady. A *mechanical shutter* blocks the light before and after the exposure [9, ch. 13]. In recent digital image sensors, the sensor itself can be made inactive to light or cleared quickly from the accumulated charge; this is called *electronic shutter* [19, 20].

Shutter types All shutters serve the same purpose: to restrict light from entering the image plane before and after the exposure. There are several types of constructions for shutters, depending on the application. Some cheaper compact cameras have a small *diaphragm shutter*, or *leaf shutter* in the lens center near the aperture stop. A leaf shutter moves quickly away from the light path and back, either as a single opaque piece or as several *leaves*, analogously to the aperture stop whose size can be changed [9, p. 117].

A *focal plane shutter* [9, p. 123], used in most system cameras, is installed in front of the film plane. Leaf shutters would be expensive to install in each interchangeable lens, and a focal plane shutter can operate at higher speeds. Focal plane shutter consists of two curtains that move to the same direction; one moves out of the way of the film to let light pass to it, and another moves to cover it back again when the exposure time has elapsed. Figure 11 illustrates the working principle of a planar shutter. In film cameras, the focal plane shutter was originally two fabric “curtains”; today, they are constructed from small, interleaved metal or plastic sheets.

Some cameras have only an *electronic shutter* instead of moving parts. Electronic image sensors can only be cleared and read out at a restricted rate, and their operation is similar to a focal plane shutter. [19, 20]

Video film cameras that capture each frame to a separate part of film used a spinning wheel with a transparent window, called rotary disc shutter [21]. This rotates at a precise speed synchronized to the frame rate, exposing the film to light for a consistent duration for each frame. The size of the window constitutes to the

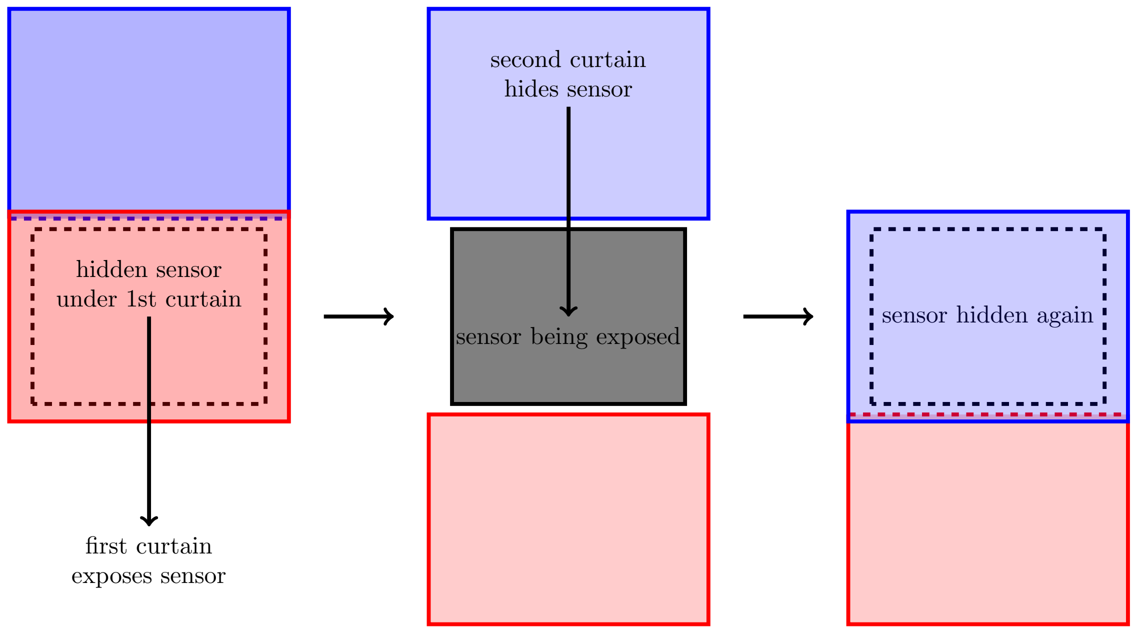


Figure 11: Focal plane shutter has two moving curtains to cover the film, opening it momentarily so that each part of the film receives light for the same amount of time. From left to right: initially, the film (grey) is blocked. Once the first curtain (shown in red) moves away, the film is exposed to light. A second curtain (shown in blue) stops the light from entering again. The shutter resets by moving back to the initial position.

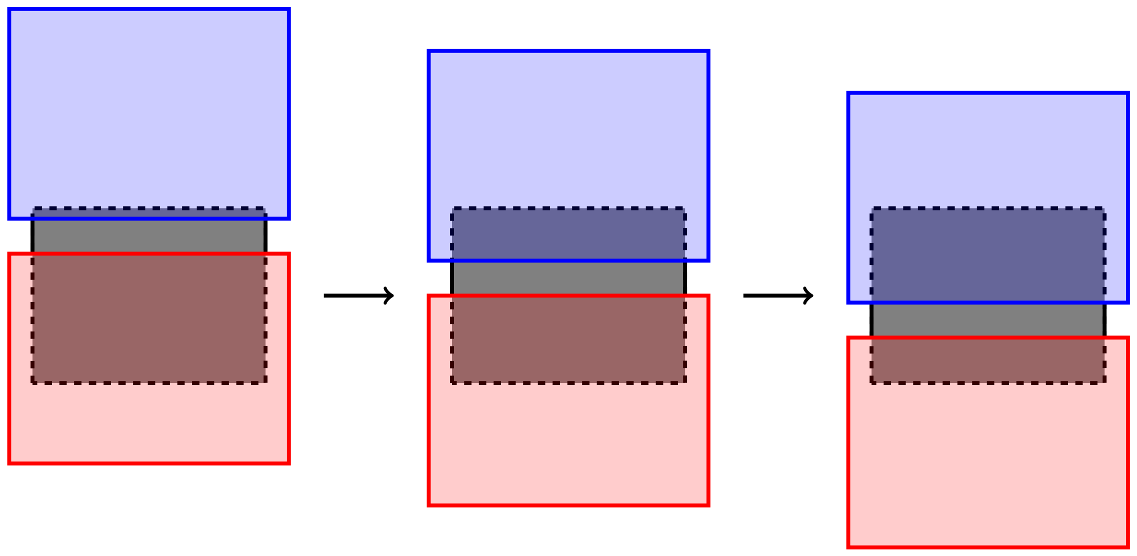


Figure 12: Rolling shutter effect in a focal plane shutter happens when the exposure time is short relative to the curtain speed. The curtains have to move at the same time, exposing only a part of the film at a time. Fast subject movement during the exposure leads to artifacts and a rapid flash light would illuminate only a part of the film.

amount of light and motion blur. Video is more discussed in Section 2.3.

Artifacts If the camera or the imaged subject moves while exposing the frame, the subject gets smeared along the movement on the image, an effect known as *motion blur*. A related *rolling shutter* effect is produced using a focal plane shutter with very high speeds, where the limit of curtain speed prohibits fully opened shutter. The closing curtain starts to move before the first has stopped, exposing only a narrow band of the film at a time. Fast mechanical shutter movement is depicted in Figure 12. When an object is moving to the same direction as the narrow shutter slit, i.e., vertically, it appears smeared; when the movement is horizontal, the subject appears to bend. An electronic flash light that outputs a very brief pulse of high-intensity light also cannot be used with high focal plane shutter speeds, as it would illuminate only a narrow slit of the sensor. Electronic shutters also suffer from a similar effect; this is more discussed in Section 2.2.4. Since a leaf shutter opens radially near the aperture stop, it does not have this effect, and it can be used at higher speeds with flash lights. Leaf shutters suffer from another artifact: at higher speeds, the longer time spent covering the far sides of the image leads to uneven brightness. Result of shooting a moving object with rolling shutter is depicted in Figure 13. A ruler was attached to a rectangular plate, and moved left by hand aligned to a table, with different shutter speed and sensitivity settings. Skew induced by rolling shutter is visible always, whereas motion blur is reduced with a faster shutter speed.

Lag When taking a picture by pressing a button, the camera does not instantly start to expose the image. Each camera has a specific *shutter lag* that is the time that elapses between when the shutter button is pressed and when the actual exposure starts to happen. Some sources include in this term also the time for the camera to perform autofocus and measure sensitivity settings which can take a significantly longer and unpredictable time. Shutter speed has also small variations over time when the camera ages and mechanical parts wear out.

Some difference also exists among same camera model from mechanical variations and software processing. Shutter lag of consumer devices typically varies between tens to hundreds of milliseconds [22].

2.2.4 Image sensors

Historically, photographs were saved on a photosensitive film. With digital cameras, a rectangular electrical *sensor* replaces the film. The sensor consists of a uniformly spaced grid of small sites that convert the visible optical image to electrical signals, built as an integrated circuit package. In conjunction with the digital signal processing in the camera, the sensor produces a digital image file.

As introduced in the beginning of this section, digital images are represented as arrays of colored pixels. In order to obtain an appropriate image, the sensor must capture the light it sees also as a discrete array. The separate red, green and blue color bands are captured in color cameras with separate parts of the sensor, as will be explained.

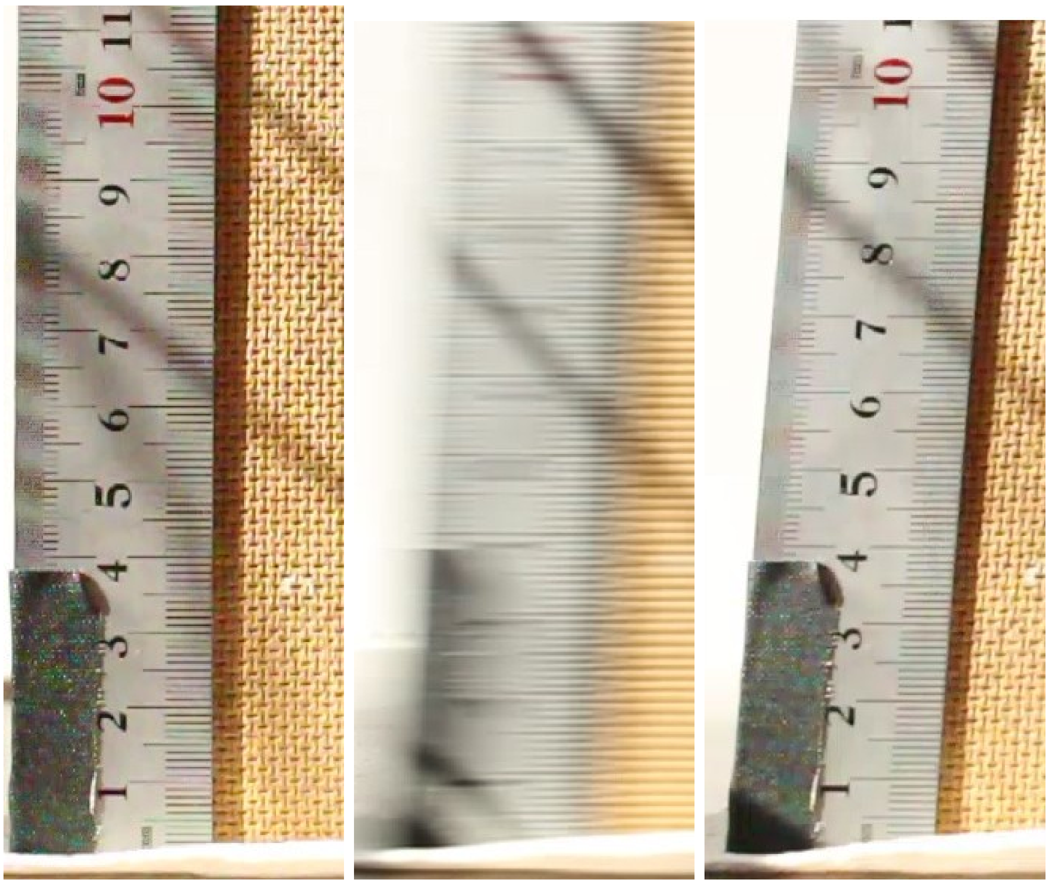


Figure 13: Motion blur and rolling shutter in a ruler, moving left. The ruler does not actually skew or tilt physically; it is vertical and rectangular at all times, but appears skewed. Left: a stationary ruler. Middle: 1/100 s shutter speed. Right: 1/4000 s shutter speed. Camera aperture was set to f/8; the pictures are cropped from 720p video recorded with a Canon EOS 700D. (Color filter and low sampling rate artifacts are visible as color errors in the high-contrast lines of the ruler.)

Photosites A sensor consists of individual photosensitive semiconductor sites that each accumulate charge and ultimately encode a single numerical value, relative to the amount of light received [23, ch. 3]. Physical pixels cover less than the full area of a sensor, as there are gaps between them; microlenses between the sites direct the light hitting the gaps to the neighboring pixels in modern sensors, with varying color performance [23, p. 64] [24].

A single pixel sensor acquires all light hitting it, independent of the wavelength. The most common method to capture the three primary colors is to interleave red, green and blue colors in a single sensor by filtering the wavelength that can enter each pixel with a colored film over the sensor, with a special repeating pattern called *color filter array*. The missing values for other colors bands are interpolated from surrounding pixels. The most popular interleaving pattern is the Bayer filter [6, p. 76], illustrated in Figure 14. Other arrangements than red, green and blue are also used; e.g., red, green, blue and emerald or cyan, yellow, green, and magenta [25].

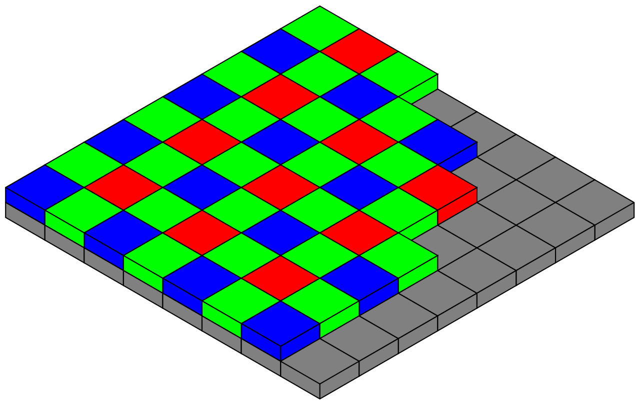


Figure 14: A small partly drawn subset of the repeating $2x2$ Bayer pattern. Each square represents a single pixel; the colored pattern on top filters the light per each pixel on the sensor array.

Alternatives have slightly better color accuracy at the expense of spatial resolution.

Sensor technologies Near the raw pixel sensors is analog and digital logic for converting and reading out the data and passing it to an image processor. Quality and type of this conversion affects some properties, most important of which are analog-digital converter noise and dynamic range, and speed at which the image is read out from the sensor.

Two major sensor types exist: CCD (Charge-coupled Device) and CMOS (Complementary Metal-Oxide Semiconductor). The types vary by their manufacturing techniques and photosite internals [23, 24, 26].

Both CCD and CMOS have their applications, but it suffices to say that CMOS is the one used in most still cameras because of its lower price and power consumption, and higher speed. A CMOS sensor converts light immediately to voltage, and has additional transfer circuitry at each pixel, reducing the *fill factor*, the sensitive parts of the photosites [23, p. 62]. In a CCD, the charge is first stored as electrons as in a capacitor, and a row is transferred serially from pixel to pixel to a separate voltage converter and amplifier. Still, both are typically read out ultimately in a serial fashion, requiring longer reading time for larger amounts of pixels. A CCD typically uses additional photosites permanently blocked from the light for storing the accumulated charge temporarily. A CMOS sensor, on the other hand, cannot easily employ additional storage pixels, and is read sequentially, making the reading process resemble the rolling shutter effect. Rolling shutter is especially noticeable in videos shot with a CMOS camera, because a mechanical shutter is not used for video and the readout process is relatively slow [19, 26].

Error sources Various steps of the image processing contribute to image noise; the major sources are shot noise and several electrical noises [6, 27]. *Shot noise* exists due to the particle-nature of light; each photon adds a constant electrical charge to the photosite it hits, and if the overall number of photons is small, statistical

Poisson-distributed noise will be present [23, p. 74]. This type of noise is visible in less bright areas. Clearly, no electronically better sensor can improve the visibility of shot noise. The only way to reduce it is by exposing the sensor to more light: for this reason, sensors with a larger surface area are preferred in applications where noise is unwanted.

The Gaussian-distributed *electrical noise* has various sources in sampling, transfer and conversion. Each pixel receives an additional random signal level that shows up as additive brightness, also noticed usually only in poorly illuminated conditions or dark areas. Noise increases with sensor's signal gain, or *exposure sensitivity*, described using the ISO system ("film speed"). Sensor *responsivity* usually refers to the raw signal the photosite sensor produces per unit of optical energy [23, p. 78]. The resulting digital value depends on this and many other terms on the signal path.

A more sensitive ISO setting on a sensor requires less light for a bright-looking image, because the signal is amplified more before digitization. This amplification also amplifies the original signal noise. Additionally, the reduced exposure time allows the shot noise to be more visible. Due to these reasons, the gain level should be kept as small as possible and the amount of light entering the sensor as large as possible. This leads to requiring bright lighting in the imaged scene or a longer exposure time. Higher ISO should be used only if shorter exposures are desired due to a moving subject. Better sensors advertised as more responsive to light typically have larger sensors or less noise-sensitive circuitry, resulting in higher signal to noise ratio [24].

Photosites may also accumulate leaking charge on their own even though no light is entering the site because of *dark current* [24, 27]. Additionally, the analog-digital converter introduces *quantization error* where an analog signal is not perfectly represented by a discrete number; with the high bit depth of modern sensors, this is practically not an issue. A high bit depth also presents larger *dynamic range*, i.e., the difference between the brightest and darkest possible representations [24, 27].

2.2.5 Lighting

In addition to multiple cameras at different locations and poses to encode a subject's geometry, uniform and soft lighting is also required for consistent color reproduction. A large variety of different studio lights are available in the market for different lighting situations, emitting either continuous light or bright flashes.

Surface color The *perceived* color of a point on a surface is the light exiting it and reaching the viewer. Therefore, the light sources should be uniformly white and cast no shadows. Surfaces reflecting light can be modeled as a combination of *diffuse* and *specular* components that are functions of light direction, viewing direction and sometimes additionally location on the surface. Surface reflection models are described with a *bidirectional reflectance distribution function, or BRDF* [29]. Such distribution model of the surface's global or local microgeometry specifies how much light reflects in different directions [28]. *Diffuse* means a material that reflects light uniformly to all directions, depending only on the angle between the

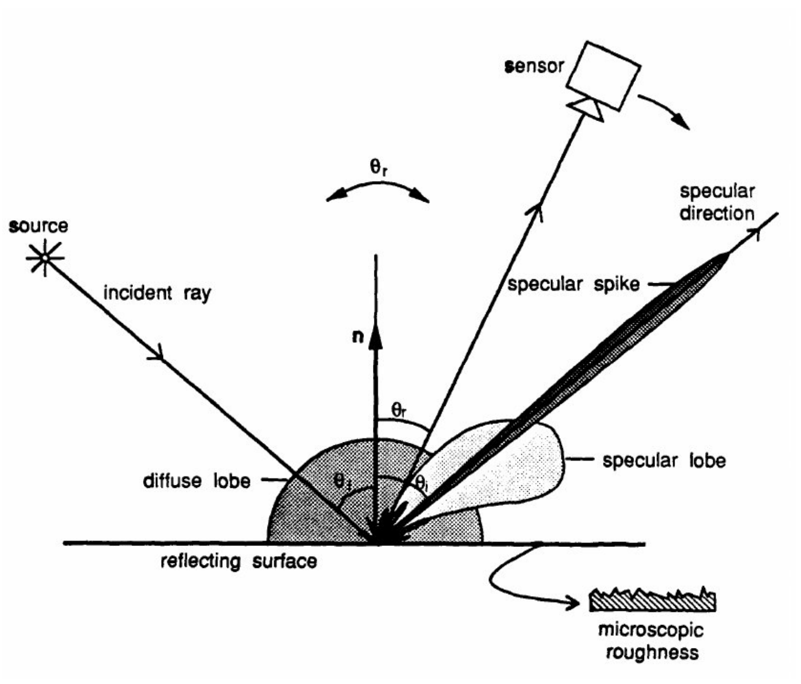


Figure 15: Polar plots of three reflection components as functions of the viewing angle, as suggested by Nayar [28], showing the brightness seen from different angles from the surface normal. In a reconstruction subject, the bright specular spike and lobe reflections should be minimized, and only the diffuse color is desired.

surface normal and light location, whereas more *specular* surfaces are composed of a different microstructure that reflects light in a mirror-like way; the surface brightness depends on where it is looked from. In Figure 15, different types of reflections seen by the camera are visualized as a function of the camera viewpoint.

Color capture Ideally, the camera would only record the diffuse reflection that represents the surface color (*albedo*); any shadows and the specular components are unwanted artifacts from the standpoint of simple color reconstruction. *Specular highlight* or *specular reflection* refer to a mirror reflection of the lamp on the imaged surface. Because most reconstruction methods fundamentally assume uniform lighting conditions, a property called *brightness constancy* (i.e., the brightness and color of a surface looks the same, regardless of viewing direction or time moment), they perform poorly when a light shines off a surface differently in different views [7, ch. 8.3]. As images describe the color via the brightness that reflects off of a surface, the perceived brightness must be held constant. The same restriction provides a basis to other algorithms, such as object tracking [30, 31]. Examples of complex surfaces containing much specularity are glossy textures and sweaty human skin.

On the contrary, in photographs and image synthesis, shadows and reflections are natural, and images without them would look odd to the eye; capturing only albedo of surfaces – the color reflectivity of a surface – is particular to texture capture in reconstruction. In practice, polarizing filters may be used in front of the lights and in the camera lenses, to filter out most of the specular lighting. Advanced capture and rendering methods may take advantage of also the specular component [32]; however, capturing material properties is easily done separately from the geometry capture [33].

Light sources As the surface should be illuminated as evenly as possible to minimize specular highlights and shadows, it is important to pay attention to light sources. The often used photography terms *soft* and *hard lighting* refer to the size and power of light sources: bright spotlights are hard, and larger and relatively less bright lamps are soft. Softness can be added by increasing the light's area while holding its power constant using a "diffuser", such as a large surface where light is reflected off or a white cloth between the lamp and the imaged subject. [34, p. 108]

Powerful electronic flash units use usually a gas-filled tube that is excited with high voltage to produce a short, bright burst of light for some milliseconds; since this tube is small, a diffuser should be used for uniform lighting. Flashes are especially suitable for human subjects, because the subject sees the bright light only for a brief moment; thus, the light does not annoy the subject as continously bright lamps would. Short bursts of light in an otherwise dark environment are also suitable for stopping motion and synchronizing multiple cameras, as only a short time is effectively exposed to light [34].

2.2.6 Data capture path

The image information travels from the subject through the optics to the image sensor's surface, via an analog/digital converter, the camera's processor(s) and finally to mass storage [6]. The final mass storage is typically on the camera in consumer devices, or on a computer reached by a wired connection for more sophisticated machine vision cameras [35, 36]. Many parts in this path contribute to the quality of images and the rate at which the images can be taken, usually measured in frames per second (FPS).

For example, for the high-end Canon DSLR camera EOS-1D X, a continuous shooting speed of 14 frames per second is advertised [13]. With its resolution of 5184 by 3456 pixels and 14-bit processing, an estimate for the bit rate from the sensor is

$$5184 \times 3456 \times 14 \times 14 \approx 3.5 \text{ Gb/s} \tag{9}$$

by average, or 0.4 gigabytes per second, requiring also high speed from the processor and mass storage. Due to the cost of transferring and processing large amount of data, video is typically recorded in lower resolution for faster FPS.

A "raw" image file can be saved with more advanced cameras, containing the plain sensor data with very little processing done. The raw files saved by DSLR cameras are packed in a proprietary lossless format and contain also metadata about certain camera settings. When a raw file is not used, the camera processes the image with device-specific correction filters and applies image enhancements, e.g., white balance correction and sharpening, and finally saves the image as a standard image file such as JPEG [6, p. 412]. For such automatic in-camera processing, all cameras should be identical for consistent reconstruction results.

Video files are almost always saved in a lossy video format by consumer devices, the H.264 format being common for modern cameras. Compression introduces loss of detail and artifacts typical to the particular algorithm, often visible as blocks or blurring [37]. Reconstruction quality from frames extracted from such a compressed video may not be as high as from separately saved stills that encode more spatial information. Machine vision cameras and some professional video cameras can record raw uncompressed video, and some others can be unofficially enhanced; Canon DSLRs can be tricked using a third-party software modification [38] to capture the raw stream to a mass storage card, but only some models support the cards necessary for the high recording speed.

2.3 Video

Video is ordered electronic pictures displayed one after another. Humans perceive *motion* when a previously seen object is seen again in a nearby location. Current digital video technology encodes motion pictures as in films, in a sequence of still images, recorded and displayed at a constant rate. Three-dimensional motion is generally no different: it is encoded as discrete poses in sequence. In order to do motion capture in stereo vision, video material from two or more of cameras is used to initially capture a sequence of still photos.

To reconstruct the frames in 3D, each set of frames encoded by the cameras should be *synchronized*, i.e., shot at the same time, to fulfill the reconstruction assumption of a static subject.

2.3.1 Sequence of frames

When scanning a scene continuously, a camera shoots frames using the same principles as in photos, but does it in sequence, at a speed that is called *frame rate* [39]. For each frame, the shutter is open for a duration of *exposure time* or interchangeably *shutter speed*. In *interlaced video*, two frames make up a whole picture that covers all pixels. Interlacing alternates between odd and even lines to increase update rate, and was invented for cathode ray tube displays to reduce apparent flicker [39]. The alternation results in tearing artifacts when imaging horizontally moving subjects (shown in Figure 16). Many CCD video cameras still produce interlaced stream. Video that contains full frames only is called *progressive*.

Consumer video cameras, video-capable pocket cameras and DSLRs typically offer a few choices of frame rate. Traditionally, the movie industry has used 24 FPS; 25 and 30 are common alternatives used in TV broadcast, some cameras being capable to twice the speed at a lower resolution. Actual FPS is slowed down from the advertised by a factor of 1.001 because of historical reasons. 30 and 24 FPS refer to approximately 29.970 and 23.976 FPS, respectively. [40]

Externally triggered cameras can clearly be configured to record at any arbitrary rate and exposure, as long as it is in the limits of the camera speed capabilities.

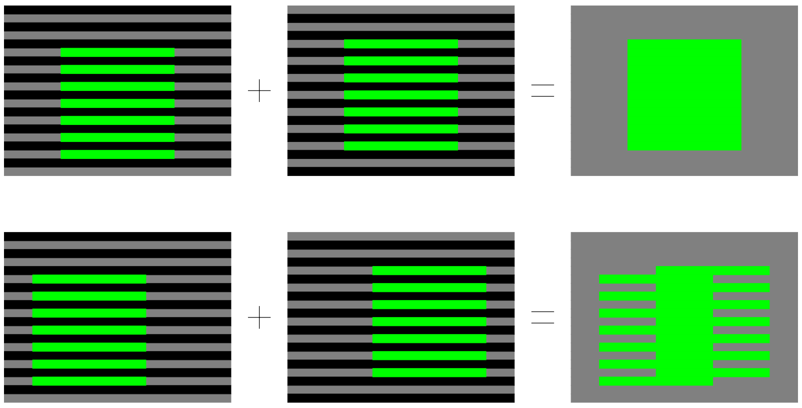


Figure 16: A cropped area of interlaced scanning. From left to right: even lines only, odd lines only and full picture combined. Top: stationary square. Bottom: a moving square, showing tearing if the frames would be combined.

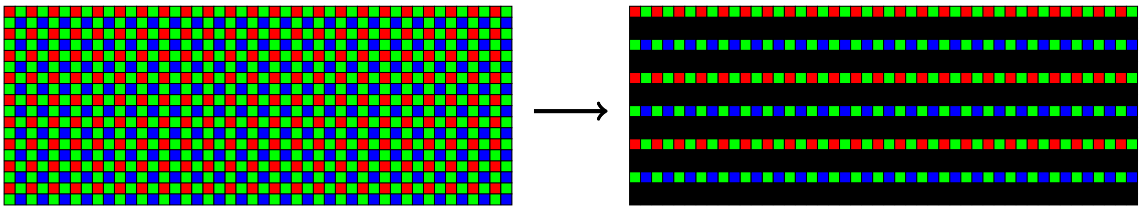


Figure 17: Line skipping in a DSLR camera in video mode. Left shows a part of the full sensor and its bayer pattern; on the right, only every third line is captured.

Synchronized machine vision cameras do not record video on their own, but instead monitor a clock signal and output frames at the clock rate. The frames are then recoded to a computer using *frame grabbers* for interfacing with the cameras. [35]

DSLR video modes use almost the sensor's full size but skip some of its lines when recording video, resulting in the same field of view as still images but suffering from significant aliasing patterns with certain subjects with high frequency information. An alternative would be to crop a smaller part of the sensor for the frames. Additionally, pixels in the video frames do not represent physical pixels anymore, posing awkwardness in camera calibration. Line skipping is depicted in Figure 17. The color filter array makes the problem even worse for small-detailed features of high color detail.

2.3.2 Frame rate and shutter speed

A higher frame rate results in more accurate representation of motion as a whole, and a faster shutter speed helps to reduce motion blur in individual frames. For reconstruction, ideal exposure time would be infinitesimally short; practically, the amount of available light restricts the time, even though a camera would be faster. Similarly, what happens when the shutter is closed is not recorded; ideal frame rate would be faster than any period of motion in the scene. For motion tracking based on frame differences, a long enough time should be passed for reliable motion

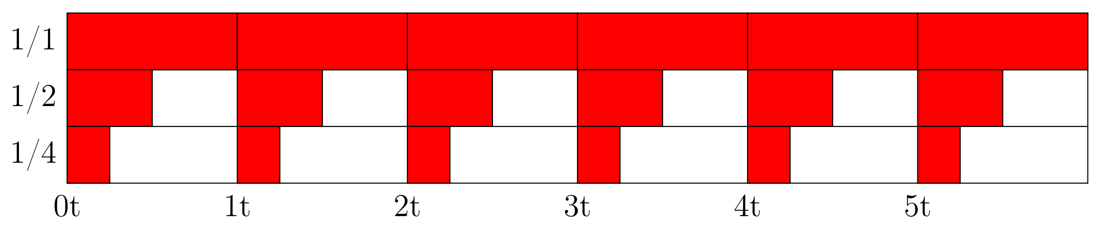


Figure 18: Varying shutter speed compared to constant frame rate $1/t$. Red rectangles illustrate exposure time, i.e., open shutter. For an always open shutter, the sensor would record continuously. A ratio of 1/2 is a "normal" rate most used in film. 1/4 would result in slightly less motion blur and less captured light intensity.

direction detection, though.

In contrast, in motion picture industry the shutter is kept open deliberately so long that fast motion is blurred, because it is considered aesthetically pleasing to human eye [21]; even though the motion is blurred, it appears less jerky and more temporal information about the scene movement is encoded per frame than when exposing infinitesimally short frames, at the expense of spatial resolution. A typical shutter speed relates to frame rate such that a frame is exposed half of the time between frames [21].

Film cameras and some professional video cameras use a rotary disc shutter that has an opaque 180 degree sector. Having half the exposure time of frame rate is sometimes called "180 degree shutter" due to this historical reason [21, p. 37]. While the light is blocked, film is mechanically advanced to the next frame. Digital video analogously downloads data from the image sensor and clears it during this time.

2.3.3 Multi-camera synchronization

Concurrent recording of multiple view video is more complicated than shooting still photos simultaneously. The cameras should be synchronized together to hold their shutters open at the same time during each frame. An assumption of stereo vision is that the images encode a static scene, i.e., geometrically same objects, which would not hold otherwise. Lack of proper synchronization can be compensated to some degree with stroboscopic lighting or morphing between recorded frames if hard syncing is not possible [41].

Within the scope of this thesis, synchronization issues can be divided roughly in three sources of error, assuming N cameras each having its own video file with same frame rate: *offset*, *drift* and *jitter*. With the same frame rate, the frames can be indexed with numbers starting from 0. Times when the frames taken relative to a global clock is noted as t_{Ci}, where C is a camera identifier and i is the increasing frame index.

Offset is the time difference of two streams: for an ideal camera A at each frame t_{Ai} for i, and an offset camera B at t_{Bi}, a constant error e is added:

$$t_{Bi} = t_{Ai} + e \tag{10}$$

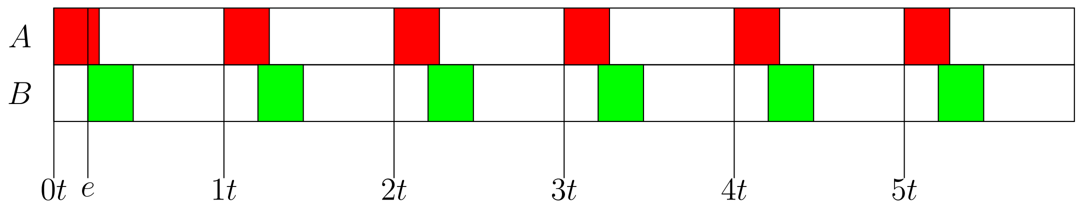


Figure 19: Video phase difference, i.e., offset, illustrated as a timeline. Red rectangles illustrate exposure times of camera A, green rectangles the same for camera B. Frame period is t, and the cameras have a constant exposure time offset of an arbitrary e.

Drift is a property of one a camera that advertises to record at some speed and actually works at some other speed. Similarly for cameras A and B, a constant d is added for each frame index i:

$$t_{Bi} = t_{Ai} + di \tag{11}$$

Jitter is the random offset $j(i)$ between frames of an ideal camera A and a camera B:

$$t_{Bi} = t_{Ai} + j(i) \tag{12}$$

For an actual video camera, drift and jitter are negligible; offset originates from starting the recording at a different time and from camera's internal processing before the recording actually starts. If the cameras take pictures only when instructed, such as machine vision cameras, a common clock generator becomes another source of error. Many consumer cameras support a live preview feed when connected to a computer; the feeds can be useful for previewing but jitter would be too significant. If the offset between two cameras is not an exact multiple of the frame speed, they effectively cannot have any frames that would display exactly the same scene.

Frame accurate offset sync with clapperboards works by matching the clap sound to visual cues in the video, a method well known in film production. Cameras that record audio to the same video file do not need visual cues and can be synced together by matching the sound streams. Unsynchronized shooting still leaves an offset of at most half a frame between the camera sequences; this most significant sync issue is illustrated in Figure 19.

Professional video cameras can be synced to a single clock generator, so that they all operate on the same frequency (i.e., no drift) and phase (i.e., no offset), called *genlocking* or *generator locking*. Similar method is used when shooting with machine vision cameras that have external trigger input [39]. Synchronizable camcorders are expensive, and consumer-grade hardware usually lacks all possibilities for proper sync.

2.4 Digital camera types

This section quickly reviews the most common types of digital cameras available. More in-depth survey on properties important in reconstruction are given in Section 5.2.

Camera category overview *Consumer cameras* have good availability, but they are targeted for artistic photography and basic users. They are divided roughly in two categories: *compact (pocket, or point-and-shoot) cameras* and *system cameras*. The presence of a mirror divides system cameras in two: *digital single-lens reflex cameras (DSLR)* and *mirrorless interchangeable-lens cameras (MILC)*. A MILC lacks a separate viewfinder and the reflex mirror, resulting in smaller size. System cameras typically support an interchangeable lens and a number of auxiliary hardware, such as electronic flash units, remote shutter releases and power adapters to replace batteries.

Industrial cameras for *machine vision*, targeted for engineering applications with high repeatability, are more controllable and contain no unnecessary bulky parts, but are more expensive and inconvenient to set up, and may need proprietary control tools. Their product cycle is typically longer and thus more reliable than in consumer electronics. In industrial cameras, the different parts of the system are typically defined more accurately in terms of, e.g., lens quality, sensor and pixel size, sensor noise, processing speed and communication protocol.

Camcorders, video cameras with recording capabilities, also vary from consumer to professional grade devices. Camcorders are not quite considered in this work because of their lower resolution relative to still cameras; many of their benefits lie in user interfaces designed for different purposes in video. Many still cameras support acceptable video recording.

Pixels Camera sensor resolution is declared in millions of pixels, or megapixels (MP). Sensor resolution and image size are related via *pixel size* that is the physical size of one photosite on the sensor. Pixel size is described as the physical length of one side of a photosite; virtually all sensors today employ square pixels. System cameras have largest sensors and pixels, and pixel size typically decreases as the sensor size decreases, as manufacturers pursue for high resolution despite of sensor size. The physical size of the image sensor varies according to use, and ultimately determines resolution, pixel size and light sensitivity. Currently, typical sensor sizes for different camera categories are as follows:

- System cameras use a "full-frame" (36 x 24 mm) or APS-C (1.5-1.6x crop of full frame: 22 x 15 mm), resolution usually over 15 MP with pixel size at about 5 micrometres
- Compact cameras use a small sensor; sizes vary but are approximately 1/2.3 inch (6.2 x 4.6 mm), resolution about 10 MP or more
- Smartphones use sizes between 1/3-1/4 inch (less than 4.8 x 3.6 mm), resolution between 3 to 8 MP
- Sizes used by industrial machine vision cameras vary by application, in 5 to 10 mm range having resolutions from 0.5 to over 10 MP.

Lens quality To benefit from the sensor resolution, the camera objective should have an equivalent or better *resolving power*, provided as a chart of *modulation transfer function* (MTF), which is a function of frequency, or number of black-and-white *line pairs* per millimetre. A lens attenuates the frequencies of the optical patterns, such that at some point, dense line pairs are not resolved from each other but show up as gray. [12, p. 71]

Machine vision cameras and system cameras use interchangeable lenses with often well specified MTFs provided by the manufacturer, while cheaper compact cameras, phones and webcams are designed for a single lens that cannot be removed and has poorer MTF that usually is not provided [42]. Compacts also accompany a *zoom lens* for versatility; zooms are more complex systems and therefore more expensive to reach an image quality comparable to others. Calibration is also an issue with moving zoom elements that may not align to precisely the same setting. Interchangeable lenses have a large selection of fixed focal length ("prime") lenses with less moving parts and typically better optical quality. The lack of zoom of a prime lens is acceptable in a controlled environment, where the distance to the subject can be adjusted as necessary. With DSLRs that have a moving mirror and a shutter, the mechanical vibration could move the zoom lens each time a little, changing the focal length.

Data storage Storage size and speed and camera processing power in general follows camera price. High-quality motion capture of short performances is aided by continuous burst mode that can achieve anything between 2 and over 10 frames per second, depending on camera model. Professional-level DSLRs and many MILCs can handle over 10 continuous raw full frames per second, while entry-level DSLRs handle around five; compact cameras are generally slower. Machine vision cameras with resolutions comparable to DSLRs also exist, but at a significantly higher price. Overall storage speed is determined by several factors between the sensor, the camera's processor and mass storage. Reading the pictures to a computer can be done via USB bus or a memory card reader; the latter method is not practical, since the card must be removed from each camera and placed to a reader manually. Industrial cameras typically have temporary memory for only one frame that is read out immediately via a bus or transfer the read frame from the sensor directly with no buffering. For video recording, this needs relatively large amounts of auxiliary hardware to save the raw frame feed. Additionally, either fast hard disks or large amounts of RAM for live data storage would be required.

Software support For a number of cameras requiring identical parameters and control, setting the parameters and downloading pictures manually is not acceptable. Major consumer manufacturers provide a software development kit (SDK) for controlling their cameras remotely to some degree, and some support a common standard for file access. For machine vision cameras, manufacturers provide their own software packages, in addition to sometimes specifying that the camera complies to a standard. The standards specify data formats and buses, image streaming, and control methods. Key standards [35, 36] for industrial cameras are:

- Camera Link, partly proprietary serial transfer protocol with special cable and connector types
- GigE vision, standard interface for video data over gigabit Ethernet connection; licensed so that writing open source software directly based on the standard is not officially possible
- IIDC/DCAM, Instrumentation & Industrial Digital Camera or 1394-based Digital Camera Specification, data format standard used in IEEE 1394 (FireWire) cameras; an open specification
- USB3 Vision, a computer vision standard for the USB 3 bus

A standard-compliant industrial camera enables to use readily available software, reducing the need for in-house software and protecting from vendor lock-in. Some of the largest manufacturers that offer standards-compliant cameras are Point Grey, Basler, Allied Vision, and Imaging Source.

Remote trigger For a multi-camera setup, reliable remote trigger is important. It must be possible to signal the shutter release remotely for still pictures and video recording. Industrial cameras are designed to be remotely controllable, and system cameras have input sockets for remote trigger signals. Others have limited offer, mainly supporting infrared remote if anything.

Conclusion Summarizing for the general feasibility of the most popular camera types in the market:

- System cameras with bigger sensors have high light responsivity, excellent configurability with auxiliary hardware available, such as flash units and remote shutter releases, and decent software support.
- Industrial machine vision cameras have both small and medium sensors, support interchangeable lenses and need significant amount of additional signal processing hardware, and custom or expensive software to control. Video synchronization is usually not an issue, and parameter control has more freedom at the expense of more work.
- Compact cameras, designed for the general public for all-purpose point-and-shooting are cheap, have non-changeable zoom lenses and are in general slower. Raw images and external utilities are usually lacking. Sensor resolution may be less than in system cameras, and high-resolution compact cameras perform worse in low light because of smaller pixel size.
- Others (smartphone cameras, webcams) have poorly documented features, cheaper optics and are not configurable properly in general.

3 Static 3D reconstruction

Multi-view stereo reconstruction is one way to recover the structure of a static three-dimensional scene. In this thesis, a multi-view setup with ordinary digital photo cameras is assumed; fundamentally different methods, such as laser rangers or light field imaging, are not considered.

A complete reconstruction pipeline consists of many parts of hardware and software. This section concentrates on the steps of the software pipeline, which is illustrated in Figure 20. The subject is broad and this section aims to only describe the main ideas in a nutshell. For detailed descriptions and analysis, refer to textbooks, e.g., [2, 6, 43].

Images taken with digital cameras are first scanned for distinct features and matched. To uniquely detect the 3D position of a point in a scene, its image location is required in multiple views. Initially, matching points are required also for calibrating the structure of the scene and the cameras. Then, optical distortions are corrected. The correction requires matching points of different views, and as the feature scan can be performed on original, distorted images, this step often follows the first or is done simultaneously. Distortion correction can also be done as the first step if the distortion parameters are measured separately; this is not always the case for instant reconstruction.

Using the matched features, the cameras are calibrated together, resulting in information about how each camera views the scene, and how the cameras are positioned. This calibration is required for mapping the image points uniquely back to 3D. Once the coarse structure of the matched points and the camera system is acquired, it is further refined for best accuracy. Using the obtained camera parameters, a dense reconstruction of subject geometry is calculated, which is often the most time-consuming calculation and the biggest goal of reconstruction. The camera parameters are jointly used to map each pixel seen by the cameras to the 3D scene. Finally, a surface is fitted on the resulting dense point set, and color images are projected on the surface. A surface is more natural than the point-based output of the dense reconstruction, as the points are only a discontinuous model of the scene.

3.1 Coordinate systems and transforms

The previous image acquisition section described how to record a view of a scene with a camera. From now on, *camera* refers to a particular *configuration* for the camera, or *view*, which can even be a single physical camera moved to different locations. The camera is reduced to the pinhole model; distortions caused by the optical system are assumed to be corrected.

The camera is a projective object located somewhere in the imaged scene. Its *intrinsic parameters* model the properties of the projection, but do not take into account the camera location in any global coordinate frame. The *extrinsic parameters* contain the camera location and rotation in another global coordinate frame, often structured as a matrix [2, p. 41]. This section reviews basic transforms whose results are needed in the later reconstruction steps.

Projection Light rays travel through the pinhole camera's aperture to the image plane that is f units behind the aperture, while the camera origin is set to the aperture [2]. This configuration is depicted in Figure 21. The pinhole model (or, *perspective projection*) states that the world point (P_x, P_y, P_z) is projected via direct light rays to the 2D image plane at (p_u, p_v):

$$\begin{pmatrix} p_u \\ p_v \end{pmatrix} = -\frac{f}{P_z} \begin{pmatrix} P_x \\ P_y \end{pmatrix} \tag{13}$$

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 on the other side of the origin where the actual point lies, where the image is not rotated; this can be more convenient to analyse, since the image does not flip. Physically, the projected point in a camera is behind the aperture, on the film or sensor.

Homogeneous coordinates In computer graphics and vision, points are usually described in *homogeneous coordinates* [2, 43, 44]. An extra dimension is added to the interpretation of coordinates, and each point becomes a line that crosses the origin in a dimension one higher than the original. Several vector operations become more convenient to manipulate: for example, 3D translation is embedded in 4D matrices with homogeneous coordinates. The space is scale-invariant; all points (xw, yw, zw, w) $(w \neq 0)$ map to the same 3D point (x, y, z). This freedom of scaling makes many camera matrices defined only up to scale, as will follow.

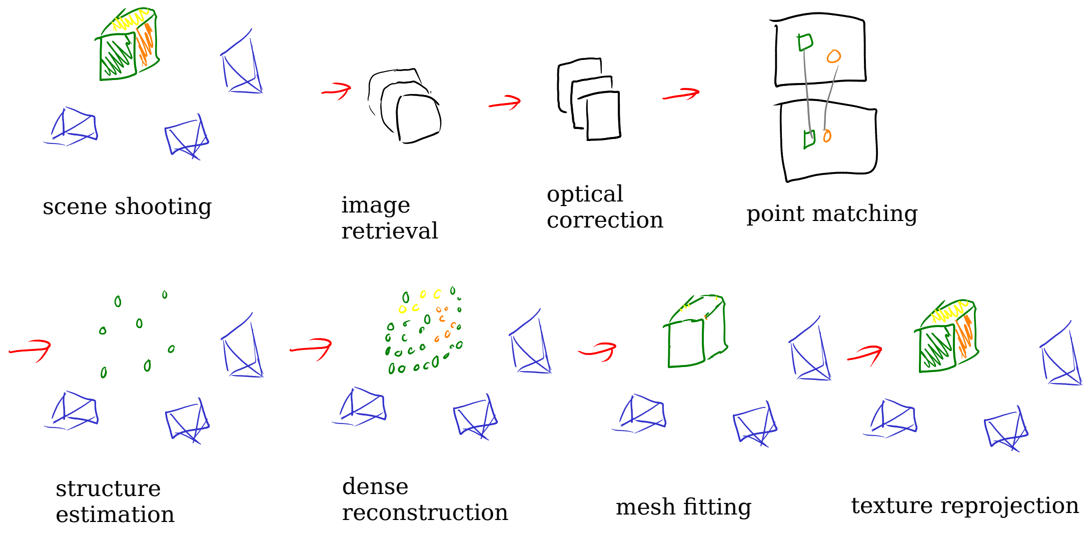


Figure 20: Typical steps in a full structure-from-motion type reconstruction pipeline with no prior calibration information.

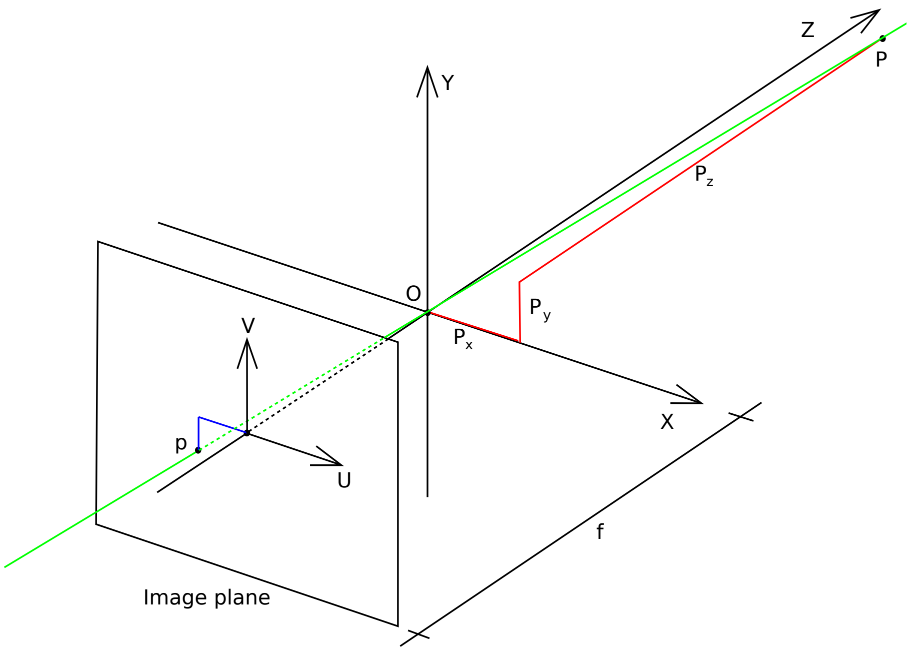


Figure 21: Pinhole camera geometry. Camera coordinate system origin and the physical pinhole at O, axis Z is parallel to the optical axis, U and V specify the image plane axes and the optical axis intersects the image plane at the principal point, the image center. The point $P = (P_x, P_y, P_z)$ projects to p, as well as everything else on the line joining them. The image plane is f units away from camera origin.

Camera parameters The imaging process essentially captures a projection to a flat two-dimensional plane of the camera's view. When comparing points between different cameras that view the same scene, the cameras' relative 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 (many programming libraries may assume this, e.g., OpenCV [16]).

The transformation chain [2, p. 163] is encoded as follows in Equation 14, given a point $\vec{P}$ in homogeneous coordinates (4-dimensional vector) representing a 3D location described in physical (*metric*) coordinates:

$$\vec{p} = \boldsymbol{K}_s \boldsymbol{K}_p \boldsymbol{T} \boldsymbol{R} \vec{P} = \boldsymbol{K} \boldsymbol{T} \boldsymbol{R} \vec{P} = \boldsymbol{M} \vec{P} \tag{14}$$

where $\vec{p}$ is a 2D pixel in a discrete image, and $\vec{P}$ is still in physical units. $\boldsymbol{R}$, $\boldsymbol{T}$ encode the camera rotation and translation (extrinsics) and transform the point to a camera-centric coordinate frame; $\boldsymbol{K}_p$ projects the camera-centric world coordinates to the camera sensor (still in world coordinates), and finally the affine $\boldsymbol{K}_s$ transforms the points from the sensor to pixel coordinates on the digital image (not necessarily yet discretized).

The whole projection $\boldsymbol{M} = \boldsymbol{K}_s \boldsymbol{K}_p \boldsymbol{T} \boldsymbol{R}$ can be used as-is without decomposing it to separate matrices, unless the individual parameters are needed. The matrices $\boldsymbol{K}_s$ and $\boldsymbol{K}_p$ are usually not decomposed but presented as a single intrinsic matrix $\boldsymbol{K}$, and $\boldsymbol{R}$ and $\boldsymbol{T}$ are usually given as a single extrinsic matrix. As the chain consists of several matrices, some of them are defined only up to scale; the coordinate systems' units can be chosen freely.

The internal parameters, intrinsics, encode how the image is formed on the sensor: they consist of focal length f, pixel size (m_x by m_y) and principal point (u_0, v_0):

$$\boldsymbol{K} = \begin{pmatrix} m_x & \gamma & u_0 \\ 0 & m_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \alpha_x & \gamma & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{15}$$

For simplicity, the final matrix where pixel size and focal length are combined is often used. Also, $\boldsymbol{K}$ is often given without the homogeneous part, as a 3×3 matrix. (u_0, v_0) specifies the image center (principal point). The parameters can also be presented separately without the matrix notation. For square pixels, $m_x = m_y$, and for a non-skewed (rectangular) sensor, $\gamma = 0$, which is often the case and assumed by computer vision programs and libraries [2, 6, 43].

3.2 Camera calibration

Calibrating a camera means *measuring* its intrinsic and extrinsic parameters in order to map its data to a known coordinate frame. For only one camera, the camera itself could be chosen to be at the origin and its extrinsic parameters could be chosen as identity transform. Intrinsic parameters are still necessary for knowning the projection. For multiple cameras, one of them can be chosen as origin, the extrinsics of the others are then relative to that. Calibration is not necessarily a manual step before scanning, but is still often performed with a physical calibration object if exact metric calibration is necessary. Self-calibration and structure from motion techniques can recover camera calibration from a static scene automatically [2, 45].

It is sometimes convenient to store intrinsics and extrinsics separately if the intrinsic matrix is constant for several pictures, for example. The intrinsics need then be calibrated beforehand only once. For a camera rig that does not change, whole intrinsic and extrinsic calibration can be retrieved beforehand and applied on further scanned datasets for direct reconstruction.

Manual calibration is done with a known pattern, such as a planar checkerboard pattern [46, 47] or manually selected point pairs. By knowing the geometry of the calibration object and the matching structure in the projected images, the projective transformation performed by the camera is obtained. A known pattern is fast to detect and results in correct physical units when the physical pattern size is given.

The checkerboard calibration step is often programmed to measure optical distortion at the same time [16, 48]. A single image of a three-dimensional calibration object is also sufficient for single-camera calibration [2, p. 181].

One general way for calculating the calibration is *direct linear transform (DLT)*: the whole matrix $\boldsymbol{M}$ is solved from $\vec{x}_i = \boldsymbol{M}\vec{X}_i$ by constructing a system of equations from the projections of some known points i, and minimizing an error metric, as the case is practically overconditioned and the measurements contain errors [2].

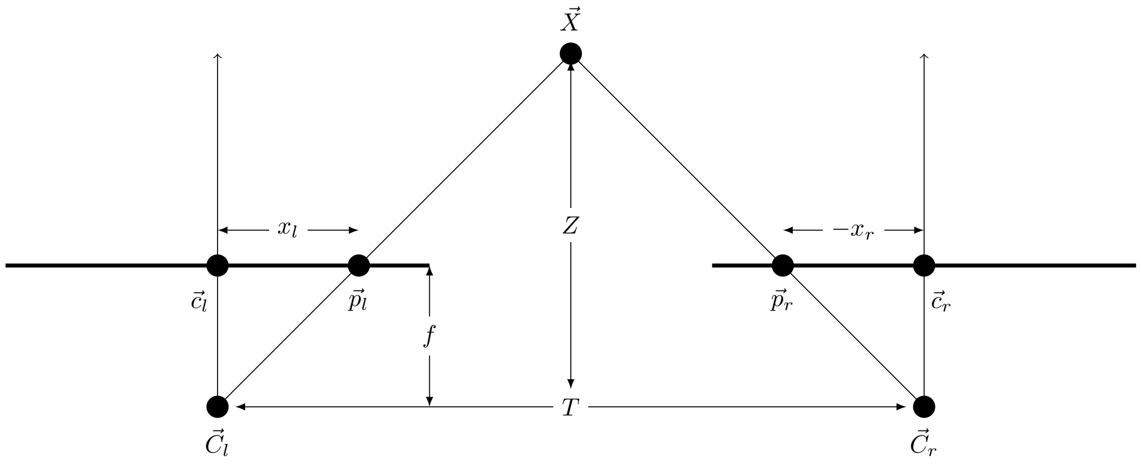


Figure 22: A stereo setup, pictured from above. The image planes (thick lines) are imaginary, as a real film in a camera would exist behind the camera origin and project the image as rotated around principal axis, as described earlier in Section 2.2. The coordinates are given in units of the world coordinate system, common with the camera origins. The symbols $\vec{C}$ are the camera origins (T units between each other), $\vec{c}$ the principal points, x the one-axis image plane coordinates of $\vec{p}$ w.r.t. the principal points, and f is the focal length. The unknown is Z, i.e., depth of point $\vec{X}$.

A practical video-based method has been proposed by Svoboda for self-calibration [49]. The method uses a number of synchronized frames and a laser pointer in a dark calibration volume to automatically find corresponding points, not requiring any precise objects.

Structure from motion (SfM) recovers full calibration for a larger amount of cameras with given matching point pairs, with no special requirements on the subject. It can be used to calibrate an arbitrarily large unknown set of cameras automatically. SfM is described further in Section 3.4.

3.3 Calibrated stereo vision

Given two or more views of a same scene, depth for a point can be extracted by comparing the positions of the point in different views. Whole-picture depth extraction results in a depth map imaged from a single view, giving a depth value for each pixel in a picture, resulting in a set of three-dimensional points seen by that view. In order to compare the point positions in two images, given a point in one image, its matching pair must be found on the other. In the following section, this search problem in finding corresponding pixels, mapping them so that they can be compared, and the comparison that results a depth value is formalized.

3.3.1 Binocular disparity

Assuming a planar setup with identical cameras (same focal length and sensor) with parallel optical axes as visualized in Figure 22, a 3D point can be triangulated given its known projection on both cameras. From similar triangles with a common vertex at X, the depth Z is obtained as follows:

$$\begin{aligned} \frac{Z}{T} &= \frac{Z-f}{T-x_l+x_r} = \frac{Z-f}{T-d} \\ ZT - Zd &= ZT - fT \\ Z &= \frac{fT}{d} \end{aligned} \tag{16}$$

The disparity d is defined as the difference of the points in their image planes, $d = x_l - x_r$. Note that $x_r < 0$ as it's to the left, towards to the negative axis, from the corresponding plane's origin. It is not possible for d to be negative.

As Equation 16 shows, depth is inversely proportional to pixel disparity in the images. 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 units in d, also the pixel size should be taken into account in scaling x_l and x_r. All of these are encoded in the camera parameters: the pixel size is part of the intrinsics, and the baseline is encoded in extrinsics. Algorithms such as those in OpenCV [16] can compute depth from disparity images based on the difference between each pixel. The disparty image is a depth difference of each pixel value, which is calculated based on a correspondence search for each pixel; more general camera calibrations and this search are discussed in the following.

3.3.2 Epipolar geometry

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 as in the stereo setup described above. *Epipolar geometry* [7, ch. 7.3] 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 similar triangulation as in the simple stereo setup [2, 7]. As seen in Figure 23, a 3D point $\vec{A}$ seen by the left camera $\vec{C_l}$ could lie anywhere on the line between $\vec{C_l}$'s origin and $\vec{A}$, because a line passing through the principal point always projects to a point (the camera position is equal to the principal point, i.e., the pinhole). This line is seen as a single point $\vec{A_l}$. From another view, seen by the right camera $\vec{C_r}$, this line equals to some line on the right image plane. The matching point must be on that line. The inverse applies for any point on $\vec{C_r}$ and a line on $\vec{C_l}$. These special lines on the image planes are called *epipolar lines*, and they all coincide at a special point called *epipole*, $\vec{e_l}$ and $\vec{e_r}$ (on the same plane as, but not necessarily inside, the projected image). For example, the line from $\vec{C_l}$ to $\vec{A}$ projects to the line from $\vec{e_r}$ to $\vec{A_r}$ on the right plane, and likewise, the line from $\vec{C_l}$ to $\vec{B}$ projects to the line from $\vec{e_r}$ to $\vec{B_r}$, which can be easily seen from the triangles in the figure.

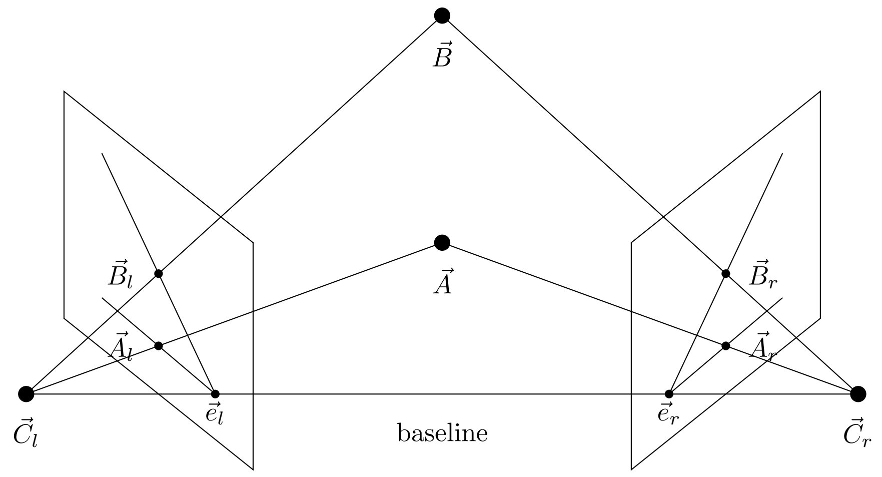


Figure 23: Two camera views on same scene. World points A, B project to planes of different views imaged from C_l and C_r on the left (A_l and B_l), and to the right (A_r, B_r). The actual images cover some area of the total projection plane, and the epipolar point may or may not lie visible in the image. 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).

Essential matrix [2] describes how the camera poses differ, to match one image's point in another. When $\vec{A}_l$, $\vec{A}_r$ specify the points in Figure 23, and the baseline difference is a vector from $\vec{C}_l$ to $\vec{C}_r$, it holds that

$$(\vec{A}_l - \vec{C}_l) \cdot (\vec{C}_r - \vec{C}_l) \times (\vec{A}_r - \vec{C}_r) = 0 \tag{17}$$

as all the vectors are coplanar. The cross product yields a normal to the plane, which is perpendicular to all the vectors, and then the dot product equals zero. In practice, the coordinates are given within the camera coordinate frames:

$$\begin{aligned}\vec{X}_l &= \vec{A}_l - \vec{C}_l \\ \vec{X}_r &= \vec{A}_r - \vec{C}_r \\ \vec{T} &= \vec{C}_r - \vec{C}_l\end{aligned}$$

Then, Equation 17 becomes

$$\vec{X}_l \cdot \vec{T} \times \vec{X}_r = 0. \tag{18}$$

Essential matrix $\boldsymbol{E}$ is a matrix form of this relation, and it provides a standard foundation on the epipolar geometry. It includes the relative rotation and translation of the two cameras in a single matrix:

$$\vec{A}_l^T \boldsymbol{E} \vec{A}_r = 0 \tag{19}$$

Fundamental matrix [2, ch. 11] is a similar object to essential matrix, as it also 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. The described vectors ($\vec{X}_l$, $\vec{X}_r$) are in normalized camera coordinate frames, but from the photographed images in actual image processing, only pixel values can be measured. Thus, fundamental matrix encodes also intrinsic parameters. Given pixel coordinates $\vec{a}_l$ and $\vec{a}_r$ that represent the points $\vec{A}_l$ and $\vec{A}_r$ as seen from cameras $\vec{C}_l$ and $\vec{C}_r$, respectively, the fundamental matrix $\boldsymbol{F}$ is given as:

$$\vec{a}_l^T \boldsymbol{F} \vec{a}_r = 0 \tag{20}$$

The essential and fundamental matrices can be seen as a more general configuration of the triangulation in Figure 22 and Equation 16. The matrices can be estimated by forming a system of linear equations using Equation 19 or 20 and at least eight point correspondences, using some method to solve the equations, e.g., the *eight-point algorithm* [2, p. 155]. Having estimated the matrix, the epipolar geometry and thus the 3D position of any point with two matching pixel coordinates are easily recovered [2, p. 162].

3.3.3 Features and matching

Previously, fundamentals for reconstructing a 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* [6, ch. 4]. Matching is often also called *correspondence* searching: Given a pixel in one image, which is the corresponding pixel in another image taken from the same scene? Pixels correspond to each other if they represent the same physical point. Matching pixels for the previously introduced methods are found in the image space by comparing pixel intensity of some fixed window. To describe the point's visual characteristics and to efficiently compare the characteristics of different points, the surrounding environment is typically encoded as a *feature*, an easily recognizable property vector [6, ch. 4].

For successful matching, and correct geometry, images from different views should represent the same object. Different views must be imaged at the same time, and be free of visible view-dependent artifacts, such as specular highlights. Artifacts may introduce outliers if matches are detected wrongly, or holes in the geometry if no matches are detected at all.

Features Edges or corners are the typical features that are detected automatically at specific steps of reconstruction. They are essentially high-frequency information in the image that can be interpreted as a 2D discrete function; thus, *edge detection* can be performed by a discrete high-pass or band-pass filter, zeroing all but those pixels where a high difference is found [50]. Another popular method is *blob detection*, based on, e.g., the laplacian of a Gaussian-filtered image. Blob detection is typically performed in computer vision using difference-of-Gaussian-based (DoG) methods due to its faster execution than in Laplacian-of-Gaussians [6, p. 152]. Edge and blob detection methods work as a basis for several feature matching and detection algorithms.

Scale-invariant feature transform (SIFT) [51] is a well-known algorithm for local feature detection and description. A fast GPU implementation is also available [52]. After detecting keypoint candidates with blob detection, SIFT encodes the local pixel neighborhood as a spatial histogram of gradients in the 2D image domain. Invariance to scaling and rotation in feature descriptors such as SIFT make them useful in describing features that can be matched between unaligned images.

Search Searching for well distinguished features in an image is called *sparse* search. A sparse set of feature matches is suitable for reconstructing initial calibration, as all the feature vectors can be compared against each other relatively quickly. For finding a complete depth map of the whole image, a slower *dense* search is performed, using the obtained calibration. Dense matching runs through each pixel of the search space and tries to find a matching pixel from another image, e.g., with template matching [53]. The surrounding pixel values themselves inside a *window* centered at the searched pixel are compared in the search area, which typically runs along the epipolar lines only.

3.3.4 Image rectification

In order to triangulate a 3D point from two photographs, the location of the point in both images must be known, as described above. Given a single pixel in one image, its corresponding pair would naively be searched by looking at every pixel in another. Rectification is a process that simplifies this search problem by restricting the search to a single dimension using the epipolar lines [7, p. 157] [54, ch. 7.1].

A common *planar rectification* projects the images on a common plane and rotates them such that corresponding lines are axis-aligned (horizontal or vertical) in both images [2]. By virtually aligning the cameras such that their images are coplanar and their axes parallel, the search only has to be performed on a line that is parallel to the line connecting the camera centers. In Figure 24, two cameras are situated at arbitrary positions, viewing different directions, and their view images are projected on a common plane, where they no longer are rectangular. In practice, a new computer image is created that contains the projected image, and the search can be done along a single axis, or the search can be performed in the original image in the direction of the epipolar line. Other more complex methods, e.g., polar rectification, can handle more general camera configurations [54].

3.3.5 Multi-view stereo

Multi-view stereo (MVS) is stereo vision using more than two cameras. In the research literature, MVS varyingly refers to both the big picture of a whole reconstruction pipeline using several pictures, and also just the dense reconstruction after system calibration has been done. Reconstruction using a large set of cameras has two major choices: One way is to extend the two-view geometry theory further to many cameras, recovering the whole subject at once using many views simultaneously. Other very common method is to arrange the cameras in binocular pairs,

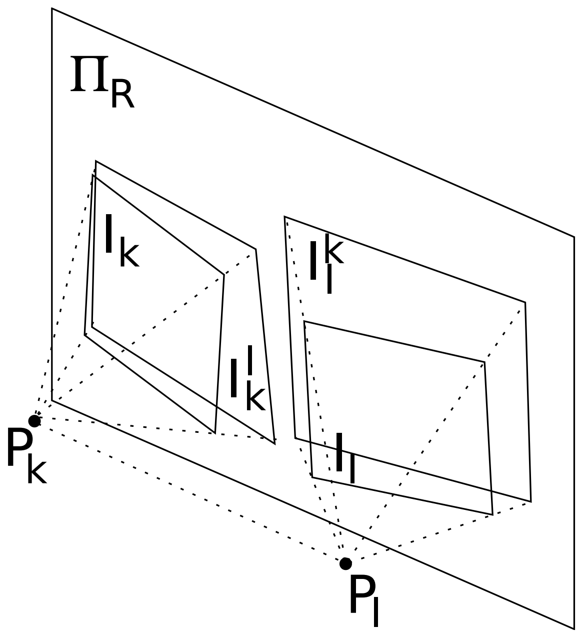


Figure 24: In planar image rectification, two images (I_k and I_l) are projected to a common plane Π_R as I_k^l and I_l^k respectively. The common plane is parallel to the baseline vector $\vec{P}_k - \vec{P}_l$. Image by Pollefeys [54, p. 66].

reconstruct each pair separately to obtain a depth map from each pairwise view, and finally merge the resulting points to a single set. Many of the following studies build a point cloud first and apply Poisson surface reconstruction [55] to recover a surface.

Okutomi and Kanade [56] noted that a short baseline suffers from low precision and a longer one requires a larger space in searching and matching. They approach the issue by combining several stereo pairs with different baselines, making correct point selection simple by using a minimized sum of errors. A clear disadvantage of this approach is the increased number of cameras required.

The method by Fitzgibbon et al. [57] assumes a restricted geometry using a turntable with only single-axis rotational motion. A single camera only is required, but the method does not extend to general cases. The setting is comparable to a number of cameras photographing the subject arranged around it; surrounding geometry must first be removed.

Bickel et al. [58] acquire a 3D face mesh surface with a commercial solution by 3dMD [59] and then apply basic triangulation methods on markers to track a deforming face and its wrinkles. Their method uses a hybrid of fast low-resolution cameras for tracking and slower high-resolution cameras for recovering fine details. Additional computation is done on post-processing the generated mesh.

Beeler et al. [60] perform a dense reconstruction in pairs using image pyramids and cross-correlation in block matching. Normals on the points are computed using finite differences on disparity maps, and a surface mesh is computed using Poisson reconstruction. The mesh is augmented with pore-level details assumed from lighting properties on surface microgeometry smaller than possible for normal disparity computation.

Bradley et al. [61] use seven stereo camera pair arrangements directed to different parts of the subject, and combine the resulting 3D point clouds into a single one for meshing and animating with optical flow. The geometry is tracked by using optical flow to compute vertex positions between successive frames. An initial mesh is computed and manually refined, which is then modified preserving the vertex connectivity. Too fast movement not suitable for optical flow is tracked separately with a priori knowledge of the subject, such as special constraints on a human mouth.

Ghosh et al. [32] compute a single output directly with no merging of intermediate point clouds by assuming that the subject is a deformed cylinder. The cylinder is subdivided in small parts that individually deform based on the captured data, which easily preserves the subject structure.

The large-scale approach by Snavely et al. [62] presents methods and user interfaces that are not restricted on pre-calibrated or pairwise cameras, but instead are designed for unordered image collections. The method presents only the structure from motion solution and is suitable for thousands of views.

Toldo et al. [63, 64] present an algorithm with visibility constraint filtering on a point cloud combined from a disparity map on each camera, with no requirements on camera structure such as pairing. The point cloud is converted to a mesh with Poisson surface reconstruction.

A global *surface element* (*surfel*, *patch*) approach is used by several researchers recently [65–68]. The surface is not interpreted as a simple point cloud, but as a surface consisting of small elements, each visible in several views. Visibility constraints can then be employed based on occlusion between the elements and the outliers can be filtered away effectively.

The patch-based method by Furukawa et al. [66, 69] collects initial correspondences from all source images, and expands 3D patches to a dense set iteratively spreading current matches to nearby pixels, finally filtering them by visibility constraints globally.

Energy minimization with graph cuts applied in computer vision [70] is also increasingly used in dense reconstruction [67, 68, 71, 72] for labeling the depths in a discrete voxel grid or a triangulation.

3.4 Structure from motion

The *structure from motion (SfM)* problem, also called *structure and motion*, uses the scene feature matches between different views to determine both the scene structure and the camera motion between the views [54, 57, 62]. The camera does not have to physically move; the method can be used for multi-camera reconstruction with a static rig. Before SfM, the images must be undistorted and features matched; the algorithm works on point locations instead of input images. While SfM retrieves the scene structure and camera parameters, a dense reconstruction usually follows it as only a fraction of the points in the images need to be considered [54].

SfM begins with two reference views that are used to retrieve an initial reconstruction, and extends each new view into the set iteratively. A *bundle adjustment* method is used to globally optimize the scene parameters [73]. Many reconstruction programs use structure from motion as an initial step to determine calibration and proceed then to dense reconstruction.

The initial stage builds a reference frame and structure for the selected features (points) in two cameras. The structure is determined with triangulation as in Section 3.3.1. In practice, perfect reconstruction cannot be achieved due to noise, and the points are found by minimizing a reprojection error to the extent of a specified threshold limit.

After a coordinate frame is built as a basis using two views, the next camera views are iteratively added to obtain an initial estimate of the camera poses and the selected scene feature positions. Scene features can be matched between only last few views if the camera motion is known to be continuous; the matches can also be done between all possible image pairs, which is useful if there is overlap.

As a last step, bundle adjustment globally optimizes the *bundles of light rays* from the scene to all cameras. Several algorithms and implementations are available, while the basic idea and the underlying problem is the same: a cost function is used to optimize jointly all camera parameter estimates such that the reprojection errors are minimized through all views. Pollefeys [54] gives the following formula to minimize:

$$\min_{\boldsymbol{M}_k, \vec{P}_i} \sum_{k=1}^{m} \sum_{i=1}^{n} D(\vec{p}_{ki}, \boldsymbol{M}_k \vec{P}_i)^2 \tag{21}$$

when the goal is to find the projection matrices $\boldsymbol{M}_k$ and the 3D points $\vec{P}_i$ for all views k and points i, where $D(\vec{a}, \vec{b})$ is the Euclidean distance between $\vec{a}$ and $\vec{b}$ and $\vec{p}_{ki}$ the known, fixed matching 2D image position in the kth image for the unknown 3D point $\vec{P}_i$. In practice, this nonlinear optimization problem is approached in iterative means. According to Wu [74], bundle adjustment should be performed using a Levenberg-Marquardt or Preconditioned Conjugate Gradient method.

An incremental algorithm linear-time in number of the cameras has been suggested for structure from motion for large-scale reconstructions [74]. A specialized scanning rig in this work is not considered large-scale. Large-scale refers to hundreds or even thousands of images of a huge geometry, such as an outdoor building or terrain.

3.5 Error metrics

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 the matching original point, the *reprojection error*. [2, p. 95]

Furthermore, if the scene structure is known beforehand (e.g., as a calibration object, whose physical properties are measured), the resulting quality of the structure can be compared to ground truth. Ground truth data can also be obtained with a real 3D scanner, such as a laser ranging device.

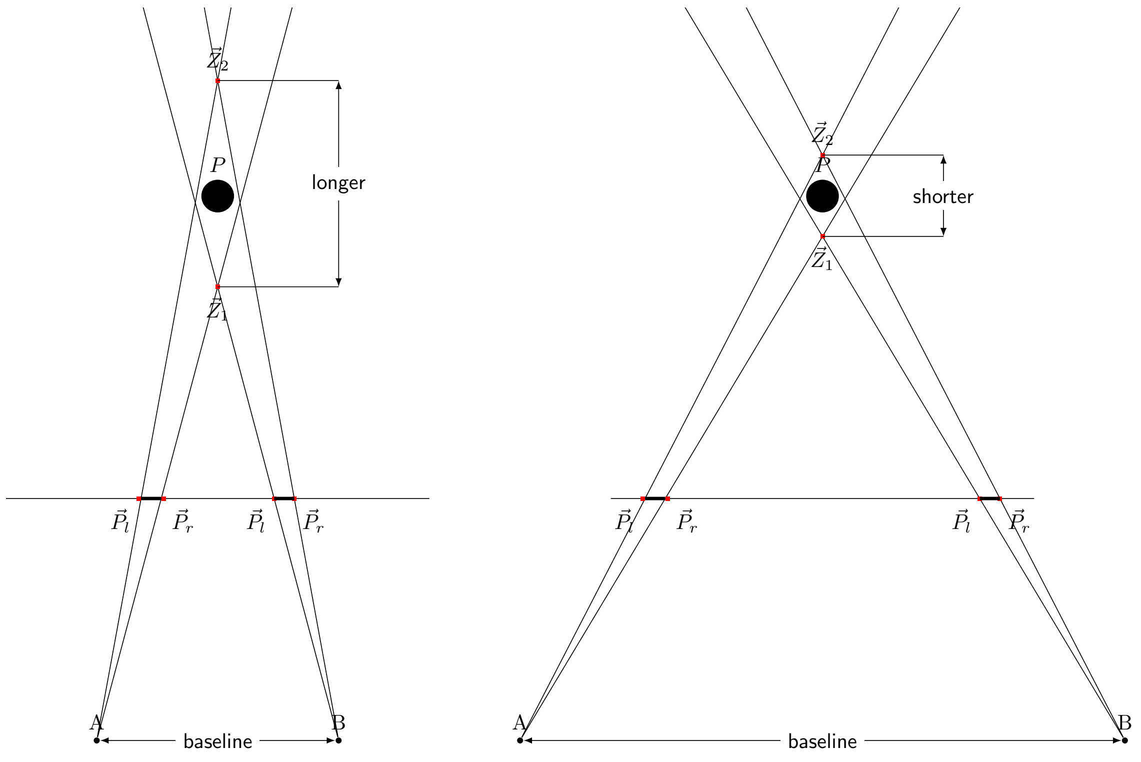


Figure 25: The effect of baseline difference $||\vec{A} - \vec{B}||$ on the depth error $Z_2 - Z_1$, with a small (left) and large (right) baseline. Point $\vec{P}$ is seen by the two cameras A and B. Points $\vec{P}_l$ and $\vec{P}_r$ depict the left and right possible points on the raster image, i.e., the possible range for the pixel.

In addition, uncertainty in depth depends more heavily on the pixel errors (i.e., disparity precision) if the baseline is short, as depicted in Figure 25. The ability to discern the position of tiny features in images is some absolute value in pixel units. When the disparity approaches the scale of this uncertainty, it becomes increasingly difficult to extract an accurate value for depth. Given an absolute uncertainty Δd for the measured disparity d, the error

$$\frac{\partial}{\partial d} Z \Delta d = \frac{\partial}{\partial d} \frac{fT}{d} \Delta d = -\frac{fT}{d^2} \Delta d \tag{22}$$

clearly increases quickly as the measured disparity d becomes smaller.

When sparse feature matches are searched for matching picture points in different views, a number of features are first generated for each picture, and the matches are tested against picture pairs. False positive matches should be filtered out; common way to handle feature outliers is Random Sample Consensus (RANSAC). Random subsets of the sample space are iterated, and samples that do not fit well to a model (i.e., the camera parameters) that is constructed for a smaller set are ignored. The iteration that matches most samples is selected. [2]

Practically, the scale of the camera sensor size and focal length in physical units (millimetres) are often known, making conversion from pixel units to millimetres

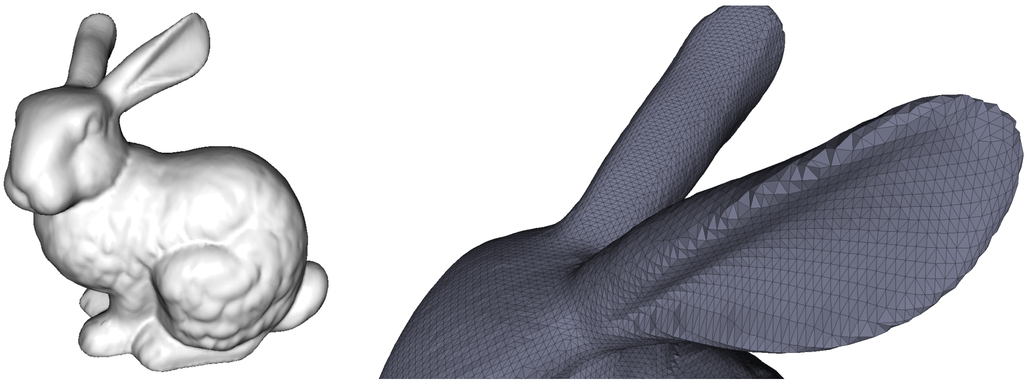


Figure 26: Stanford Bunny, a standard 3D test model by The Stanford 3D Scanning Repository, Stanford Computer Graphics Laboratory, scanned with a laser ranger. On the left: the whole model, shaded with a simple Phong lighting model. On the right: a closeup of the left ear, showing the edges of the faces as black lines connecting the vertices.

possible. Units originating from reconstruction of raster images can be simply scaled to millimetre units, resulting in more intuitive error interpretation.

3.6 Surface fitting

A multi-view reconstruction results in a discontinuous set of three-dimensional points, with no *connectivity information*, i.e., interpretation of *surfaces*. In reality, all the pixels are projections of nearby surfaces, and a more natural and standard representation is obtained by fitting a surface on the points.

Computer graphics typically uses data structures based on piecewise surfaces [1]. The surfaces are described as collections of small patches with a polygon outline. When ultimately rendering the polygons on a computer screen using the graphics processing unit (GPU), pixels inside them are interpolated to contain all color information available in the source photographs.

3.6.1 Data structures

A *(polygon) mesh*, a data structure often used to describe 3D models, consists generally of *vertices* connected by *edges*, forming *faces* of objects [1]. The meshes often consist of triangles, which is the simplest possible polygon. Polygon meshes have a *topology*; the connectivity information of vertices on a surface, describing neighbours of each point. Surface *normals* describe the orientation of the polygon patches. Normals are important in rendering the surfaces realistically in any given lighting environment. Figure 26 depicts the vertex structure of a typical 3D model.

A *point cloud* is a loose term for a disorganized set of 3D points with no topology information, i.e., a cloud consists only of a list of vertices, with possibly some attributes such as colors or normals. Point cloud is a natural output format for 3D reconstruction, as each point in the cloud maps to some pixel in a source image.

3.6.2 Geometry

Assuming that the recovered 3D point cloud represents the surface of an object, the next step of interest is to recover the surface topology from the point cloud. The point cloud is all geometric information that is available, but the images probably consist of more pixels than the number of points in the point cloud. To work with the generated model and to render it properly with colors, it is desirable to recover the topology information.

A popular method for surface reconstruction from a point set is *marching cubes* [75]. The algorithm builds triangles in a divide-and-conquer approach based on whether points are inside or outside a cube in a hierarchial grid. A number of surface recovery methods have been proposed that utilize marching cubes as a final step. The algorithm of choice is currently *Poisson surface reconstruction* [55, 76].

Before fitting a surface on the points, the point data set should be processed either manually or automatically to remove outliers, such as wrongly detected points or geometry of the surrounding scene not part of the object itself. Specular highlights and other artifacts on the surface may also introduce outliers. Manual work in a point cloud editing software is a simple way to delete the points that are not part of the actual object. If manually filtering the subject would be undesirable, e.g., when recording many expressions on a human face in a constant environment or rendering the scan in realtime, automatic methods could be used, e.g., nearest-neighbor or point grouping methods [77].

3.6.3 Texture reprojection

When a polygon mesh is available, it should be rendered in color, including the areas between the original points. The de facto method in computer graphics is to uses *texture mapping* to color pixels in between the vertices, when the number of vertices is less than the number of pixels on the screen. A *texture coordinate* is set for each vertex in the mesh to *map* a smaller image from the texture on to the particular polygon. [1, 78]

Building the texture is done in a similar way of rendering the image on a screen or capturing a scene with a camera: The reconstructed mesh surface is projected on the raster images, registering the image coordinates on each vertex [6, p. 610] [43, p. 98]. When the mesh is viewed from another viewpoint, the pre-projected texture data is used again. Great accuracy can be acheived if a MVS structure is combined with a laser scan, by registering the point clouds generated by both together, using the laser scan as geometry and registered camera poses to project textures [79].

When several cameras share the same view, the correct texture must be chosen on each vertex. The selection can be taken based on some goodness criteria, e.g., the image with largest viewed area or highest brightness.

To optimize the triangle rendering resources of GPUs, the deepest level of geometry is typically colored using texture maps describing not color, but *surface roughness* and *glossiness*. Accurate lighting models make heavy use of highly spatially varying surface *normals* and microgeometric details, instead of the absolute position of the geometry. Such information is typically kept in a *normal map* or a

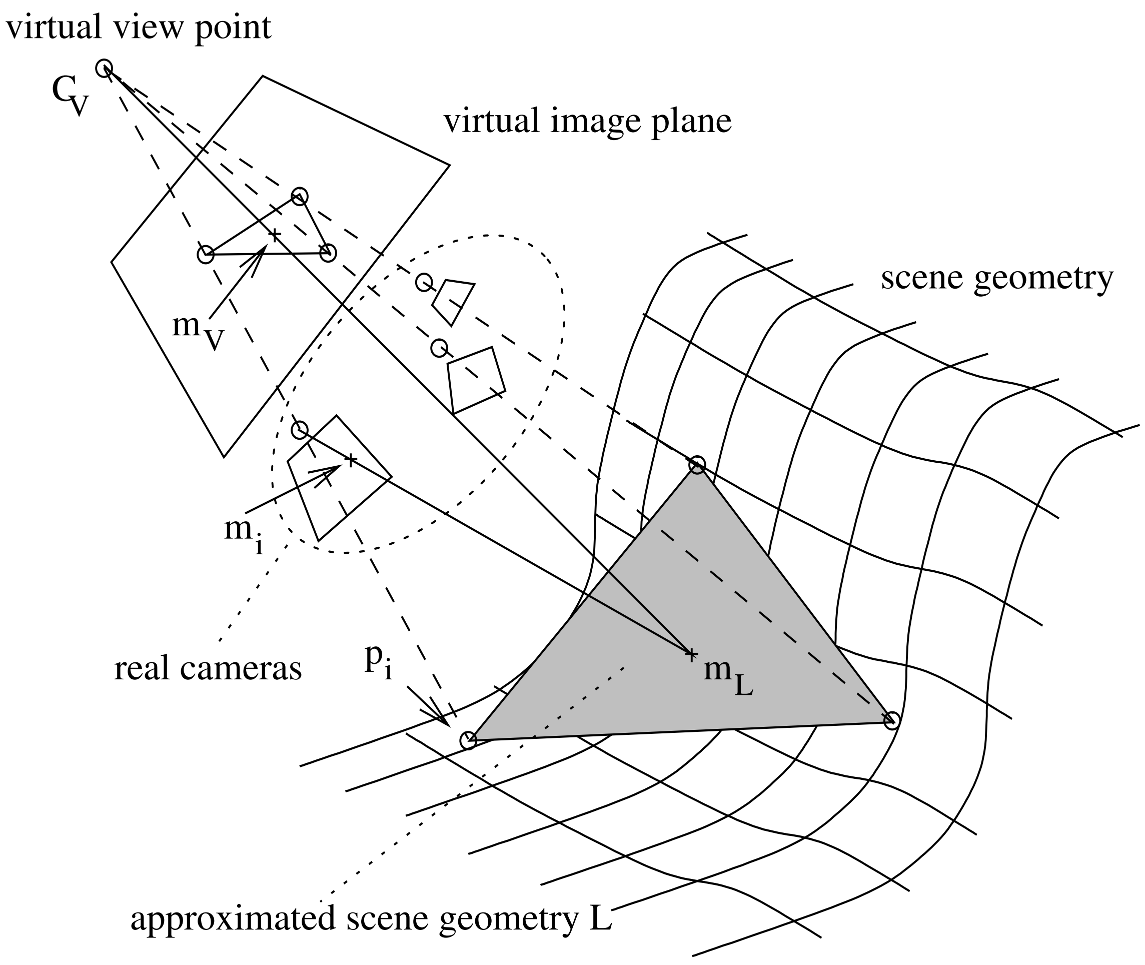


Figure 27: Texture mapping a triangle from a color image on a virtual image plane on a computer screen as m_v. The triangle geometry is mapped on the color images, or textures, of the real cameras as m_i, and the triangular patch between the image vertices is stretched on the rendered triangle. Image from [43, p. 98].

bump map, encoding the surface normal or roughness at each visible pixel. Several studies have presented methods to augment the base geometry by adding wrinkles as normal mapping [58].

4 Motion capture

Motion capture (mocap, or mo-cap) is the practice of recording object movement over time. A recorded subject can be described as a piecewise rigid object, e.g., a human, possibly encoding also some skin movement. In computer animation, the movement data is encoded as positions of certain vertices at each recorded point in time, and often post-processed to be parameterized by joint angles or muscular actions [80, 81]. *Surface capture* is a related term referring to a similar motion recording but happening on a deforming surface instead of globally moving rigid bodies.

4.1 4D data capture

The term *4D* in the reconstruction context refers to temporally varying 3D data. Analogous to the case of traditional 2D video consisting of separate discrete frames of pixels, a sequence of dynamic 3D data is, in a simple case, individual "frames" of point clouds.

For source data in the form of video files per camera, effectively a sequence of sets of synchronized pictures, each set of frames can be considered a static geometry on which the reconstruction would be done. A naive alignment to previous or first frame is possible to average out the object movement if only its local surface movement is of interest. Still, reconstructing each point in time separately would not necessarily produce the same topology for the mesh. Each frame would produce a different point cloud not connected to the previous reconstruction, unless some algorithm that takes this into account is used.

Possibilities to correct the lack of coherent geometric structure exist. Typical methods include fitting each frame of the mesh to another pre-recorded or pre-modeled mesh [58, 61, 82, 83] and matching to the previous frame iteratively, or using intermediate keyframes for later animation [84].

4.2 Geometry tracking

The applications of dynamic reconstruction require tracking of individual points or objects in order to usefully handle the moved subject, as the applications typically require a known model whose movements are recorded. A new model for each frame would not be suitable. Non-static human motion capture in video gaming or movies, where post-processing time is available, can rely on manual work to perfect the quality. In realtime applications, the full frame is typically not reconstructed again, but only selected keypoints that are tracked in 2D are triangulated, and a previously scanned model is deformed based on them.

However, the simplest tracking method 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 topological coherence in the time domain, but recomputes its work on each new point cloud.

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 well to non-rigid deforming subjects [82, 85]. Starck et al. [86] use a different approach with the same topology-preserving idea by subdividing a sphere and mapping it on the scanned subject. A similar method is used by Ghosh et al. [32] using a cylinder instead of a sphere. Common application also for the entertainment industry is *animation retargeting*: using a completely separate character, and deforming it on each frame based on the current state of the scanned object, fitting the model's vertices to the scanned set [87].

Sometimes the computer animation must be optimized for feasible playback, such as reducing the amount of data required, or making it easier for humans to adjust the animation by parameterized models. Parameterized models may include a base template pose and spatially varying deformations such as muscle models warping groups of vertices [81], or completely separate keyframes such that the animation is interpolated as linear combinations of different poses [80, 84].

4.3 2D surface tracking

In two dimensions, the dense reconstruction step can be skipped when using only features in image space, providing real-time performance [88]. Assuming that a particular object stays in the imaged volume, features can be detected and mapped to a point on the 3D surface, and tracked locally with no full-image reconstruction, while the tracked object is assumed to keep the same topology as a previously scanned template model.

Classic motion capture uses special reflective markers that are tracked in 2D with infrared cameras. Distinctive bright markers are easily thresholded from the background, tracked, and triangulated in real time. Markerless surface capture detects image features that are then used as virtual markers, working in a similar way. Matching textured objects is more computationally intensive, and needs high resolution cameras [89]. Markerless capture is important in scanning facial movement, because time-varying texture is significant in highly dynamic and detailed deformations, such as wrinkles from different facial expressions [61, 84, 90].

Tracking is done by looking for the same feature as in a previous frame in the neighborhood where it was previously. State-of-the-art 2D tracking methods include Kanade-Lucas-Tomasi (KLT) [91, 92], Tracking-Learning-Detection (TLD) [93], and Consensus-based Matching and Tracking (CMT) [94], all based on *optical flow* [31, 95, 96]. Given source and destination images, optical flow is the apparent motion in the scene between the images, given as a motion vector for each pixel. Estimating the flow is based on the brightness constancy constraint: the intensity of the source and destination pixels is the same [30].

Optical flow is also used in interpolating new frames between recorded frames for non-synchronized cameras [41, 97]. The assumption of stereo reconstruction that the cameras encode the same scene can then be virtually achieved by interpolating the motion vectors between two frames to obtain an intermediate frame that then temporally matches that of another camera.

4.4 3D registration

Aligning two meshes or point clouds together is called *registration* in three dimensions [98, 99]. Registration finds a rigid transformation from one object to another. For a moving subject, the method can be used to find the object movement between the frames [100, 101]. Registration is also used for combining different, overlapping parts of a static subject generated from different viewpoints to one larger model if the number of cameras is not enough for complete surface coverage [102, 103].

The basic idea in registration is to fit two surfaces or point clouds together so that the shapes that they present become as close as possible to each other. The practical method of choice is iterative closest point fitting (ICP). Given a sufficiently close initial guess, ICP iteratively finds a rigid transformation that minimizes the distance between every point in one mesh and its closest match in the other. In tracking applications, an identity transform is a good initial guess for the algorithm, because the tracked object has presumably moved only a little. Variants of the original ICP apply assumptions on real-world data and errors, to improve convergence speed and quality [98, 99]. Advances to the rigid ICP has been proposed to address surface noise and other issues, by applying ICP locally and allowing the surfaces to warp [104].

5 3D scanning rig implementation

Now that the background has been introduced, the practical part of this thesis follows. Specifications for a 3D scanning rig are given and different off-the-shelf camera types are compared more in depth. Suitable cameras for the task are then selected and described, along with the decisions on mechanical support construction. Control software for the rig is developed, and a survey is given on the readily available offering of reconstruction software.

5.1 Functional specification

Constraints on the rig were specified in loose terms, leaving detailed work on the background study. Furthermore, not all features of a general-purpose reconstruction rig can be addressed by the mathematical background only. Desirable properties of the system were given as:

- Ease of use and practicality: the user should not need to know the implementation details (hardware internals or algorithms) or to have programming skills to calibrate the system and acquire a 3D scan; scanning should be as simple as pressing a button
- Flexibility: for research purposes, the rig should be applicable to different types of subjects
- Mobility: the system should not pose requirements on the environment it is set up in
- Detail: high image resolution should be used for achieving data for state-of-the-art algorithms
- 3D and 4D: possibility for both static surface reconstruction and variations in geometry over time are preferred
- Open design: by documenting the rig well and selecting readily available generic components, a similar system could be built if there was such a need, and the already built system could be easily extended
- Reasonable price: the total budget of all hardware and software combined should not exceed 10 000 EUR.

Flexibility for arbitrary subjects is achieved by using adjustable connections for each camera, such that their poses can be changed individually. Distance to the photographed target should not be too short in order to not disturb human subjects, but on the other hand, a large setup may be difficult to build and would consume space unnecessarily. When capturing video, the frames should be synchronized among cameras as described in Section 2.3, which might not be perfectly possible with consumer hardware. A mobile rig should be light enough to carry, set up, and reconfigure for new subjects, but it should be rigid enough to give proper quality pictures and hold its calibrated configuration during use. Cameras are chosen among pocket cameras, system cameras, and industrial machine vision systems currently in the market.

5.2 Camera comparison

Section 2.4 presented the typical properties of different camera categories in general, and compared them to technical requirements. In this section, focus is more on the feasibility of the properties on reconstruction, and on selecting a collection of feasible cameras fitting in the budget.

General requirements Small close-up subjects in the range of a metre or less introduce concerns in shallow depth of field, forcing the lens aperture small using a large image sensor, and consequently, requiring bright lights or slow shutter speeds. For automatic capture using multiple cameras simultaneously, reliability in configurations and controls is important. Due to the large number of cameras required, practical features are also necessary to take into account, such as availability of external power supply instead of batteries. For static photography, an external shutter release tool is required for the cameras for simultaneous shooting. Furthermore, it must be possible to download the imagery from the cameras directly to a computer.

With a fixed budget, maximizing the scanned surface coverage of the subject requires to strongly consider the camera price, as long as the minimum required features are covered: with more cameras, a larger subset of the subject surface can be photographed, resulting in a more complete model. Cheaper compact cameras typically lack in lens robustness, raw image formats, remote configuration and manual modes. More expensive models have comparable features, and their prices rise into the same range as with DSLRs that are more understood in the field. In the other end of the spectrum, machine vision cameras have the best flexibility but may require customized, vendor-specific tools and are expensive. All the properties should be looked as a whole; e.g., large pixel count does not necessarily imply better pictures if the MTF of the optical system is poor.

Previous work DSLRs have been popular in commercial static capture setups [105–109] probably due to their price compared to specialized hardware, and software availability. Commercial video capture setups may use specialized machine vision devices and hardware-accompanying or customized software for image capture and reconstruction [110]. Also standard commercial video cameras have been used [61], but their resolution is limited compared to still cameras. As an example, Borshukov et al. used professional Sony/Panavision HDW-F900 video cameras for the movie The Matrix Reloaded [111] producing uncompressed video, with a list price exceeding $80000 [112]. Machine vision cameras have also been used where synchronization was especially important [58, 65]; such cameras have also limited resolution.

Bus type	max. theoretical speed	full-HD FPS
Camera Link [36]	5.44 Gb/s	218 Hz
USB 3.0 [36]	3.2 Gb/s	128 Hz
CF card [116]	1030 Mb/s	41 Hz
GigE [36]	1.0 Gb/s	40 Hz
IEEE 1394 [36]	800 Mb/s	32 Hz
SD card [115]	490 Mb/s	19 Hz
USB 2.0 [36]	480 Mb/s	19 Hz

Table 1: Bus speeds and corresponding naive maximum frame rates per second for 1920 x 1080 pixels x 12 bits; actual speeds would be lower

Specialized hardware Self-contained consumer devices contain the image processing pipeline from sensor to mass storage by themselves. In contrast, machine vision devices need several high performance computers for reliable multi-camera capture because of the lack of local storage and image processing pipelines. A machine vision camera contains typically only the imaging sensor and a processor for a control interface, and outputs easily 1 Gbit/s worth of raw data or more. A common desktop computer and one additional hard disk per camera would be necessary if the capture is long enough and would not fit in a computer's RAM, increasing the system cost and complexity. Best mechanical hard disks currently have sequential write speed up to 1-1.4 Gbit/s as measured by Tom's Hardware [113]; newer solid-state drives (SSD) generally are two to four times faster [114]. For covering all the bandwidth, expansion cards would be needed per camera, count depending on the camera protocol. Most PC motherboards can accommodate only a few cards, requiring several PCs.

Transfer speeds given by specifications and theoretical maximum frame rates for typical 12-bit raw full-HD frames are listed in Table 1 ordered by speed; achievable rates would be less depending on bus protocol overhead [35, 36]. Benchmarked [115, 116] write speeds for Compact Flash (CF) and Secure Digital (SD) cards used in consumer cameras are also given for comparison. The write speed of the cameras themselves must also be taken into account.

Preferred properties A camera is required to have several properties, and the system as a whole must be taken as a global optimization problem, since some properties conflict each other. In detail, the technical camera features that should be considered are described in Section 2 and features that vary among different cameras are listed below. Main properties affecting image quality directly are as follows:

- CMOS/CCD sensor.. CMOS suffers from rolling shutter in video mode, while global-shutter CCD is more expensive and rare in larger formats but also more responsive to light.

- Sensor resolution and physical size. Higher resolution covers more detail, until the optical limits of the system are reached. A larger sensor has larger pixels, and is thus more responsive to light and results in lower noise and lower required exposure time. On the other hand, at an equivalent subject distance, a deeper DOF is achieved with a smaller sensor.

- Low noise. Sensor noise degrades image quality in general, adds error to subpixel position estimation and complicates correspondence search, etc.
- Dynamic range, i.e., bit depth of the processing pipeline. Whereas JPEG pictures in some cameras have only 8 bits per color channel, the typically higher bit depth of the sensor is available in raw image formats.
- Lens quality. High sensor resolution is only useful if the lens is sharp enough, and color aberrations degrade image quality in general. Fixed focal length lenses have less moving parts and hold their calibration better. The performance of good lenses is usually given as an MTF, modulation transfer function, describing the contrast.

On practical matters affecting image quality:

- Optical stabilization works by moving a glass element inside the lens or the sensor itself, effectively moving the image on the sensor, having also a negative effect on calibration. Such feature would be turned off as harmful and unnecessary, since the cameras would be mounted rigidly.
- Dust reduction. If enabled for some consumer cameras, it is typically implemented as high-frequency vibration of the sensor on camera startup. Moving the sensor might result in small calibration problems.
- Shutter lag. Lag measured between ordering the camera to take the picture to the actual moment where exposure starts should be consistent, and preferably small.
- Video frame rate, resolution, and compression. Consumer cameras that support a raw format only use raw for still pictures and compress the video. Raw video is seen often unnecessary in the consumer market and poses high requirements on memory space and processing speed. For image processing, each frame should be compressed as little as possible. Some system cameras support writing only full keyframes instead of predicted frames, resulting in higher quality and file size.

On practical matters affecting usability:

- Lens features. While photography lenses typically support electronic and even automatic focusing, machine vision lenses typically have mechanically adjustable focus and aperture. Of photography devices, compact cameras feature an integrated zoom lens, while system cameras feature interchangeable lenses.

- Continuous shooting rate, i.e., the speed at which full-size pictures can be taken. Controlled by many factors, on consumer cameras, image processing speed and memory card write speed have most effect, and on MV cameras, the bus speed typically limits the rate, and sensor speed is selected based on the bus speed.
- Configurability. Aperture, shutter speed and others should be manually controllable remotely, and repeatable without automatic features.
- Remote trigger. Practically all DSLRs support wired or wireless remote shutter releases; some machine vision cameras have optionally an external logic-level trigger input in addition to one that is set by the protocol in the data bus.
- For consumer cameras, USB read speed supported by the camera. Retrieving saved image or video data takes less time on faster interfaces.
- External flashes. Practically all system cameras can be wired to an external flash unit; only some compact cameras support this.
- External power supply. Not needing to charge batteries adds value to the ease of use.
- Weight and size. A smaller and lighter camera requires less bulky support structures.
- Price. With a fixed budget, cheaper cameras can cover a larger area of the subject at a time.
- Availability. For an easily replicable system, it should be straightforward to acquire the hardware.

5.3 Selected cameras

A number of compacts, system cameras, and machine vision cameras were reviewed. In this section, the selected key parts are described in detail. Precise information on all hardware used can be found in Appendix A.

In general, machine vision cameras were considered overly expensive, although they perform best in synchronized video and output the frames in lossless raw format. Compact cameras do not have significant advantages over DSLRs, when comparing models where both have comparable required features, such as external shutter release options and options for external flash units. The biggest advantage of DSLRs over compacts is the proven compatibility with other parts of the system (hence the name *system camera*) and mature support for remote controls.

A DSLR is a good choice for still photos because of moderate price compared to image quality, and availability of usable software and best range of accessories. Video recording is notably worse than with machine vision cameras that can produce raw data, synchronized output, and (typically) higher resolution. Because of

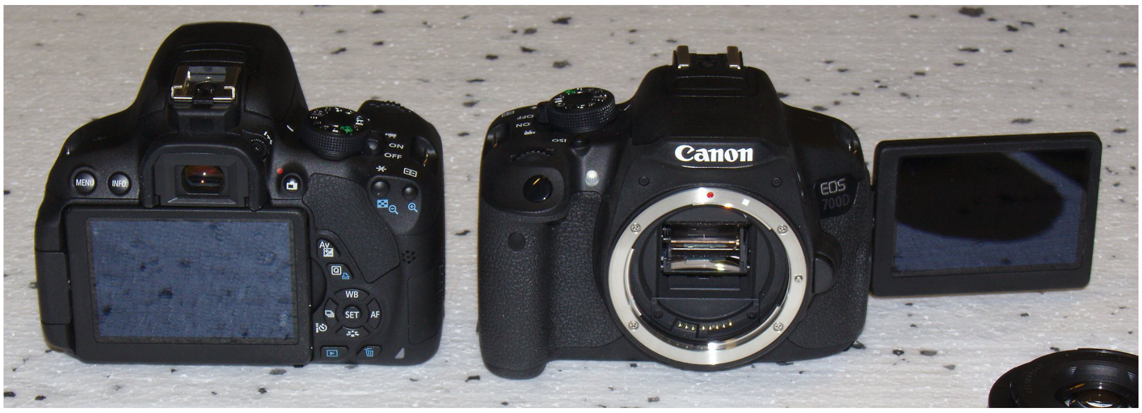


Figure 28: Canon EOS 700D DSLR camera body

Sensor type	CMOS
Sensor size	APS-C 22.3 x 14.9 mm
Pixel size	4.3 x 4.3 µm
Image resolution	5184 x 3456 pixels (17.9 million)
Processor	Canon Digital Imaging Core (DIGIC) 5
Bits per pixel	14
Burst shooting speed	5 FPS
Video mode	1080p in 30 FPS or 720p in 60 FPS
Max write speed	$\approx$ 40 MB/s

Table 2: Canon EOS 700D key features

a price difference of almost an order of magnitude for cameras having a comparable resolution, it was decided to leave high-performance video tracking as a minor feature, and concentrate on static subjects. Canon was chosen over other brands because of firmware customizability [38], previously acquired personal knowledge of the manufacturer, well specified and mature remote control abilities compared to alternatives (see Section 5.6.1), and previously proven work using the same brand [32, 105–108].

5.3.1 Canon EOS 700D DSLR

Canon EOS 700D body, also known as the Rebel T5i, introduced in 2013, is the newest of Canon's consumer range DSLRs at the time of writing, with a price at about 600-700 EUR. More expensive models would not add considerable value to the rig, as their advantages are mostly in durability for practical photography. Key features are shown in Table 2 and the front and back view of the camera are depicted in Figure 28. The built rig consists of nine cameras arranged as a 3×3 grid.

Properties Relatively large pixel count combined to a good lens captures high-resolution detail. The sensor's pixel size is not unnecessarily small but not as large as in full frame DSLR cameras. The 5 FPS continuous speed for full-size images can be used for testing high resolution motion, but the video abilities are also reasonable. Ghosh et al. [32] use DSLRs operating in burst mode for special illumination capture, which presents one augmentation for the rig in future work. The quality can be tweaked with firmware modifications and raw video recording is also possible, at a reduced resolution limited by storage speed [38]. The sensor is labeled “Hybrid CMOS”, a new Canon’s technology that embeds phase-detection autofocus pixels in the sensor for fast continuous autofocusing in video mode. A number of pixels are missing color sparsely around the middle of the sensor and are interpolated; they only introduce artifacts in raw video.

The recent Canon EOS M MILC shares many features and internal elements with EOS 700D and was strongly considered, but it is missing support for remote shooting from USB or wired remote release.

Resolution of the sensor has stayed same since the EOS 550D, introduced in 2010. At the same time, also cheaper new models are available, such as the EOS 1200D; at a reduced price, less processing power is provided, resulting in slower operation but otherwise the cameras share mostly same features.

Storage The camera uses Secure Digital (SD) memory cards for mass storage, supporting the UHS-I standard (Ultra High Speed) [117]. Maximum write speed has been found experimentally to be approximately 40 MB/s [118]. A class 10 UHS-I memory card was selected; manufacturer claims 45 MB/s write speed.

Conveniences The camera also features a tilting LCD screen that has provided to be useful when looking the camera from the front. Additionally, AC power adapters were chosen to operate the cameras without the need for charging batteries. The adapters plug in the camera’s battery holder, providing continuous power at the expense of an additional cable per camera. For wireless remote shutter, the camera natively supports an infrared remote controller that must be pointed towards the camera from the front.

5.3.2 Canon EF 50mm f/1.8 II

Canon’s EF 50mm f/1.8 II lens (priced at about 110 EUR), shown in Figure 29 is well known for its excellent image quality at a low price. The lens was introduced in 1990. The 50 mm focal length with the crop factor of 1.6 of the camera body provides a good shooting distance for the purposes of this work. Like most lenses at this focal length, the lens is nearly distortion free. Despite the poor plastic build quality, its optical characteristics are well known to be of very high quality for the price. The lens changes between autofocus and manual focus modes with a mechanical switch, and the focus ring in manual focus mode is loose enough that it might need to be locked with tape if left on for a long time shooting to hold its position for a calibrated setup.

At a distance of one metre $d = 1$ m, the sensor size of $s_x \times s_y = 22.3 \times 14.9$ mm

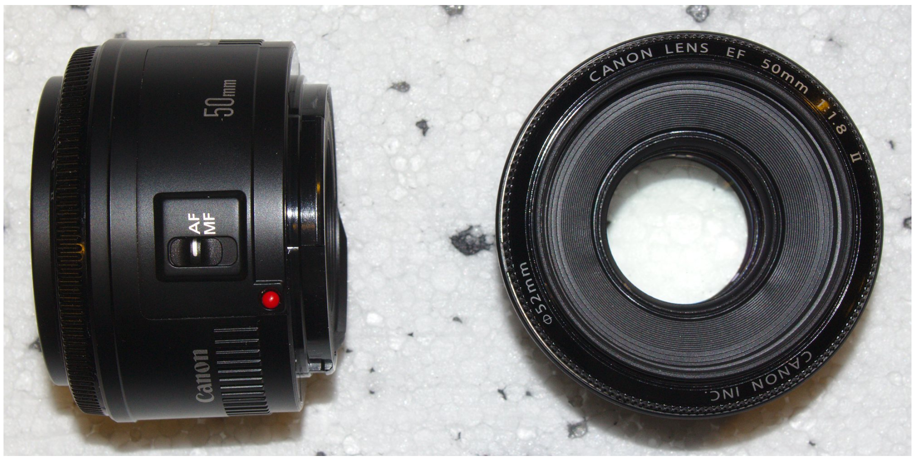


Figure 29: Canon EF 50mm f/1.8 II, a fixed focal length lens with manual focus and autofocus

and $f = 50$ mm focal length, the area that fits in the frame is $a_x \times a_y$

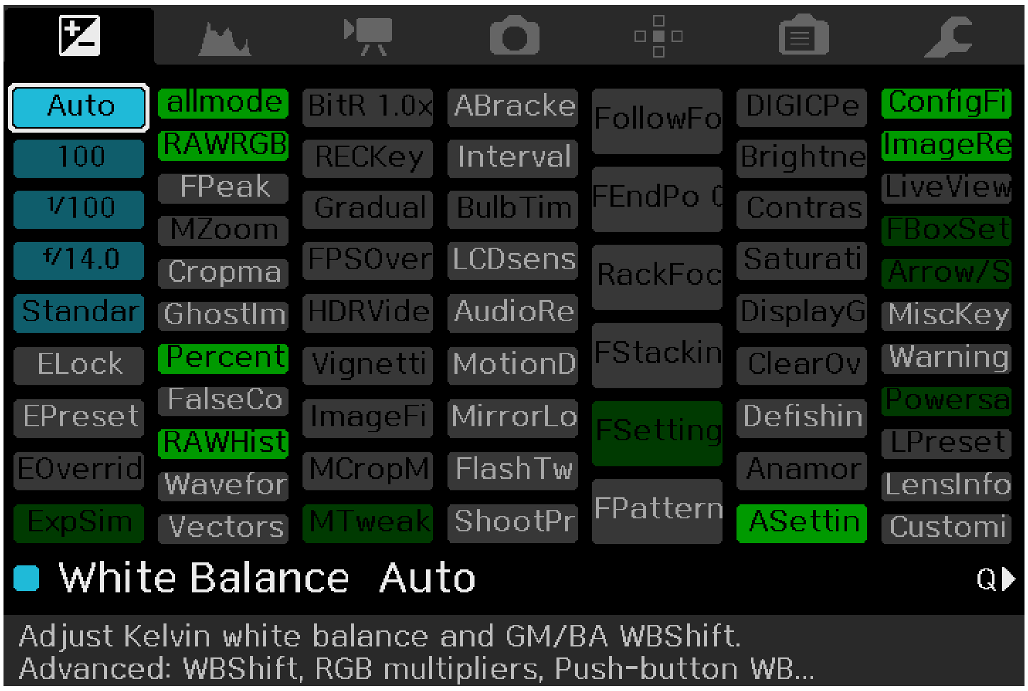
$$\begin{aligned} a_x = s_x \times \frac{d}{f} = 22.3\text{mm} \times \frac{1\text{m}}{50\text{mm}} = 446\text{mm} \\ a_y = s_y \times \frac{d}{f} = 14.9\text{mm} \times \frac{1\text{m}}{50\text{mm}} = 298\text{mm} \end{aligned} \tag{23}$$

which is easily seen from similar triangles with a common corner. From Equation 3, the depth of field (using the d/1500 rule) at this setting is about 16 cm for a f-number of f/11, or 20 cm for f/14, at circle of confusion of 0.019 mm, a reasonable depth for the given area size. This circle of confusion would span 0.019mm/22.3mm×5184px ≈ 4.4 pixels.

5.3.3 Magic Lantern

An important consideration was the ability to use the third-party Magic Lantern (ML) firmware add-on [38]; it was originally developed for improved user interfaces in video recording, but has been extended for numerous other features. It was installed on each camera in the rig. Firstly, it provides experimental support for raw video recording, easier remote triggering for video, and other hacks; additionally, being open source, it can be modified for any custom purposes, such as multi-camera synchronization aids, if necessary. It is a separate native program for the ARM-based DIGIC processor, and runs in the camera alongside the original firmware, loaded from the memory card, using the same operating system and configuration menus [38]. A screenshot of a menu grid displaying some of the additional functions is shown in Figure 30.

The add-on is installed on a memory card, and it does not make permanent changes to the camera's original firmware. Still, being third-party software based on reverse-engineering efforts, there is a small risk of unexpected instability. Magic Lantern has an active user forum [118], a wiki, and an automatic software build system for releasing nightly testing versions. As an open source project developed in a



White Balance Auto

Adjust Kelvin white balance and GM/BA WBShift.
Advanced: WBShift, RGB multipliers, Push-button WB...

Figure 30: Magic Lantern grid menu in movie mode, showing a subset of its features.

distributed fashion, it suffers from inconsistent documentation and coding style, and is occasionally best examined by reading source code, written in the C programming language. No "stable" release is offered for EOS 700D yet, but the unofficial testing version has been long in use.

5.4 Hardware construction

The cameras need additional support structures for mounting. A trivial solution would be a tripod for each camera, which would not allow stacking the cameras vertically, though. Two main methods were considered and are described below.

5.4.1 Mechanical options

With flexibility and mobility as requirements, the system should be built of removable parts that need no modifications to the environment, such as attachments to walls. The separate parts should be small enough for transportation, but large enough for rigidity and build simplicity. Each camera should be able to turn at least in landscape and portrait mode to fully utilize the picture frame for portrait subjects, such as human faces.

Two different major designs were considered: a hinged frame consisting of industrial aluminium profile system, or individual tripod stands available in most photography stores meant for supporting audiovisual equipment. For connecting the cameras, a custom profile could make use of also custom built parts; also, photography stores sell screws and clamps meant for connecting cameras, lights and other hardware.

Aluminium profiles such as those by MiniTec, Item or Bosch Rexroth consist of rectangular tube with a T-shaped slot running along each side for screws. Mechanical dimensions vary by manufacturer but most supply a large catalog of connection brackets, hinges, screws etc. that allow much flexibility in the frame design. A hobbyist-purposed manufacturer OpenBuilds targets open source projects, such as 3D printers and CNC routers. The cost of such profile is around 10-20 EUR/m, with varying prices for connection pieces; availability varies by supplier.

Ready-made light or speaker stands come in a variety of sizes and share a common design with an extendable center rod and three legs. Typical light stands have a universal mount in top end of the rod for fastening a lamp. Speaker stands typically share a similar standard; in general, they are also heavier and thicker to support a larger mass. Most photography and audio/video studio stores offer a variety of light and speaker stands.

A custom aluminium profile system would have its benefits in rigidity and positional flexibility, because it could be built in any shape. On the other hand, one rigid shape would be more difficult to modify when any change were needed. Standard light or speaker stands are attractive because of their availability in normal stores. They also can be moved around, which also means more work in configuring the camera poses again. A stand consisting of mostly round rods lacks flat surfaces and slots that could be used for fastening parts together. For that purpose, special clamps and ball heads are sold separately. Machine vision cameras do not generally use the same tripod screws aimed for consumer market, but are screwed on the frame with custom bases.

5.4.2 Selected design

The method of choice was to use ready-made, heavy duty lighting stands. Millenium LST-310 (a brand by the Thomann music equipment store [119]) is a sturdy three-legged lighting stand with a pole that can be extended several meters high. Additionally, it has a rotating horizontal bar at the top. Similar tripods were used in other projects in the same laboratory and they were found useful and rigid. The relatively heavy weight (approximately 10 kg per stand) makes the support structures steady. Four of these stands were bought, which should allow a moderate coverage around a subject in pairs of cameras.

For flexibility, each camera should be able to rotate in at least to horizontal or vertical pose, with arbitrary aiming position. Manfrotto 494 ball heads [120] were used for this purpose, connected to the tripod rod with Manfrotto 035 Super Clamps [121]. Figure 31 shows a complete connection from a stand pole to a camera.

The total structure of all nine cameras divided in three supports (Figure 32) takes up relatively little space, and is flexible to set up around any human-size subject or smaller. The stands can be carried in bags and the more fragile parts are fitted in two cases shown in Figure 33.

Wires and power adapters are fastened to the stand legs in long-term use to reduce wires running on the floor.

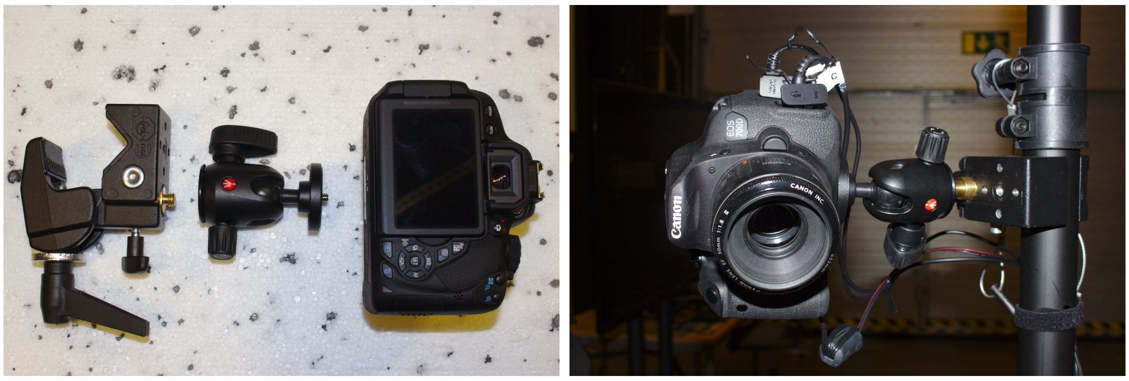


Figure 31: Left: the Super Clamp, ball head, and a camera separately. Right: one camera connected to the pole and its wires.

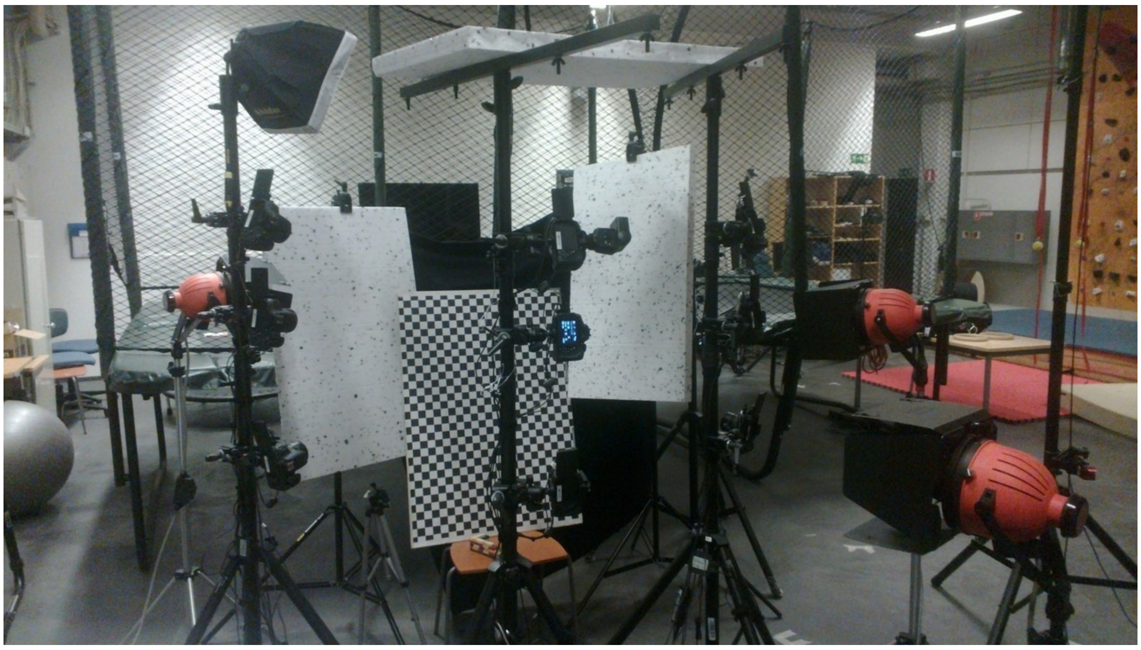


Figure 32: Complete setup with cameras and spot lights for video recording. In typical scans, the cameras would be slightly closer to each other.

5.4.3 Lighting

An even lighting is necessary to capture the subject properly, as described in Section 2.2.5. Unwanted lighting artifacts include harsh shadows and sharp reflections. Most shadows affect the scanned colors in an unwanted way, since the subject is to be relighted in a digital environment where shadows are formed artificially by calculating the brightness in every illuminated pixel separately. Sharp reflections make reconstruction harder, since the position of the reflection depends on the viewing direction, making the appearance of the surface different in different views.

In still photographs, flash units "bounced" from white objects are a practical way to achieve a good lighting. Typical flash units are either bounced away from white

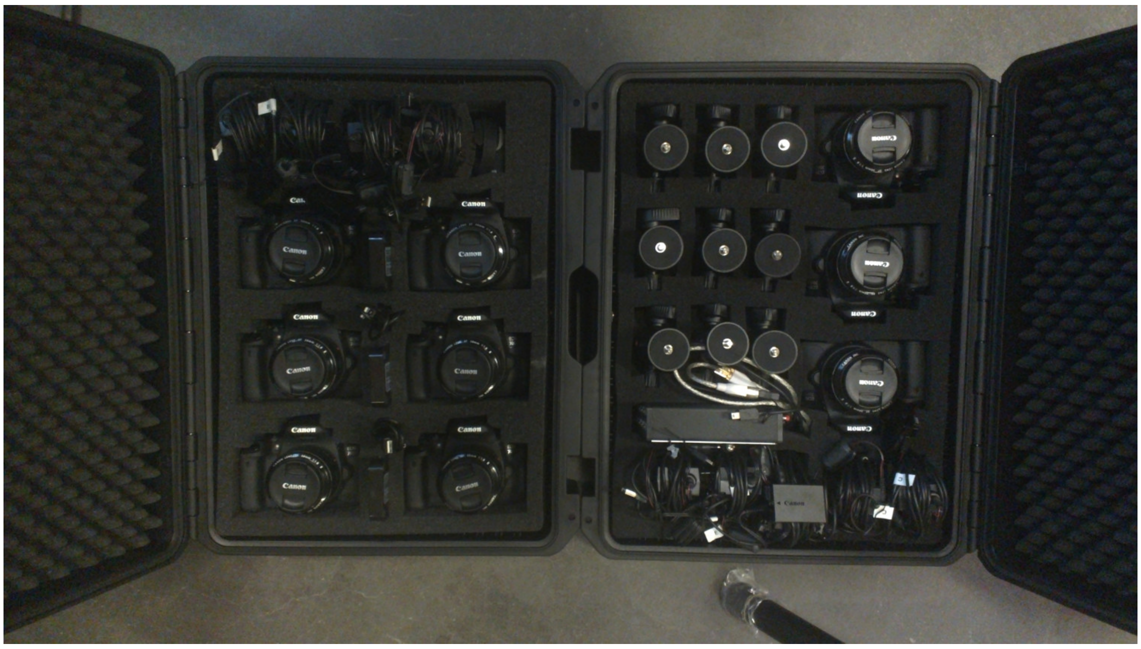


Figure 33: The fragile parts of the system fit in two equipment cases padded with soft foam rubber.

umbrellas sold for this particular purpose, or shot through a large and semitransparent cloth. For video recording, either a continuous light source or a high-frequency stroboscope synchronized to the video frame rate would be needed. Since the selected cameras do not support frame synchronization, a continuous light is necessary.

In the scope of this work, no lights were purchased, as readily available flash units and spot lights in the workplace could be used temporarily to evaluate the amount of light needed. Two standard flash guns attached to the cameras were used for still photograph samples, and three studio spot lights were tested for video recording. Detailed description on the lights is given in Section 6.2.

5.5 Universally synchronized multi-trigger hub

To fully automate the system, it is desirable to trigger the cameras remotely from a computer that also downloads the photos. Additionally, for a rig with possibly non-static subjects but an intended result of a still subject, the cameras have to be synchronized to record the subject at the same instant in time. This section describes the remote shutter mechanism and a custom tool that was built for the task. The cameras are enabled to shoot in arbitrary order, while the most common order would be to shoot all cameras synchronously. An alternative method in synchronization could have been one third-party wireless remote trigger per camera; however, they are expensive, may be not as consistent as a wired solution, and would require modifications to work connected to a computer. Similarly, custom and undocumented solutions previously used for hard synchronization for machine vision cameras cameras, e.g., by Bickel et al. [58] include ready-made USB I/O boards such as those provided by Data Translation [122].

5.5.1 Remote shutter synchronization

The synchronization was implemented as a wired remote trigger for each camera, driven from a single source. Additionally, when using flash units for lighting, the short light pulse emitted by the flash typically synchronizes the cameras if the surrounding environment is dark, and the camera shutter synchronization has less strict requirements. However, the trigger hub was designed to not rely on this fact. In video recording mode, the same remote wire can be used with 3rd-party firmware to start recording.

Canon remote trigger Canon EOS 700D has an input port for focus trigger and shutter release, in addition to the integrated focus and shutter button. The camera uses a standard 2.5 mm stereo jack for connecting external remote controllers. Wired and wireless electronic remotes are available in the market, but no standard devices for triggering several cameras arbitrarily appear to be well available; expensive wireless devices (such as the PocketWizard series [123]) advertise support for multiple cameras using one shutter release, with no well defined latency. Fortunately, the triggering method is simple and widely researched among hobbyists; it is well enough documented, as will soon follow.

The remote release jack is a three-contact connector, where one pin serves as a common ground, and connecting another pin to the ground triggers the camera's autofocus and metering, and the third pin releases the shutter when connected to the ground [124]. The pins supply some current that flows back to the camera via the ground pin. It is safer to separate the cameras from each other electrically instead of connecting the similar remote wires of all cameras together. A common method among hobbyists and even suggested by Breeze Systems [125] is to use opto-isolators to control each camera individually, isolated from the shared control circuit. Breeze Systems sells popular software for controlling multiple cameras simultaneously, and advertises third-party USB controlled relay boards for triggering the shutters remotely as programmed from a computer [125]. However, mechanical relays are slow and their timings are inconsistent compared to transistors or opto-isolators.

Isolation An opto-isolator provides a galvanically separated switch that can be used to electrically "connect" the release wire to the camera's ground such that no electrical signal path is shared between the cameras. The switch is a phototransistor that is triggered externally by the light transmitted from a light emitting diode (LED). Both the phototransistor and the LED are installed in an opaque plastic housing. As an example, the opto-isolator device used in this work is shown in Figure 35 next to the circuit it was used with. The particular device works in open collector setting, exposing the collector pin that is left unconnected ("open") and the emitter pin of the transistor. Another common setting is a powered device that outputs a voltage signal from a power supply.

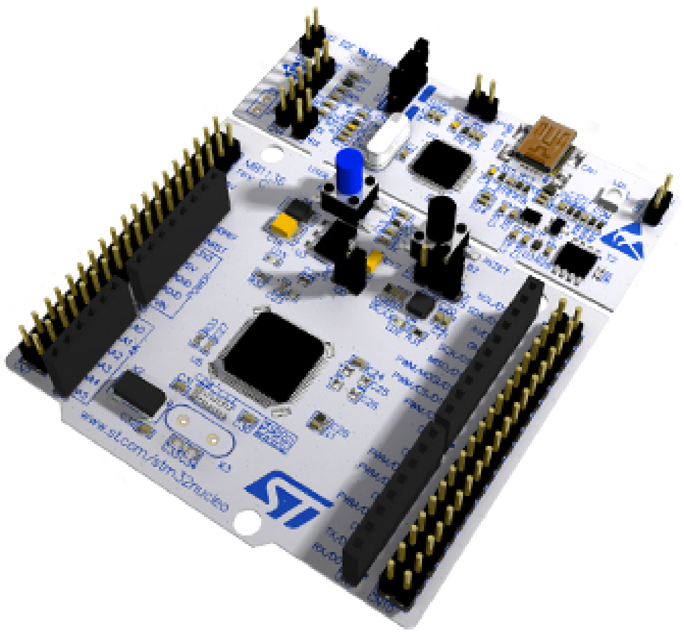


Figure 34: STM32 Nucleo-F401RE microcontroller development board.

5.5.2 Microcontroller platform

A microcontroller (MCU) is a tiny computer in a single package containing a microprocessor core, RAM, nonvolatile program memory and peripheral devices; all parts required to run a single user-defined embedded application typically with no general operating system. The Arduino microcontroller board family [126] is a popular example. The typical programming languages for microcontrollers are C and C++. Communication on the host PC end of a USB connection, when supported, can be implemented in any language.

In this work, a microcontroller prototyping platform called Nucleo-F401RE by STMicroelectronics [127] was used. This platform has a built-in USB port for simple communication.

A MCU was chosen over other alternatives, e.g., a computer-controlled relay board, because of its high customizability and programmability at a low price. On one hand, such a device is more complicated to set up than a simple switch, but on the other hand, it can be programmed to automatically sequence the cameras in any arbitrary order. It can also be used to measure the shutter delay and the actual precise time when each camera takes the picture; each has small unpredictable variations. When capturing moving targets, measuring this lag variation could be of interest.

5.5.3 Hardware

The Nucleo-F401RE (Figure 34) is a prototyping board designed around the STM32-F401RET6 microcontroller IC. This MCU contains a relatively powerful 32-bit ARM core, space for 512 KB of program code, 96 KB of RAM, 51 general-purpose I/O

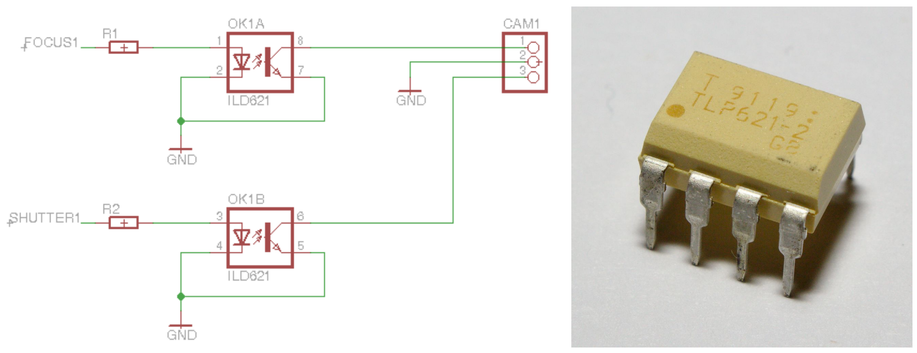


Figure 35: Left: schematic diagram for a single opto-isolator pair for shutter and focus of a camera, including the current-limiting resistors and a pin header connector for the camera signals. Right: TLP621-2 dual opto-isolator in a DIP8 through-hole package, with physical dimensions of approximately 10 by 7 mm.

pins available on the board, and other peripherals such as timers and analog/digital converters. A single-wire debugging (SWD) method is supported to debug the running software in real time [127].

The trigger was constructed for ten cameras, functionally transferring trigger pulses from a PC to all cameras synchronously through opto-isolators and 2.5 mm stereo cables connecting to the cameras. The microcontroller has unused I/O pins for 22 outputs in total after the ten camera outputs, and if individual control is not necessary, the number of connections can be arbitrarily large. For each output, a Toshiba TLP621-2 dual opto-isolator was used, mainly because they were readily available; any similar device would work. An output pin in the microcontroller drives an LED of the isolator, making the corresponding transistor conductive. Schematic of one camera controller is shown in Figure 35. Each LED is driven through a current-limiting resistor of 220 ohms, resulting in approx. 10 milliamperes of LED current with the LED voltage drop of 1.15 V specified by Toshiba datasheet [128]. This results in the transistor conducting the current in the camera remote connection.

The circuit was initially constructed on a solderless protoboard, "breadboard", shown in Figure 36. After proving that the circuit worked, a proper circuit board was built and installed in an extruded metal enclosure. Board layout is shown in Figure 37 and the built case without final lid in Figure 38. Full schematic for the whole circuit is available in Appendix B.1. The circuit was drawn with the Cadsoft EAGLE, a printed circuit board (PCB) design software [129]; complete schematic and layout files are available in `http://github.com/sooda/thesis`.

The circuit board was designed to fit in a recycled enclosure that has inside dimensions of exactly 5 by 3.9 inches (approximately 127 by 99 mm). The used housing has a removable lid, but any box with a height of 35 mm or more should fit to the design.

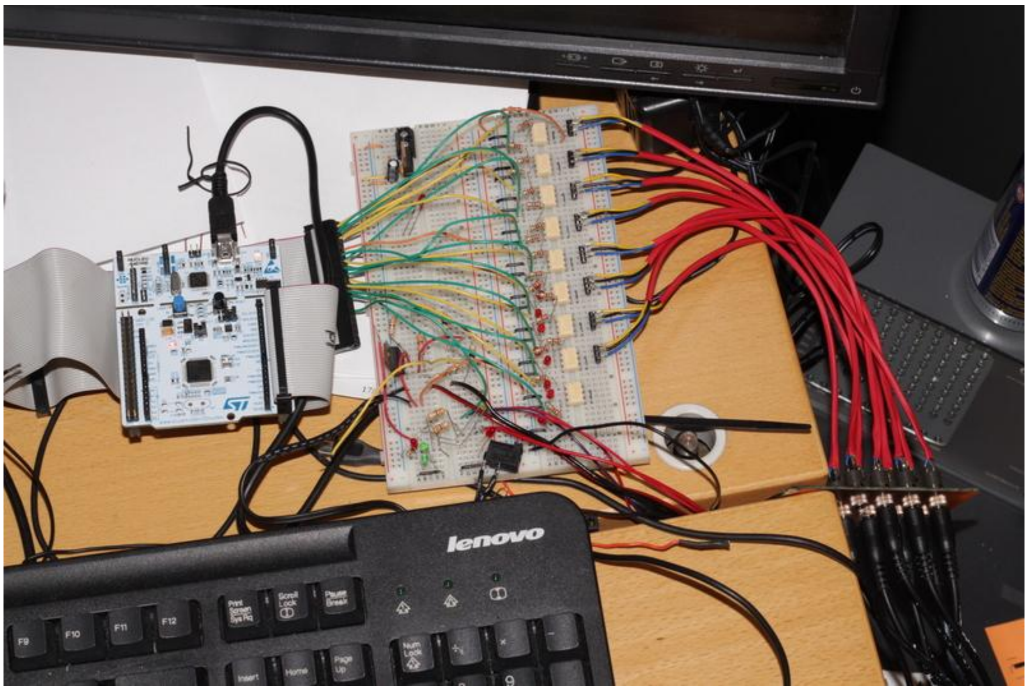


Figure 36: Remote trigger tool prototype on a solderless breadboard.

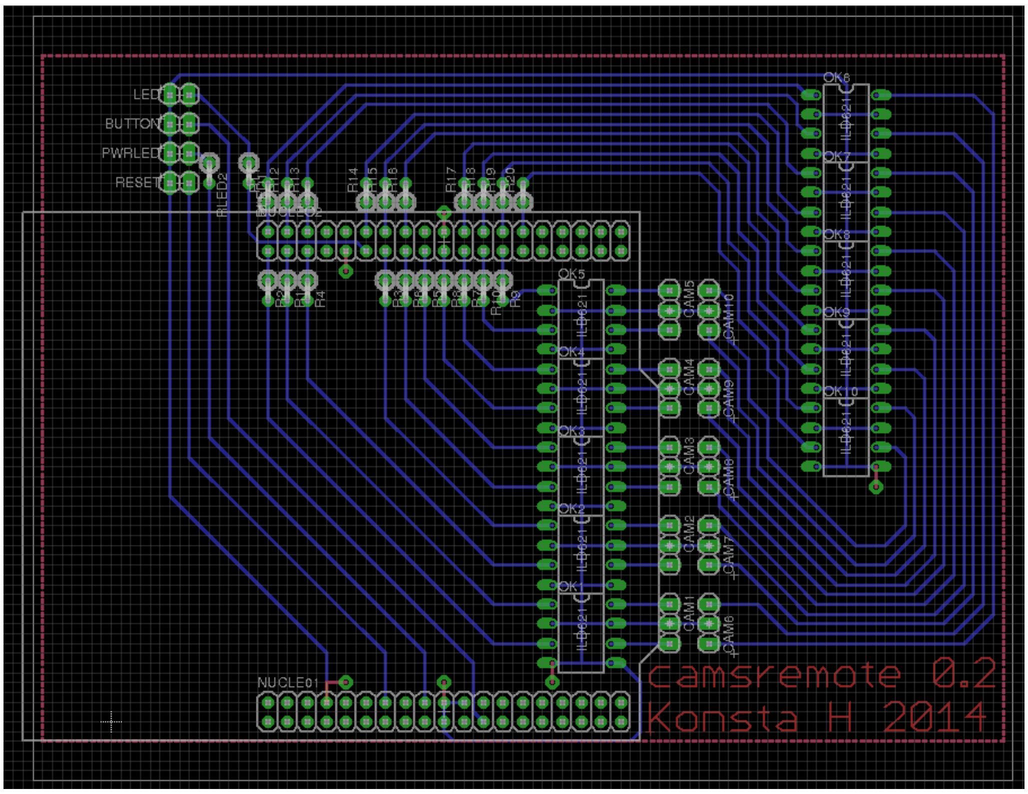


Figure 37: Final version of the printed circuit board layout for the trigger box. Full schematic is available in Appendix B.1.

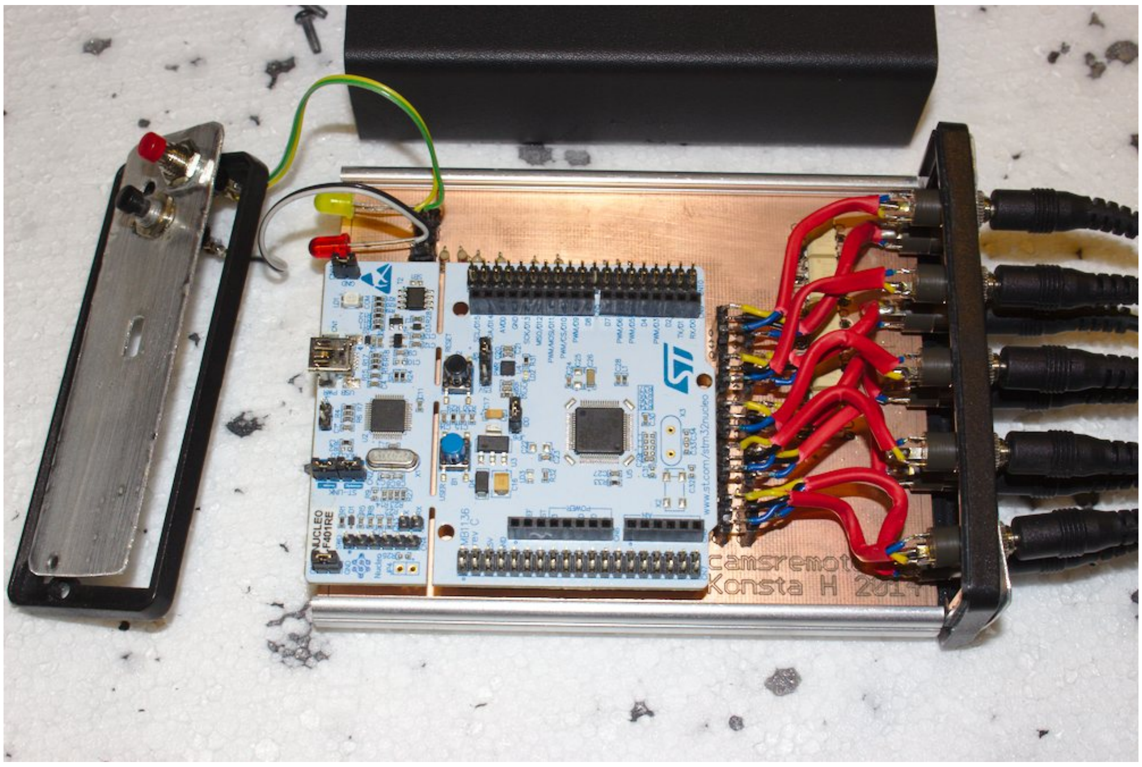


Figure 38: Remote trigger tool in a case, with lid removed. The left side includes buttons, indicator lights and a USB connector; the camera connectors are on the opposite side. The microcontroller board is attached to solderless connections, so that it can be removed later for testing purposes if necessary.

5.5.4 Code

STMicroelectronics advertises the Mbed programming environment and library for the board as an option for writing software for the MCU [130]. The toolchain that compiles the binary for the board works also via a web browser in a cloud service, but the libraries can be also installed locally. All MCU-side code was developed using the Mbed libraries, built locally with the GCC-ARM Embedded toolchain [131]. Mbed hides the hardware complexity of the platform, making the total application source code small (less than 200 lines of simple C++) and relatively independent of hardware initialisation details. A significant advantage in this hardware abstraction is that a novice could extend the program without previous knowledge on the particular hardware architecture.

The MCU software for the trigger hub, or *firmware*, communicates via a serial line that the board passes through USB as a standard virtual serial port, requiring no special drivers. Communication protocol consists of flags for setting focus or shutter mode and entering bit flags that describe which cameras should be affected, in text mode. The least significant, i.e., rightmost, bit signifies the first camera. The device echoes the sent characters back, making it convenient to test with a serial console. The protocol is described in detail in Table 3.

Set focus pin state	Fxxxxxxxxxx
Set shutter pin state	Sxxxxxxxxxx
Reset all to unpressed	R
Example: push down focus on 1st, 2nd and 4th	F1011

Table 3: Remote trigger protocol; the letter x signifies a single bit (digit 0 or 1) and can be omitted from the left, in which case it is assumed 0. Note that the messages set the whole button state instead of sending "keypresses"; to emulate a full keypress, the state must be set first to 1, followed by a set to 0 or a reset. Each command must be finished with a space or a newline character.

The code also monitors the Nucleo's integrated general-purpose pushbutton that is used to focus and trigger all cameras simultaneously. One press focuses all cameras, and subsequent presses send a short shutter signal. The reset button resets the whole microcontroller, restarting the program with no focus or shutter signals selected.

The used Mbed library does not actually turn all pins on at exactly the same time but in fast succession. This is not a problem because of the high clock speed of the processor; the difference between first and last pin changes was measured to be a negligible 1.6 microseconds.

Control software for the host PC was implemented as a small command-line Python program communicating to the virtual serial port of the Nucleo board. The host controls the trigger's outputs individually, normally driving all the outputs at once. Sequential or other forms of triggering are possible, because the cameras are controlled individually in hardware. For example, as each camera is able to record five full-resolution frames per second in burst, interleaving the nine cameras would result in short 45 FPS full-resolution imagery. Also, when using the internal pop-up flashes in the cameras, they can be fired separately to circumvent issues in lighting synchronization or exposure control.

The same MCU can be reprogrammed with another firmware at any time; it was used also for measuring shutter delay of one camera, see Section 6.1.1. The programs are available in http://github.com/sooda/thesis with source code.

5.6 Custom camera control software

Some custom techniques were used to preview images of all cameras simultaneously, configure the settings of each concurrently, and retrieve the captured images from them in order. Popular commercial programs for controlling multiple cameras include Breeze systems DSLR Remote Pro Multi-Camera [132] ($129 per camera) and Kuvacode SmartShooter [133] (600 EUR for unlimited cameras). Both advertise features for controlling camera settings, and remote shooting with a delay of over 100 milliseconds between cameras. Custom tools were designed and implemented for flexibility and for a convenient live preview. Like the MCU board layout and firmware, all programs and their sources can be downloaded from http://github.com/sooda/thesis. The programs have been verified to work under recent Linux and Mac OS X systems, with minor still unresolved issues in Mac OS X.

5.6.1 Camera control library

Camera libraries gPhoto2 [134] is a well-known free and open-source application and library in the C programming language for controlling digital cameras on Unix-like operating systems, supporting over a thousand cameras. The software consists of a command-line control tool of the same name, *gphoto2*, and a library API, *libgphoto2*.

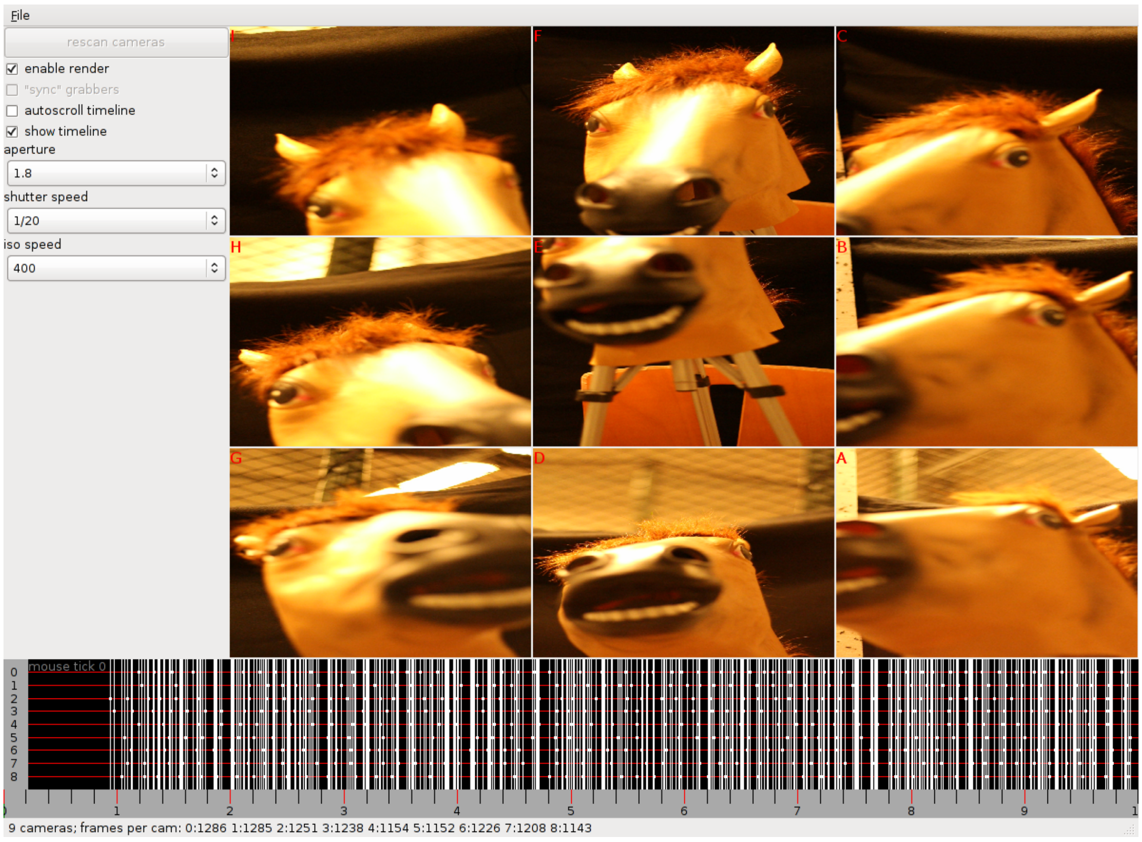
Instead of relying on each camera manufacturer separately for a software development kit, gphoto2 abstracts common operations behind the same interface. For example, Canon and Nikon, some of the biggest camera manufacturers, both provide SDKs for controlling their devices remotely via a USB connection from Windows and Mac OS X [135, 136]. Sony provides an SDK for Android and iOS [137]. Olympus has had an SDK, but it is no longer available [138]. To support all vendors, one would have to write code for all APIs. In this work, extendability to other vendors was seen as an optional advantage. In addition, development kits provided by the vendors may be more restrictive and may not be fully available.

Furthermore, the Canon EDSDK claims in the manual not to support sessions to more than one camera in parallel [135]. This technical issue could probably be overcome with additional workarounds in software, though.

Libgphoto2 implements the *Picture Transfer Protocol (PTP)* [139] for setting properties and transferring pictures via USB. Details on list of configurable properties and sequences for capturing pictures and preview vary among manufacturers, and the library has most thorough support for Canon and Nikon cameras.

gphoto2 wrapper A "wrapper" code library was written in C++ for libgphoto2 to automate memory and resource management, simplify the usage of the raw library, and to write the programs themselves in clean C++; libgphoto2 itself is implemented in the C language and is not well documented. Libgphoto2 exposes the data as objects, and the wrapper simplifies their handling and implements some more complicated operations, such as creating a new camera object instance, configuring parameters, and downloading preview frames and actual shots. Libgphoto2 itself is also not thread-safe: it was discovered that if a camera were used in two or more threads of execution simultaneously, unexpected errors or crashes would happen. Locking mechanisms were written for the wrapper to prevent more than one executing threads from accessing a camera instance simultaneously. The library initialization is also protected, because camera-specific sub-libraries are loaded after first use. Initializing multiple cameras concurrently using gphoto2 only would result in a crash.

The wrapper is written exclusively with libgphoto2 in mind, but as it actually hides the implementation details strongly, it could be ported to use other libraries, e.g., EDSDK, without affecting the actual applications using it.



- [✓] enable render
- [ ] "sync" grabbers
- [ ] autoscroll timeline
- [✓] show timeline

aperture: 1.8
shutter speed: 1/20
iso speed: 400

```
9 cameras; frames per cam: 0:1286 1:1285 2:1251 3:1238 4:1154 5:1152 6:1226 7:1208 8:1143
```

Figure 39: A custom program, *gphotogrid*, was written for displaying a preview feed of many cameras in a grid and configuring exposure time, aperture, and ISO sensitivity jointly for all cameras. New settings are easy to add if needed.

Mac OS X issues The programs described in the following were written and tested in GNU/Linux systems, using libraries that would work also under Mac OS X. The gphoto2 library does not yet have official support for the Windows operating system. In a single Mac OS X laptop tested, parallel use of gphoto2 (the supplied application with the library) for multiple cameras showed concurrency issues, as the programs occasionally crashed due to a segmentation fault, suggesting problems with differently implemented USB libraries for the operating system. The programs were also much slower than compared to the Linux machines used, which may be an issue with the laptop's USB hardware.

5.6.2 Previewing

Although each camera can be positioned by individually looking through the viewfinders, a simple preview live feed is desired to properly set up a new configuration to see the big picture. A preview matrix of all cameras makes it easy to identify the cameras and to see if they all have been pointed to correct directions, and to verify that their settings are the same by judging from the image. The program is shown in Figure 39 and is found in the git repository under the name *gphotogrid*. It is written

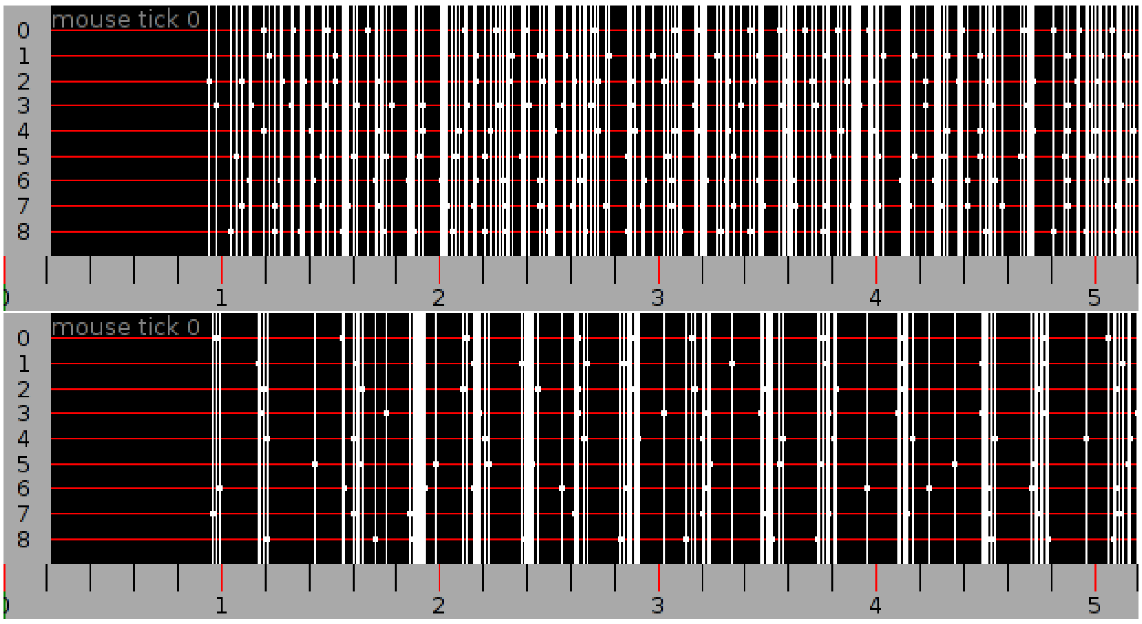


Figure 40: Timelines displaying the points in time (x axis, in seconds) when frames have been captured from different cameras (y axis, by camera number). The upper image shows all cameras running freely, and in the lower image, no new frame is captured from any camera until all have finished the latest capture.

in C++ and uses the libgphoto2 wrapper for camera control and the wxWidgets GUI library [140] for the user interface. The purpose is to view the picture frame of each camera in realtime and change the most important camera parameters concurrently.

Preview stream Libgphoto2 provides an interface for reading a camera preview frame at a rate of several frames per second, speed depending on USB hardware performance, the number of cameras used in parallel because of USB bus limits, and CPU processing power as the images are resized to the screen. A preview feed is connected to each camera to download the frames as fast as possible, and the user interface displays them as a grid, with configurable camera order. Because libgphoto2 does not offer asynchronous image or configuration transfers, a new thread of execution is set up for each camera, and the preview pictures are transferred to the main screen via concurrency-safe queues in the application itself. The code control flow is described in a simplified form in Listing 1.

Timeline A timeline widget was also developed to investigate the points in time when the preview frames have been grabbed, aiming to study synchronization accuracy using this preview method. Observing the times was found to be of no use, because the pipeline from camera to application seems to be so complex that the preview frame rate is completely sporadic, with no ability to control the frame order or timing. The application includes a checkbox for “synchronizing” the preview grabber threads so that no camera starts to read a new preview before all have read the last frame. This synchronization attempt did not have any positive effect; in fact, restricting each camera this way made the performance much worse, as can be seen in Figure 40. In the figure, a vertical bar is drawn for each frame captured, with a small circle on the line of each corresponding camera number.

Identification The cameras are identified and ordered by a name written to the configuration set under a property called *artist*, found at least in Canon DSLRs. The artist name can be set either via USB or from the camera's physical user interface, and it is supposedly originally meant for writing the camera owner's name in the image file's metadata, along with copyright information. For the camera rig, a single letter was used for each camera, ordered from *A* to *I*. A separate configuration property stored in the camera itself is most robust; USB port name can be used as an alternative to identify the cameras, but the port name changes when the cameras are plugged in another way when rebuilding the rig or using another computer.

The pop-up flashes should not be activated during use, because the camera was found to ignore the effects of most settings when the flash is up. Additionally, remote control disables the camera user interface. Taking pictures or changing settings is not possible while running the preview application.

It is recommended to spread the cameras as evenly to all USB root hubs as possible to help even out the transmission capacity, maximizing the frame rate. The same is recommended when downloading the actual images. The USB hub where a device is connected can be verified in Linux by looking at the output of the *lsusb* command-line tool.

5.6.3 Configuration

Remote control of all cameras is required to set all settings, such as shutter speed, jointly from the same computer instead of operating each camera's buttons manually. The preview matrix program allows to change exposure time, aperture size and ISO sensitivity, but more tools were written for automatic configuration. The tools are proof-of-concept type, written using the gphoto2 command-line tool as command-line scripts and are not particularly meant for final end-user. A graphical, button-based diagram specifying the next required step is suggested for future development.

- *config*: reset common parameters to defaults for all connected cameras, listed in the script itself
- *forall*: run gphoto2 with the user supplied arguments for all connected cameras serially
- *forallp*: as forall, but in parallel, saving time
- *post-download-by-camid*: download all previously shot files stored in the cameras to directories named by camera artist name
- *readconfig*: read and copy settings from one camera to all others currently connected

Listing 1: Pseudocode for *gphotogrid*. The lock task runs only if synchronization is enabled. The asynchronous UI settings are not shown.

```
per-camera preview feed {
        while running {
                download preview frame
                record current time
                push frame and time to queue
                if sync enabled {
                        notify lock about finish
                        wait for next round from lock
                }
        }
}

locking task {
        while running {
                wait for all frames
                notify loaders for next round
        }
}

screen update requested {
        for each camera {
                each queued frame {
                        store frame time to timeline
                        count number of frames
                }
                scale last frame to screen
        }
}
```

- *release-uilock*: unlock the physical user interface of connected cameras by fetching listings of their files; a workaround for the Canon glitch described below

The *forall* and *forallp* scripts are trivial shortcuts to autodetecting all cameras and passing commands to gphoto2 for each, and are useful for case-specific problems or setting individual configurations temporarily.

With *readconfig*, one camera is first configured manually, and the settings are cloned to others automatically with the script.

Either libgphoto2 or the EOS 700D camera has a bug that locks the camera's physical user interface when configuring the cameras, even only after starting and exiting the program: it appears that this is a safety feature that is left on after disconnecting the configuration connection, and probably gphoto2 does not release it properly. When setting or getting any configuration, the camera screen turns off and all buttons stop responding until the camera is rebooted, USB cable is unplugged, or if gphoto2 touches the camera's filesystem. Listing the files is thus enough and it has no side effects. The workaround script *release-uilock* was written as a reminder to perform the listing for all connected cameras.

5.6.4 Image acquisition

A command-line tool called *paraphotos* for downloading the images in realtime in parallel from each camera was developed in C++ using the same libgphoto2 library wrapper. The cameras send a notification via USB when a new photo is taken with the shutter button or the remote release. When this new file is found, it is downloaded and named by a running index and camera name. The user is notified when all cameras have finished for a single shoot; also, when the program is requested to quit, it first waits until the last pending download is finished. Each download task runs in a separate thread of execution to overcome the synchronous nature of libgphoto2, as well as the notifying task; a simplified code control flow is described in pseudocode in Listing 2.

Listing 2: Pseudocode for *paraphotos*. Ctrl-C (interrupt) signal is captured and it requests quit. Many of the possible debug messages not shown.

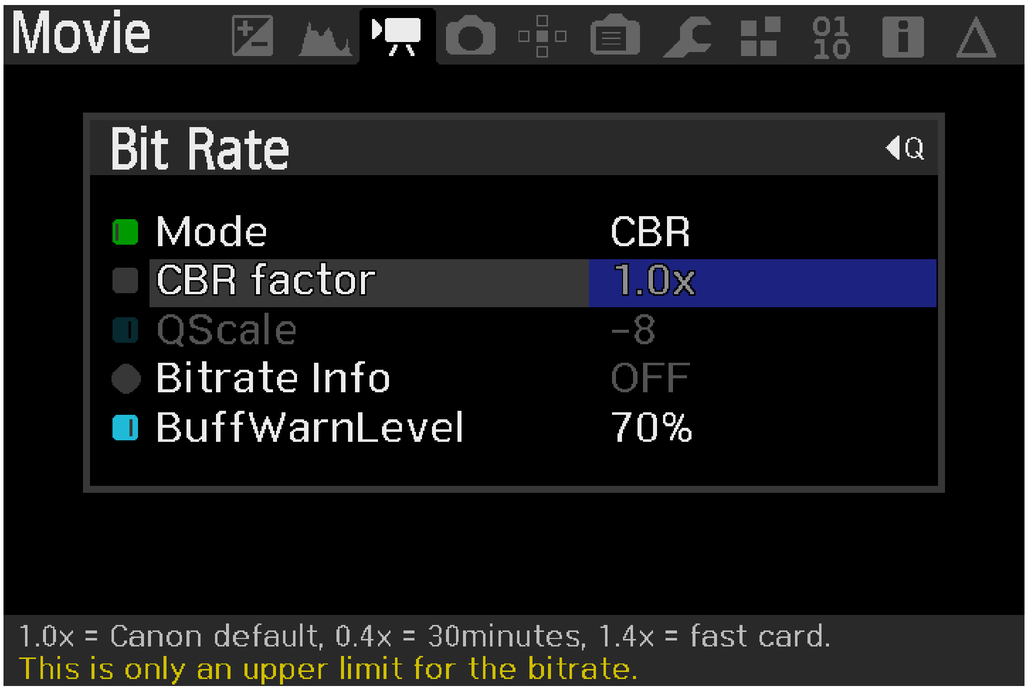
```
per-camera download {
        while quit not requested {
                poll for event
                if event is file addition {
                        download file
                        notify ui
                        wait for ui finish
                }
        }
}
ui notification {
        while quit not requested {
                wait for the number of cameras events
                notify user about a ready image group
                signal downloaders to continue
        }
}
```

Almost the same functionality could have have been achieved with the gphoto2 command-line utility; using the library directly with C++ resulted in cleaner code and probably faster operation.

Video acquisition is not possible in realtime; instead, the video files have to be saved on the memory cards first, and downloaded when the recording is complete. The *post-download-by-camid* clones the file systems from the cameras to local disk.

Video format should be set to 720p at 50 or 60 FPS or 1080p at 25 or 30 FPS. Both resolutions result in a video bitrate of approximately 5 MB/s with default settings. The video format is H.264 encoded in a Quicktime MOV file, with a stereo audio in raw lossless (PCM) format with 48 kHz sample rate and 16 bits per sample.

The bitrate can be extended from Magic Lantern's menus (shown in Figure 41), but bitrate changes are still experimental code and recording may stop if the camera or the memory card is not fast enough. Video keyframe rate can also be changed with a development version of Magic Lantern and probably will be stable in the future; in the video codec used, the frames can be *I-frames* (intra-frames), with



Mode CBR
CBR factor 1.0x
QScale -8
Bitrate Info OFF
BuffWarnLevel 70%

1.0x = Canon default, 0.4x = 30minutes, 1.4x = fast card.
This is only an upper limit for the bitrate.

Figure 41: The bitrate configuration dialog in Magic Lantern. CBR stands for constant bitrate; QScale is a quality scaling factor. Magic Lantern allows to modify the video quality at the expense of recording size and the risk of stopping the recording if the memory card cannot receive the data at the speed the camera offers.

fully encoded data, or *predicted frames*, encoding changes between frames, with less accurate information. Raw recording is also supported, but with the limited memory card speed, only a low resolution is achievable.

5.7 3D reconstruction software survey

When the imagery has been downloaded to a PC, the actual reconstruction can be started. 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. Some of them are presented in this section. In the scope of this work, new reconstruction algorithms or implementations are not implemented; thus, it is important to test the rig with readily available implementations.

Many, if not all, programs assume that the Exif data embedded in the photograph files describes the used focal length and sensor dimensions. Canon EOS 700D stores this information properly in the files.

5.7.1 Free libraries

There exist several generic computer vision and geometry and image processing libraries suitable for the purpose of 3D reconstruction. The most common ones are OpenCV [16], Point Cloud Library (PCL) [77] and Computational Geometry Algorithms Library (CGAL) [141]. These are written in the C and C++ programming languages and they have bindings to several scripting languages. For future work using the rig, these libraries present a well supported basis for developing own tools.

OpenCV contains a large set of utilities 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. PCL is popular in robotics involving three-dimensional computer vision; its algorithms apply well to point cloud data obtained using a LIDAR or similar range imaging device.

CGAL provides more strictly specified data structures and algorithms extending in combinatorics, convex hulls, polygons, arrangements, mesh processing, triangulations, voronoi diagrams and more, with an approach more suited in general computer graphics and geometry.

5.7.2 Free reconstruction programs

While libraries can be used to write new tools, many programs are readily available that implement 3D reconstruction based on methods presented in Section 3, and are distributed in source code form and/or readily usable executables. Some of the most known state-of-the-art implementations are listed below.

Camera Calibration Toolbox for Matlab, [48] also included in OpenCV, is a more or less standard tool for single-camera and stereo camera calibration. The toolbox computes undistortion maps and intrinsic and extrinsic parameters with checkerboard images using non-linear optimization and planar homographies.

Bundler [62] is a bundle adjustment and structure-from-motion system for computing camera poses and sparse point clouds, targeted for large-scale unordered photo collections, originally used for the Photo Tourism project.

SiftGPU [52] is a fast SIFT implementation for graphics processing units (GPUs). It is built on Sift++ [142], a separately developed open source SIFT implementation.

PBA [143] Multicore (parallel) bundle adjustment, is a fast bundle adjustment library built with the modern parallel nature of multi-processor computers in mind.

PMVS [66, 69] Patch-based, multi-view stereopsis, starts from a set of matching keypoints (features) and a pre-acquired camera calibration, and expands the keypoints to a dense patch set iteratively, resulting in a high number of oriented points. PMVS is usually combined with CMVS, a clustering tool that decomposes the input set to smaller clusters that are more convenient to process, when the amount of different views is large.

VisualSFM [74] is a common and free but closed-source full-scale pipeline that implements a structure from motion algorithm and simplifies the workflow of several external programs, visualizing the results in real time. The needed external tools are feature matching, bundle adjustment and dense reconstruction. VisualSFM is probably the most common tool for practical use in the open source and research community. It integrates into a few clicks the pipeline from images to 3D point cloud, using SiftGPU for features, its own radial undistortion model and SfM for camera estimation, PBA for bundle adjustment, and finally PMVS/CMVS for dense matching.

Meshlab [144] is a generic interactive editor of point clouds and meshes, implementing many scientific papers. 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 with the poisson surface reconstruction method, and finally projecting the textures to the generated mesh when given the camera parameters in relation to the point cloud pose. Meshlab is often used in conjunction with VisualSFM.

PoissonRecon [76] (Screened) poisson surface reconstruction is a stand-alone program as an alternative for surface fitting.

CMPMVS [72] 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. The software is Windows-only and provided for research purposes.

Python Photogrammetry Toolbox [145] is popular in archaeological fields. It combines Bundler and PMVS/CMVS into a full-scale pipeline.

5.7.3 Commercial solutions

There is a large selection of commercial software available, often accompanied with a 3D scanning device or a complete service. Some companies offer completely specialized rigs, some sell a scanning device and a general-purpose software alongside it which can be used as standalone and some only sell software, leaving the hardware implementation to the user. The short listings here review some of the commercial products that use 3D scanning as a key idea.

The following list includes software-only reconstruction solutions available for purchase, that work as-is for any general input.

Autodesk 123D Catch [146] is a free and popular web-based application for automatic reconstruction; photos are uploaded to a cloud service, and a textured mesh is generated automatically. Not much is left to the user, and the targeted user base appears to be those with no particular knowledge on the technology itself.

Acute3D Smart3DCapture [147] is targeted for large-scale photogrammetry and cultural heritage digitization. The software is provided as a standalone program and a library, and it is used internally in Autodesk 123D Catch.

Agisoft PhotoScan [148] has two editions, for 3D designers and professional geographic content manipulation; among the geographic features, it is very suitable for multi-view reconstruction in content generation, and has an active forum for its users [108]. It can be scripted with the Python programming language. Infinite Realities [105] uses PhotoScan commercially for reconstructing whole human bodies.

Faceshift [149] provides a realtime markerless face capture in a feature space that describes virtual muscle and bone movements, using a single camera and pretrained poses. The software runs currently with a depth sensor such as Microsoft Kinect.

3DF Zephyr Pro [150] by 3DFlow is an automatic reconstruction software that takes photographs as input. A free SfM software, **3DF Samantha**, is provided too. The company is a spin-off of a university thesis project [63, 64].

Pix4dMapper [151] converts aerial images to geometric surface models for mapping related applications.

Pixel Farm [152] provides tools for reconstruction, CG integration and match moving (i.e., augmenting live footage with computer-generated imagery).

3D-equalizer [153] is a multi-platform 3D tracking solution for the visual effects industry, including Python scripting, customer support, and more.

In contrast to the above, the following list includes complete systems that are offered as a service or as an addition to a program only.

MotionScan is the technology used for the video game L.A. Noire [154], developed by CaptiveMotion. It uses tens of cameras to recover detailed structure and is targeted for video game content generation.

Pendulum Studios [155] provides a capture system called Alter Ego that integrates with game engines, used in several video games.

FARO [156] manufactures hardware and provides scanning software for 3D documentation and surveying, and provides a free application for Kinect-based scanning. Many commercial applications are listed, ranging from product development to crime scene analysis.

Dimensional Imaging ltd. [157] provides DI3D and DI4D systems and software for static and dynamic surface capture, respectively, used for video games, research and medical applications.

Vicon [158] is a mature motion capture company, producing both both hardware and software for motion and surface capture and listing applications from life sciences to engineering and entertainment.

3dMD [59] offers healthcare-focused systems, software and research papers.

V-STARS [159] is a product line of Geodetic Systems, which provides systems for industrial photogrammetry.

5.7.4 Suggested use

From the lists above, two main methods were chosen based on popularity and availability, namely PhotoScan and VisualSFM combined with Meshlab. The former is a commercial solution, and the latter is composed of free software parts developed for research. The rig will first be tested for use with both; for algorithm development, custom implementations may freely be used to replace them, as the rig itself does not require any specific software. Agisoft PhotoScan is a self-contained pipeline, and VisualSFM does only part of the SFM reconstruction, leaving work to external programs that it runs automatically, providing flexibility in selecting the different algorithms. These two were used in evaluating the system in the following part (Section 6).

PhotoScan PhotoScan [148] is a popular commercial program for scanning subjects from several pictures for generation of 3D models and measuring spatial data. Agisoft advertises PhotoScan "to be used in GIS applications, cultural heritage documentation, and visual effects production as well as for indirect measurements of objects of various scales". The program is targeted for users without necessarily full knowledge of the technology, and is popular and successful for modeling human subjects according to the forum [108].

PhotoScan uses a typical multi-view reconstruction pipeline, taking also into account practical issues such as light variations in different images. The steps consist of photo matching, camera parameter estimation, dense reconstruction, mesh generation and texture mapping. The user interface in PhotoScan may also be used for limited editing of the resulting model, such as removing outlier data by hand. No detailed description is available on the implemented algorithms [160].

VisualSFM VisualSFM [74] performs the same structure from motion estimation and dense point cloud generation by combining several research projects into a single graphical user interface. The work pipeline in VisualSFM is composed in parts that are done mostly by external programs; feature extraction and matching is done first using SiftGPU [52], followed by SfM and bundle adjustment using PBA [143], and finally dense reconstruction is performed by PMVS/CMVS [66, 69]. Final Poisson surface reconstruction is done by opening the resulting dense point cloud in Meshlab and using its surface reconstruction plugin [144].

PMVS works by expanding a set of initially found patches iteratively in the reconstruction, resulting in a dense point cloud with oriented and colored points.

The point cloud is filtered by certain visibility constraints, and the resulting points have few outliers [69].

PMVS divides the source images iteratively to smaller sets, and begins at a halved size by default with one fourth of the pixel count of the original. Forcing the original size was found to provide roughly twice the number of points with four times longer processing time, but with no observed increase in accuracy, as if a spatially noisy object were interpolated with more points. The documentation of this parameter suggests that a halved size is used, because a large part of the pixels in the color image is interpolated in the original image because of the color filter array in the image sensor (Section 2.2.4).

No software installation instructions are provided in this thesis; refer to documentation of the presented programs for installation and usage. Testing was done on the Linux operating system, and the VisualSFM pipeline was built from source with no modifications done.

6 Experiments and results

The feasibility of the rig was evaluated based on custom techniques on syncronization and on several scans of different subjects. The purpose of the experiments was to see how suitable the rig is for the task. Synchronization was considered important from the beginning, and it was unknown how well the system would perform. Measurements were done in static and video modes. Photographing test subjects and running reconstruction using popular programs was considered a good way to evaluate the system as a whole, as the system was planned to be used for such methods. Subjects with different surface materials were chosen to test if the material types would perform as expected.

6.1 Synchronization

Synchronizing the image sources temporally is important for reconstruction algorithms to work correctly. A synchronized system captures images from all cameras with a reasonably small time difference between them, so that subject movement does not introduce unnecessary errors. Nonrigid movement would introduce inconsistent noise to the reconstructed depth, as disparity would be computed from mutually inconsistent matches.

6.1.1 Shutter delay and sync

Shutter delay consistency among cameras describes how well the cameras respond to the shutter release signals, and consequently, what exposure times can be used and how fast moving subjects would work. A repeated test was conducted using the microcontroller board to accurately measure the shutter delay and its variations on several cameras.

An input pin of the Nucleo-F401RE MCU was connected to the hot shoe flash present in the camera body. Assuming a negligible delay between the flash trigger signal and the actual photo exposure, the time between shutter signal and signal from the flash connector is the shutter delay. The camera was set up to all manual settings and manual focus, and the focus signal was given before each measurement. Measurements matched consistently a 75 millisecond delay, repeated shots varying less than 0.5 %. This level of repeatability suggests that moderately moving subjects can be captured properly. If a synchronized light was connected to the remote trigger instead of the cameras' flash contacts, this delay would be needed to be taken into account. The used test program for the Nucleo board can be found in the microcontroller programs directory in the github repository by the name *measure.cpp*. The different signals to all ten outputs in the remote trigger were measured with an oscilloscope to happen within 1.6 microseconds.

A special flash synchronization method was considered using the hot shoe signals of all cameras. When the camera orders the flash to fire, its shutter is expected to be fully open. The same controller that drives the shutter signals would then monitor the hot shoe signals, and trigger the flashes when all the shutters have opened.

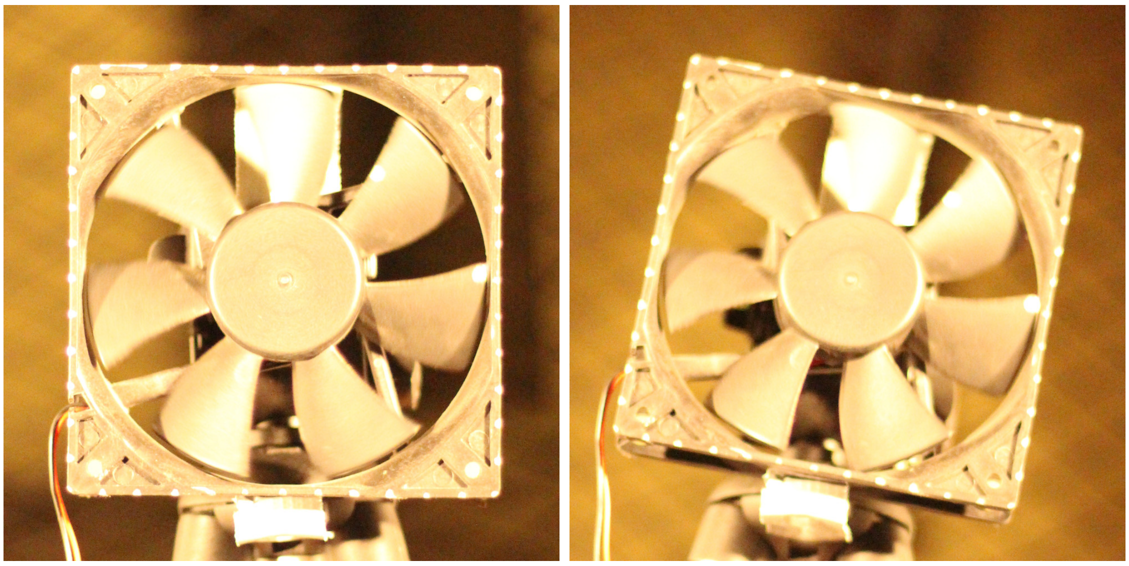


Figure 42: Frames from two cameras in a synchronization test with a PC fan. A marking on the end of a fan blade can be seen in the right side. The position of the marking does not differ distinctively in the pictures.

The method was not deployed because the cameras are synced well enough; with shutter speeds of as high as 1/100 s, no dark frames were encountered during testing. Maximum shutter speed using a flash for the Canon EOS 700D is 1/200 s.

6.1.2 Still capture sync using a moving object

In addition to the flash shoe measurement, synchronization between all nine cameras in a real setup was analyzed using a moving object.

A standard 120 mm PC fan was rotated in the view of all cameras, with markings drawn around the fan and in the moving blades with a bright-colored pen. The cameras were set to maximum aperture (f/1.8) and shortest possible shutter speed (1/4000 s) to minimize motion blur. This particular fan's speed had to be reduced by using 4.5 volts instead of the nominal 12 volts to clearly distinguish the blades without motion blur.

Without any actual measurements done but only with careful manual observation, no significant sync mismatches were found by comparing the position of the rotating measurements in all nine different images. Two such images are depicted in Figure 42.

6.1.3 Video capture

The EOS 700D does not support an external clock signal for timing the frames in video recording, but relies on an accurate internal clock to sync the frames within a single video file. It can be assumed that the clock speed has little variation, and thus, little drift or jitter occur in the videos during timespans of a few minutes. Even if errors in a single clock were measurable, offsets would still be the dominant source of errors. The processing time of each camera from the shutter button to start of first frame varies because of unknown reasons; even though a Canon wireless remote control was used for starting several cameras at once, the offset was found to be sometimes almost as much as the duration of two frames.

Visual and audio-based tests were carried out to find variations in the recording offset. Three cameras were triggered several times with a Canon RC-6 wireless remote control. The remote sends a short infrared pulse to a sensor in the front of a camera when a button is pressed. The cameras were set in video mode and configured to use the remote release. When all cameras had started recording, a flash unit (Canon Speedlite 420EX) was aimed at the lenses and fired manually several times.

A direct light pulse of a flash unit shows up as an over-exposed, completely white video frame, if the whole frame was exposing the image during the light pulse. Sometimes only part of the frame would be exposed to the flash light as a rolling shutter effect. Because the CMOS sensor is cleared and read line by line, the time of the flash can be deduced from the first bright line in the image. The rows are zeroed from light in succession in the same way as a mechanical focal plane shutter moves and exposed until read. Thus, the flash light's starting time in the video's internal clock is the frame's start time plus the duration that it takes for the rows before the overexposed one to show up:

$$\begin{aligned} t_{Cl} &= t_{Ci} + t_{Cf} \\ &= Fi_C + Ff_C/R \\ &= (i_C + f_C/rc)F \end{aligned} \tag{24}$$

where C is a camera identifier, t_{Cl} is camera-relative time for the light pulse, t_{Ci} is the starting time for the frame where the light shows up, and t_{Cf} is the time inside the particular frame. The frame index i marks the number of the frame in the video file, starts from 0, and there are $F = 1/\text{fps}$ seconds between the frames. f_C is the first row exposed to the flash, $R = r/c$ is the time for a single row to expose, r is the vertical frame resolution and c is a correction coefficient describing the ratio of exposure time and frame duration due to rolling shutter (0.5 in this case, referring to a characteristic to the camera, as it exposes the first and last rows in half the time between frames). Offset between two cameras A and B is then $t_{Al} - t_{Bl}$. The timings are depicted in Figure 43 as a timeline with matching symbols.

For perfectly synchronized videos, the frames would be identical, and time difference calculated with Equation 24 for each frame would be zero as i_C and f_C would be constants for all C.

The method for measuring the synchronization error at a sub-frame precision can be used to calibrate the differences, which is useful in artificial sync with optical flow or other methods, where frame timing is used. A similar method using the recorded audio (as in Figure 45) is more used in video production [161, 162], where similarities in the sound of the audio tracks recorded by the different cameras would be used for shifting the videos so that their clocks is in sync. For a small number of cameras, the audio tracks are simple to align manually in an audio editing software, or automatically with cross-correlation to find the time lag.

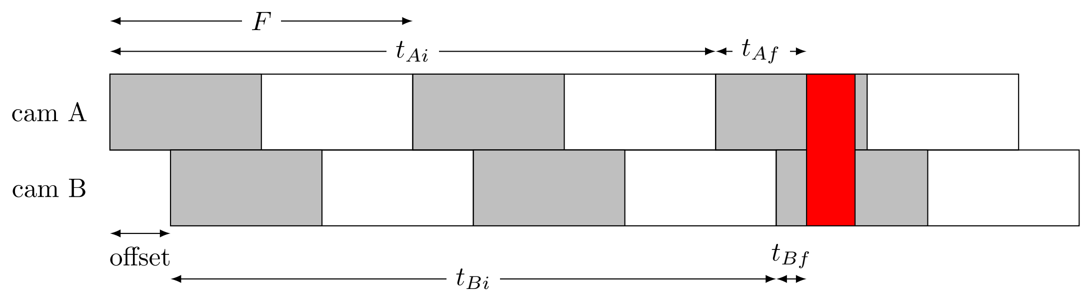


Figure 43: Illustration on timing calibration of two cameras using a flash light. The darkened areas represent the exposure times and the red band represents the light pulse. In this example, $c = 0.5$.

In a work by Bradley et al., a *moving-object synchronization* method [90] was suggested for post-process video synchronization using a detection of a rotated calibration grid. Bradley et al. noted that sub-frame offset effects were not noticeable during moderately fast motion. Optical flow was suggested for sub-frame synchronization if necessary.

Another method for triggering multiple video recordings is to use a Magic Lantern based wired remote trigger by pressing the half-shutter, or focus button when a record key has been configured in ML to start recording. This method did not show any improvements in overcoming the offset, compared to the wireless remote.

A typical measurement result is shown in Figure 44. Audio recorded by the cameras' internal microphones at the same event is shown in Figure 45, and the order of flash pulses in the three frames can be verified from the order of the sound emitted by the flash. Both methods show that the leftmost camera has started recording last, as it has been recording the least amount of time of all three before the flash is seen. Subsequent time lags compared to the first camera are approximately 21 and 31 milliseconds. A relatively long exposure time was used.

From the results tested, it was deduced that the Canon EOS 700D implements exposure time in a similar way as a physical focal plane shutter with two artificial curtains, where the first curtain clears each particular sensor row, and the second curtain reads out the accumulated charge. For a shorter exposure time, a shorter white overexposed line occurs in the video frames, but the scanning speed remains the same. For this reason, decreasing the exposure time only helps in motion blur, but not in rolling shutter artifacts.

6.2 Sample subjects

Test subjects were scanned using the system for validation and analysis. Reconstruction was performed using two software pipelines. Proposals for a typical scanning setup and minimum expected quality of instant results are shown in this section. The results answer questions on what kinds of surfaces can be reconstructed well; different types of materials were selected for the datasets. It was expected that areas with low variation in texture or high reflectivity would pose most problems.

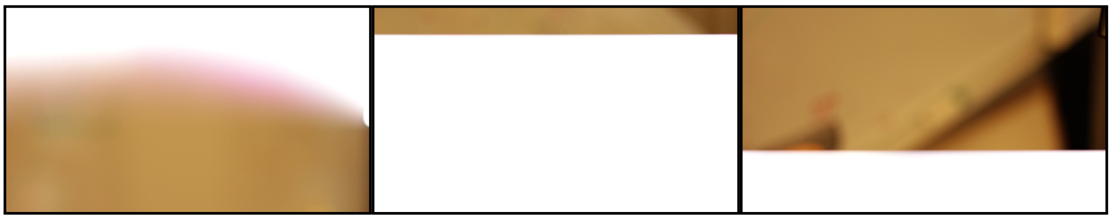


Figure 44: Video frames of three cameras started with the same remote controller, with the same frame index. The time for the flash unit to reach maximum brightness is almost instant. The start of the light pulse can be seen in the two last frames. In the first, the flash has been fired before the frame exposure started, and the frame shows the flash shutting down; the frame before it is without any overexposed light. It is impossible to deduce the exact time in this case.

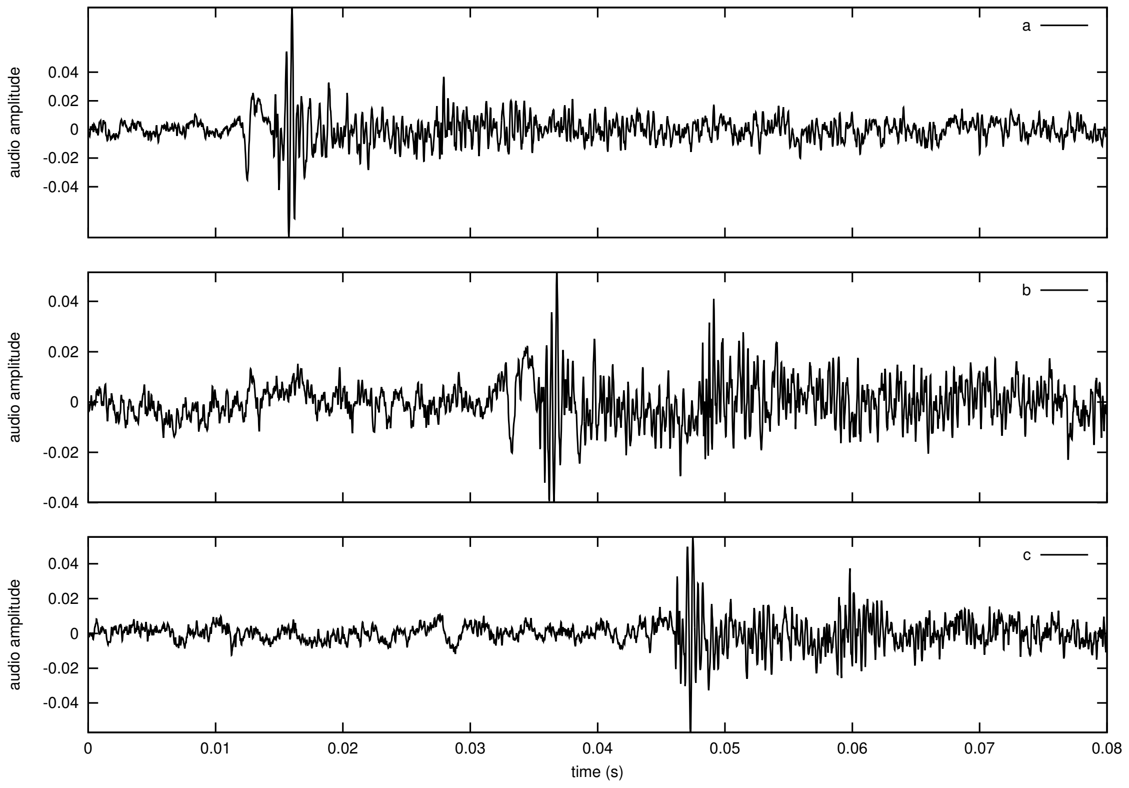


Figure 45: Trimmed audio recordings near the same moment as in Figure 44, extracted from the video files, in top-bottom order. First rows of the frames have been exposed at approx. $t = 0.03$ s. The flash outputs a distinct "pop" sound, comparable to the start of the bright part in Figure 44.

No particular tuning of any algoritmic configuration parameters were tested, which suggests that higher quality than that presented here can be achieved with more careful work. No makeup, light polarizers or any such improvements were tested; the subjects were photographed as-is in normal lighting.

The system is flexible to extend for complete body scanning with limited accuracy; tests for such subjects were left out due to lack of time and space in the hall where the system was used. For a complete body, the methods are identical but as the distance to the subject is greater, less detailed geometry is expected to result.

6.2.1 Photography setup

The nine cameras were set up in a dark hall as a 3×3 grid and two flash units were attached to two cameras, directed up and down to a white styrofoam board and a fabric diffuser to soften the light. The flash units were a Canon Speedlite 430EX and a Yongnuo YN565EX II, set to normal front curtain sync. The top-facing Canon flash has only automatic mode, which was sufficient; the Yongnuo was set to manual full power, facing downwards to a piece of fabric. The Canon EOS 700Ds have internal pop-up flash lights; they were not used, as they were found to trigger inconsistently, and they are relatively too bright and small so that shadows and highlights would pose problems.

A black background matte was used to isolate out unnecessary items in the background; the textureless out-of-focus fabric has no particular features that would be detected, and its color is not particularly important, although chroma-keying with a green cloth is commonly used to isolate the subject already in the source images. White styrofoam boards were left to the sides of the rig to help with lighting. The general setup is shown in Figure 46, and a closer depiction of the poses of the cameras and the flash unit bounced from downwards are shown in Figure 47.

The Bundler SfM system recommends to take photos at least every $\alpha = 15$ degrees around a subject [163]. Similar rough guidelines are usually suggested, such as a camera baseline of 10-20 % of the distance to the subject. With a distance s of one metre from lens to the subject, the Bundler guideline yields a baseline b of approximately 26 cm by trigonometry:

$$\begin{aligned} \sin \alpha/2 &= \frac{b/2}{s} \\ b &= 2s \sin \alpha/2 \end{aligned} \tag{25}$$

The key camera parameters used for photographing most samples are listed in Table 4. In a dark environment with flash lights, the exposure time is not particularly important for a static subject; the short flash pulse dominates all light in the scene. The baseline was chosen by the guidelines proposed and by the difficulty in using the cameras in close proximity to each other. Initially, a slightly longer baseline of approximately 50 cm was used; it was subsequently reduced to 30 cm.

Reconstruction software The programs suggested in the previous section were used for evaluation due to their popularity and availability: PhotoScan [148] and

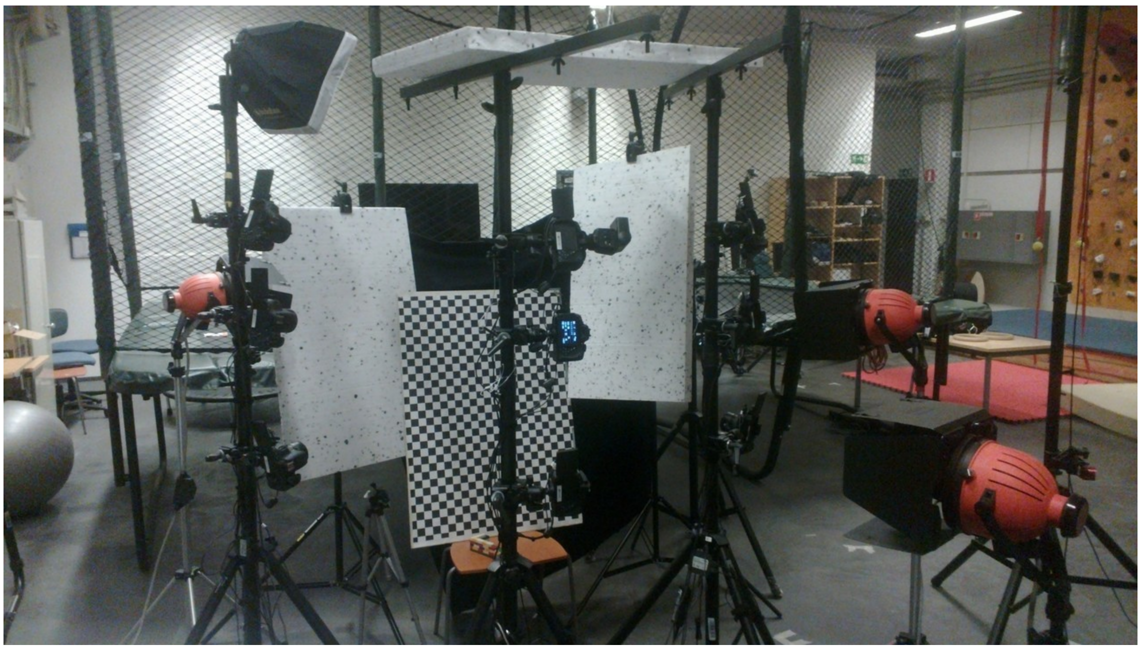


Figure 46: Complete setup with cameras mounted on the stands, surrounded by 800 watt spot lights (red) for video recording tests. For photos, the cameras were closer to each other as shown in Figure 47.

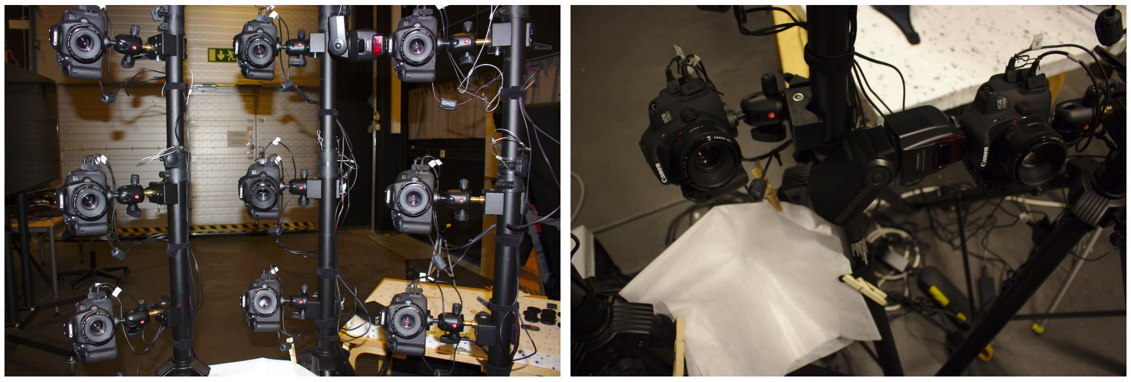


Figure 47: Left: Poses of the cameras from the subject's viewpoint. Right: Flash gun bounced from a white cloth is used to evenly illuminate the bottom side of the subject.

Camera baseline	30 +/- 5 cm
Aperture size	f/14
Exposure time	1/20 s
ISO sensitivity	200
Subject distance	90 +/- 10 cm
Lighting	Two flash heads bounced indirectly
Focusing	Automatic

Table 4: Key parameters in the sample photographs. Baseline refers to the distance to the nearest camera in the grid, horizontally and vertically.

VisualSFM [74]. Both were run on a typical modern Linux installation. A free demo version of PhotoScan was used, which does not support exporting the generated meshes; screenshots were taken in the user interface for illustration of results. VisualSFM and the related software were built from source.

All photographs were captured using the developed software presented in the previous Section 5. Identical settings were configured on all cameras; JPEG mode was used and no post-processing was done on the PC. Downloading the images in JPEG format from all cameras is done within seconds or less, depending on the USB hardware. The developed software downloads data naively from each camera concurrently, leaving the USB data scheduling to the operating system.

6.2.2 Static subjects

With the 18 million pixel count of the cameras, a human face is scanned with pore-level detail, and no markers are necessary. Markerless capture has been suggested to be important for human subjects, as both the geometry and the texture can be scanned simultaneously, and variations of the texture over time are also seen when capturing video [61].

The native resolution of each photograph obtained was 5124×3456 pixels; the subjects never filled the frame completely, but free space was left around them on purpose to make rotating and placing the subject easier. Moreover, human subjects would typically not stay completely still. The free space provides robustness that can be altered by bringing the subject closer to the cameras. Approximately 25-30 % of the pixel count was used for the subject. For the final resolution of the color texture that was combined from all views, 4096 x 4096 pixels was used in the images that follow in this section. The resulting accuracy of the color textures is easily determined by the physical size of the subject. Assuming approx. 30 cm coverage (by Equation 23) for the shorter image axis and a planar subject, image coverage would be $3456/300 \approx 11$ pixels per millimeter, or ≈ 0.09 mm per pixel. For the subjects tested, the accuracy was of the same order of magnitude.

PhotoScan results Using the source data shown partly in Figure 48, a point cloud and a textured mesh were generated in PhotoScan, using a quality setting of "High". The workflow in PhotoScan is similar to that of VisualSFM and other tools

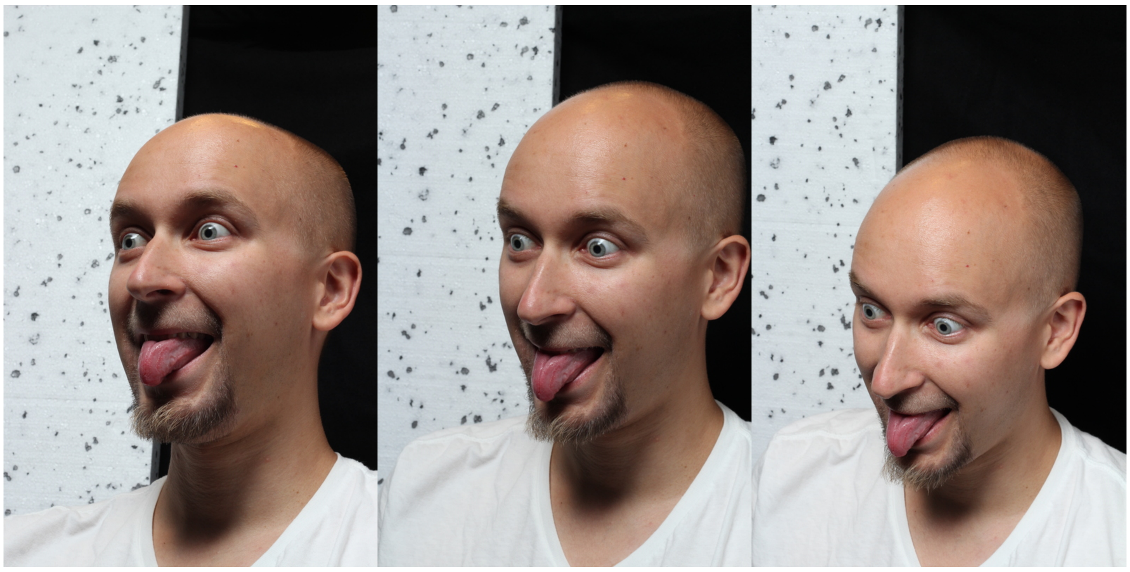


Figure 48: Three of the nine source pictures for a typical scan; for a more optimized scan, the subject should be closer to the rig to use the full frame. Note the shadows under the chin, the nose, and near the eyes; only the top flash was used for this dataset for demonstration.

based on separate programs: initially, the scene structure is computed, resulting in a sparse point set of the subject and camera parameters. A dense point set computation follows, and finally a textured triangle mesh is computed based on the dense points and the source images. Spatially uniform quality in the point cloud suggests that PhotoScan's algorithm uses all cameras in a combined way, instead of, e.g., computing disparity maps in pairs and naively combining the pairwise built point clouds, which would be recognizable by visible boundaries on the surface.

Figure 49 depicts the results from a dense point cloud and a textured mesh based on the point cloud. A typical face scan, with surrounding geometry removed, results in a high-density point cloud of 2-3 million points, which the software reduces into a textured mesh with exactly five times fewer triangles than the original points. The meshing process was found to consume temporarily 4-5 GB of RAM on a typical human subject.

PhotoScan was found to be relatively robust to baseline length, reconstructing the scene geometry well with a baseline almost twice as much as the 15 degree rule suggested in Bundler with no apparent increase in accuracy or matching performance when the baseline is halved. The tested reconstructions in PhotoScan have distinct spatial noise on all point sets and surfaces for unknown reasons, though. By the console output of the mesh generation process, the meshing algorithm is probably based on Poisson surface reconstruction or similar octree based algorithm.

The resulting point set is dense enough to include medium wrinkles in the reconstructed mesh geometry, but with a well noticeable level of noise, larger than the smallest pore-level details seen in the photographs. Medium-scale wrinkle detail tests can be seen in Figure 50; the point cloud and the mesh have enough detail to

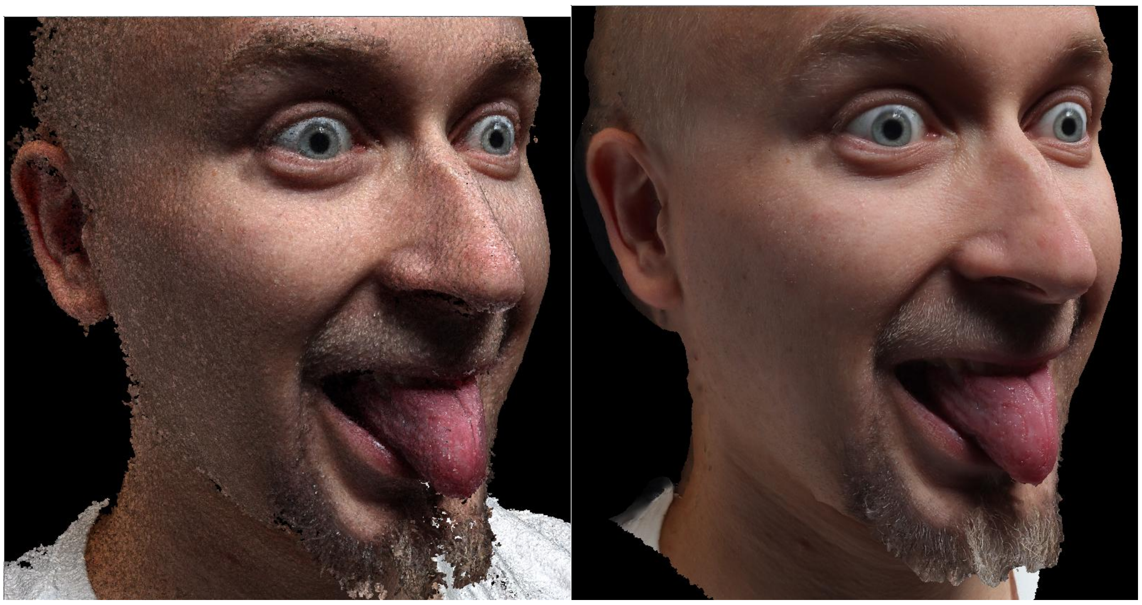


Figure 49: A dense point set (left) and a textured mesh in PhotoScan, matching the pictures in Figure 48. The underlying triangle mesh contains as much noise as the point set, which is hidden by the noiseless texture.

encode forehead wrinkles in the geometry, in addition to the texture color which is actually unwanted and a cause of less than optimal lighting.

In Figure 51, two of the texture mapping modes provided by PhotoScan are depicted. A "generic" mapping mode with no distortion in the pictures produces piecewise texture islands with original photo content. A "spherical" mode where the subject is assumed to be a sphere and the texture is made continuous but the original photos are warped shows as more intuitive and easier to modify manually if necessary.

VisualSFM and Meshlab results This alternative was found to be much faster than PhotoScan, while the resulting geometry also contains fewer, but more accurate points. Same datasets were tested as with PhotoScan with similar results. VisualSFM's feature matching seems not as robust as with PhotoScan, sometimes resulting in separate models because of a too long baseline and displaying very few feature matches between pictures. The VisualSFM manual suggests to decrease the camera baseline in this case or to take more pictures. With the shorter baseline of approximately 30 cm, the SfM process was consistently successful. This suggests that the SIFT feature descriptor may not be the best for this purpose, if it does not cope well with different viewpoints. The features could have been matched manually if desired.

The results of a face scan are depicted in Figure 53; the surface has visibly less noise than that of PhotoScan's, but without a ground truth geometry, precise analysis is not possible. Comparison between surfaces generated from PhotoScan

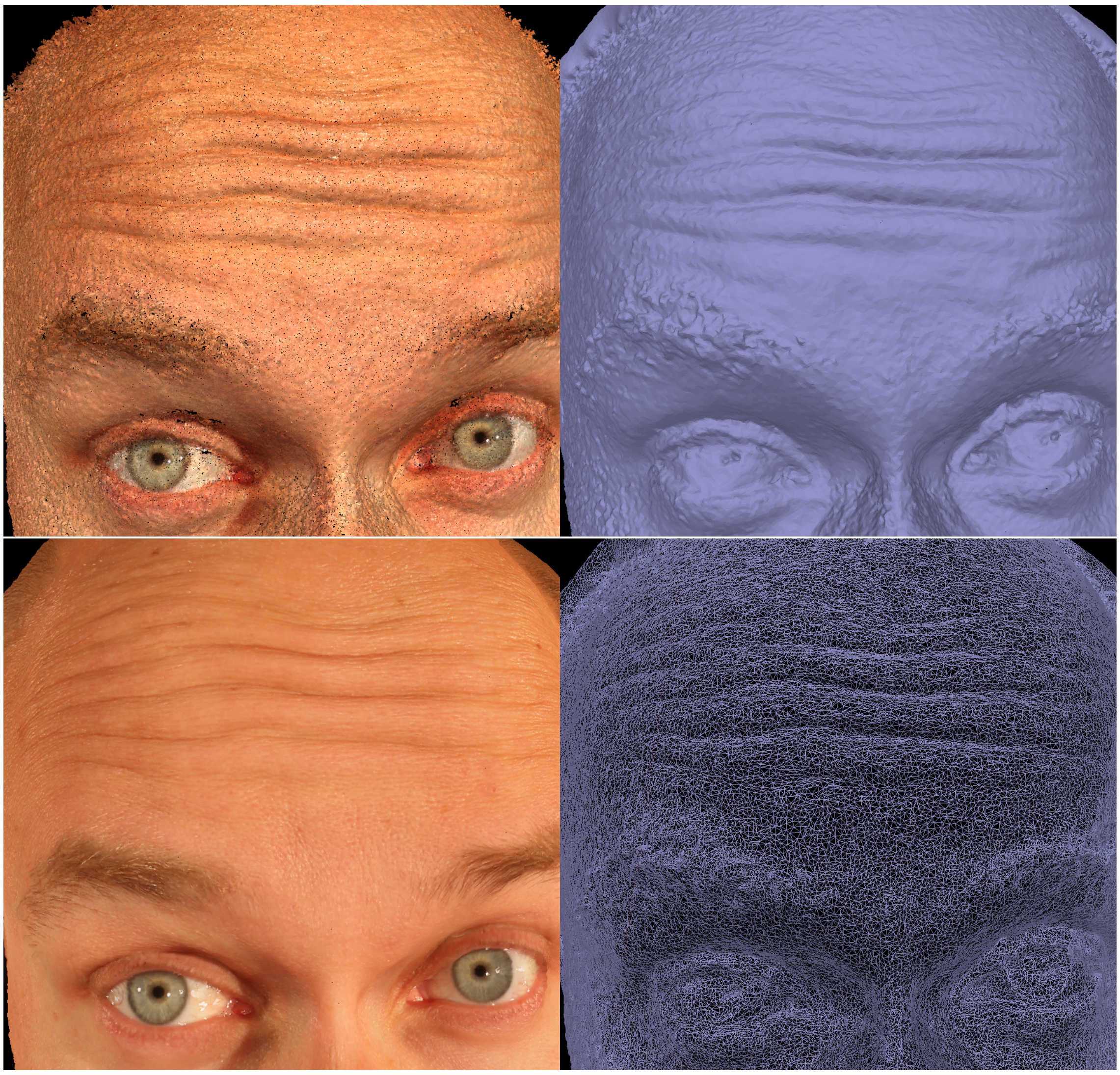


Figure 50: Forehead wrinkles in PhotoScan: dense point cloud, solid mesh, textured mesh and a wireframe mesh. The point cloud and wireframe pictures show the reconstructed detail level well. The darkened color due to shadowing in the wrinkles is directly encoded in the color texture. The eyes and eyebrows present difficulties to the algorithm, as expected.

and Meshlab is depicted in Figure 52.

Figure 54 shows a cropped region of one of the source photographs. Note that the skin pores and smallest wrinkles are visually well noticeable in the photograph, but the triangular model is missing smallest of them. High-level detail could be encoded in the color texture as it shows in the photographs, but since the detail is geometric, computer renderings using detailed light models in differently lit environments need geometric information of the surface roughness, not only a color texture. The texture type built in Meshlab is depicted in Figure 55.

Beeler et al. propose a mesoscopic augmentation from the captured color images

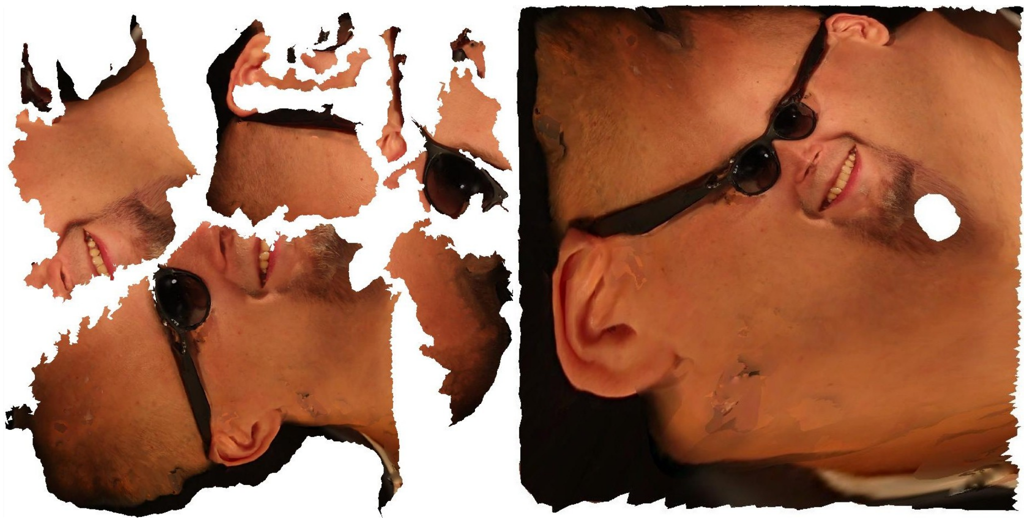


Figure 51: The “generic” (left) and “spherical” (right) texture mapping modes in PhotoScan. The missing spot under the chin is a result of poorly set up cameras that should have been set lower for more visibility.

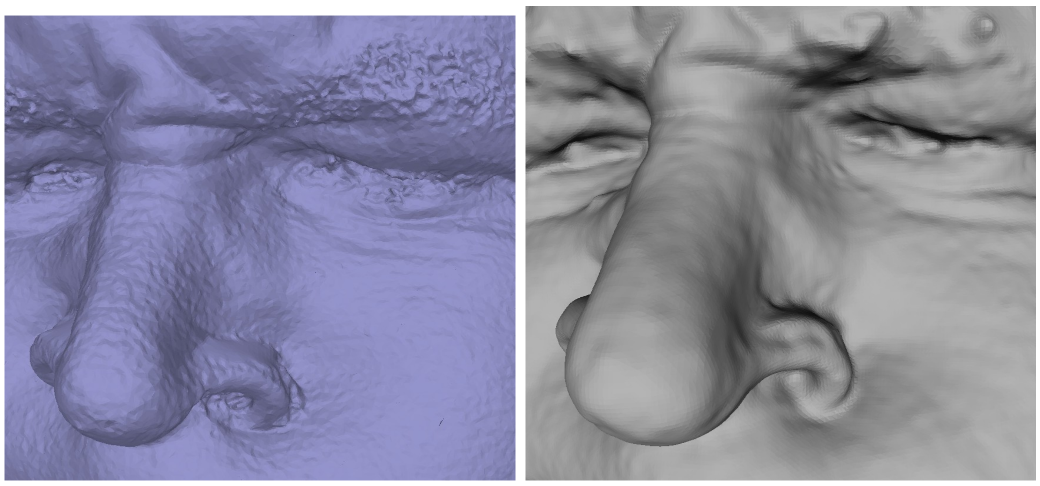


Figure 52: Surface smoothness comparison. The mesh from Meshlab (right) is visually more smooth than the mesh from PhotoScan, while the polygon count is approximately the same. Although not as visible, the same could be seen in the point clouds (not shown).

to increase the geometric detail level, and point out that they are not reconstructed to the surface only because of a too small camera resolution [60]. As high level of geometry as produced initially by the surface generation process may not be necessary if the color images are analyzed for producing textures describing the surface roughness level with texture mapping.

Other subjects that were also scanned and reconstructed using VisualSFM are summarized in Figure 56. Text in a page of a book provides enough dense points that the text is well readable, while the paper itself does not contain any texture and

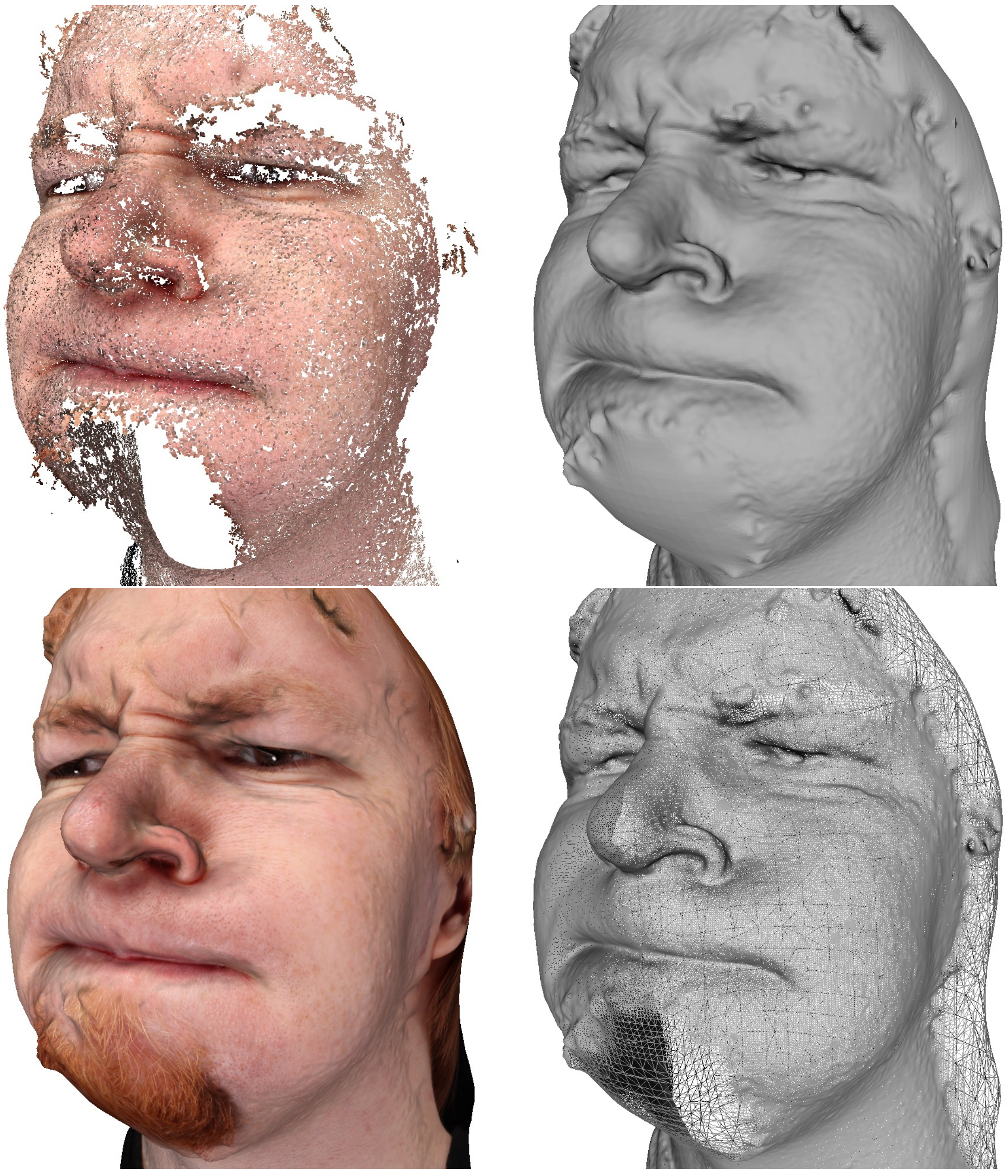


Figure 53: A dense point set, solid triangle mesh, textured mesh and a wireframe rendering in Meshlab. Point set reconstructed in VisualSFM.

is not reconstructed. The only problems in human scans are hair that is not well distinguishable (most features are similar) and also does not present any definitive surface, bright spots that result from a combination of bad lighting and sweating, and local shadowing in wrinkles getting in the color textures. A horse head mask made of diffuse rubber had problems only in textureless areas composed of exclusively white or black paint. Those areas either provided incorrect points or no points at all. Surprisingly, a poorly textured soft plushie provided well reconstructed geometry

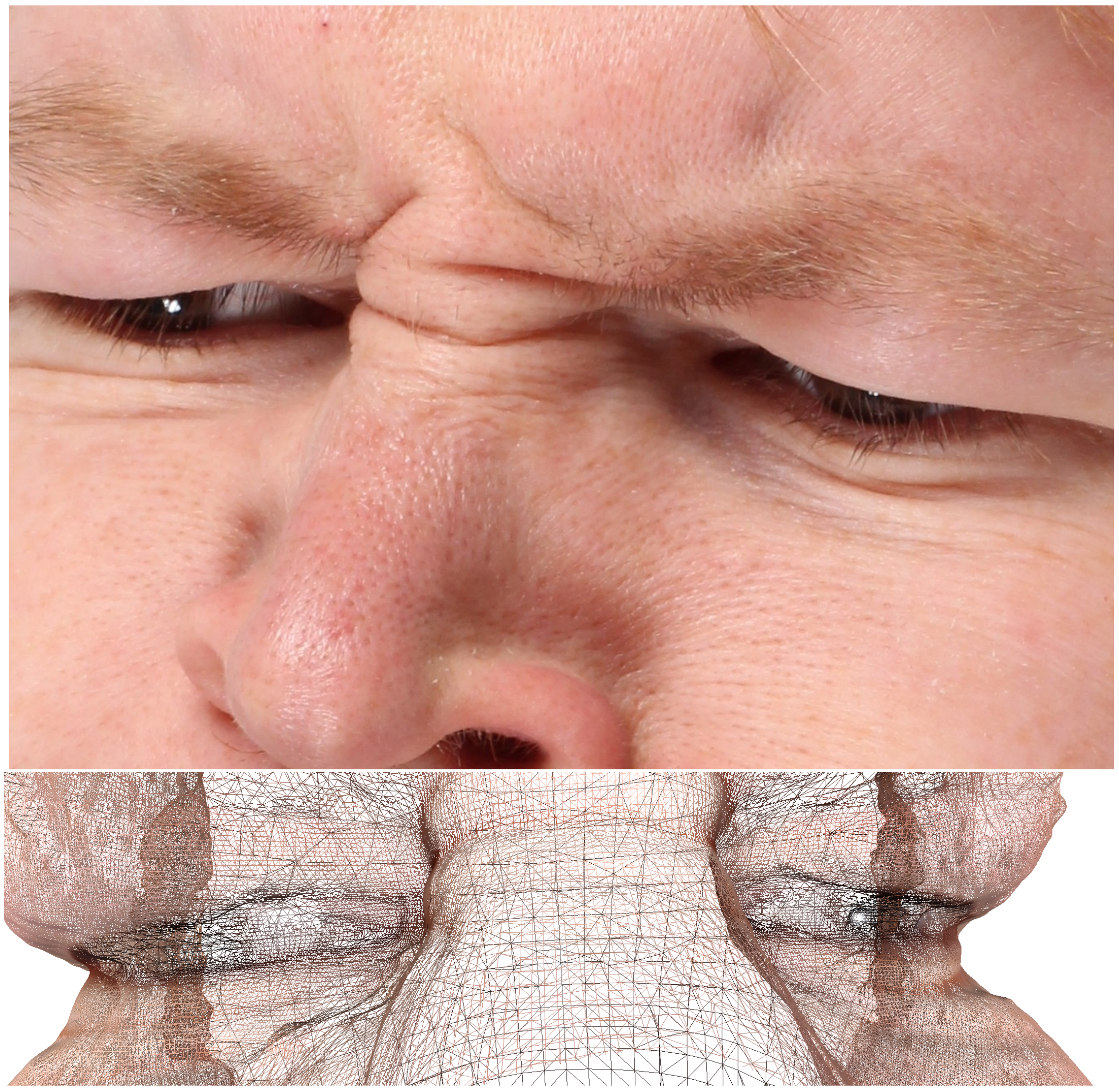


Figure 54: Close details from one camera, showing the mesoscopic level of detail, and a corresponding area as a transparent wireframe mesh, showing the amount of vertices. Note that the mesh does not yet employ the full geometric properties of the source photograph, i.e., the pores and wrinkles, that are beyond the geometric detail of the mesh.

even in less textured areas, where the fabric provides enough detail to distinguish any correspondences. A mildly reflective painted statue presented difficulties in some shiny areas, but otherwise, the whole geometry was found to be correct and dense even in visually weakly textured areas.

The statue model was scanned four times with approximately 90 degrees of rotation between the shots, and the resulting four point clouds were aligned into one in Meshlab to evaluate a surface reconstruction for a fully closed subject. Results can be seen in Figure 57. Some noise exists where it was not cleaned up, and closest details are missing partly because of noise, bright highlight artifacts and selected

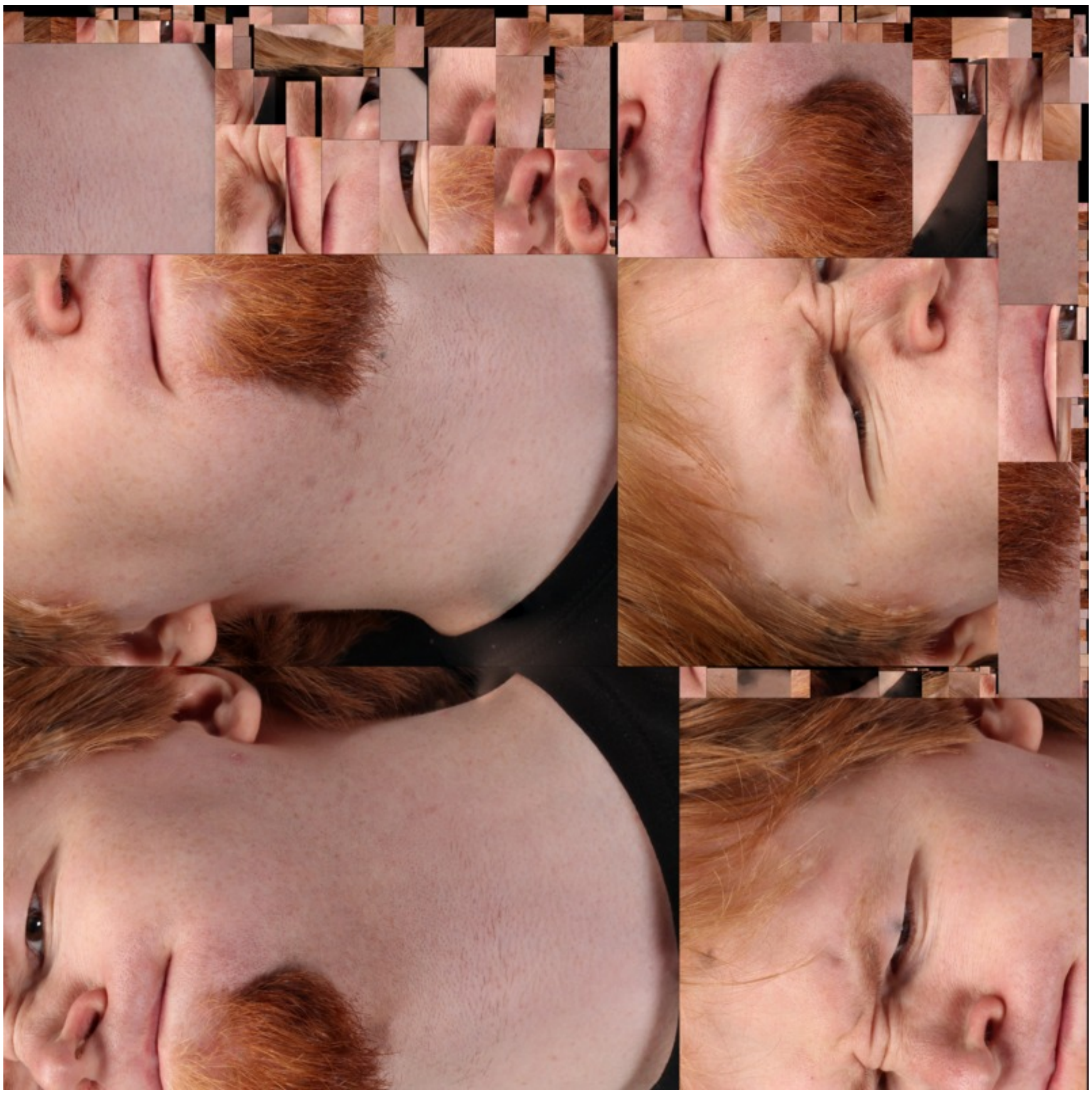


Figure 55: Meshlab produces a piecewise texture map similar to PhotoScan's generic mode.

level of detail in the surface reconstruction. Slight misalignment in the top can be seen in the point cloud version, and the holes in the bottom are errorneously filled in the meshing process. The mesh was built with Meshlab's Poisson surface reconstruction plugin with 12 as the tree depth and 7 as the solver divide depth.

A simple structure of Lego blocks was scanned with poor results, as was expected; uniform-colored blocks with no texture and no geometry variations present a difficulty to point matching. The corners between blocks of different color are the only well reconstructed areas. Figure 58 depicts the results from three vertical cameras only, as VisualSFM was unable to join the different features of all the cameras.

Typical durations of the reconstruction steps and the used hardware are listed

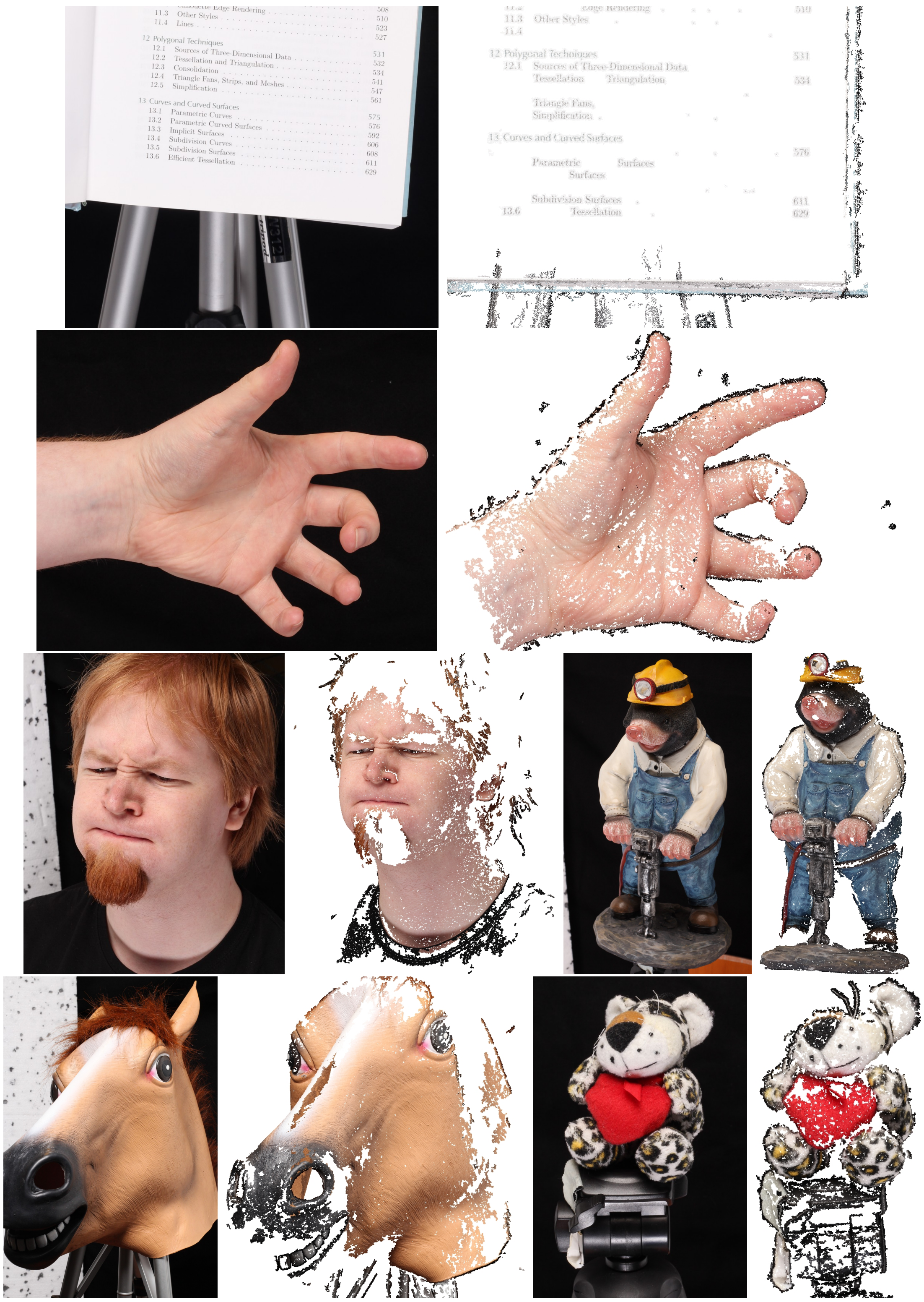


Figure 56: Comparison of original photos and unedited dense point reconstructions done in VisualSFM. Subjects listed: book, hand, head, statue, horsemask, plushie.

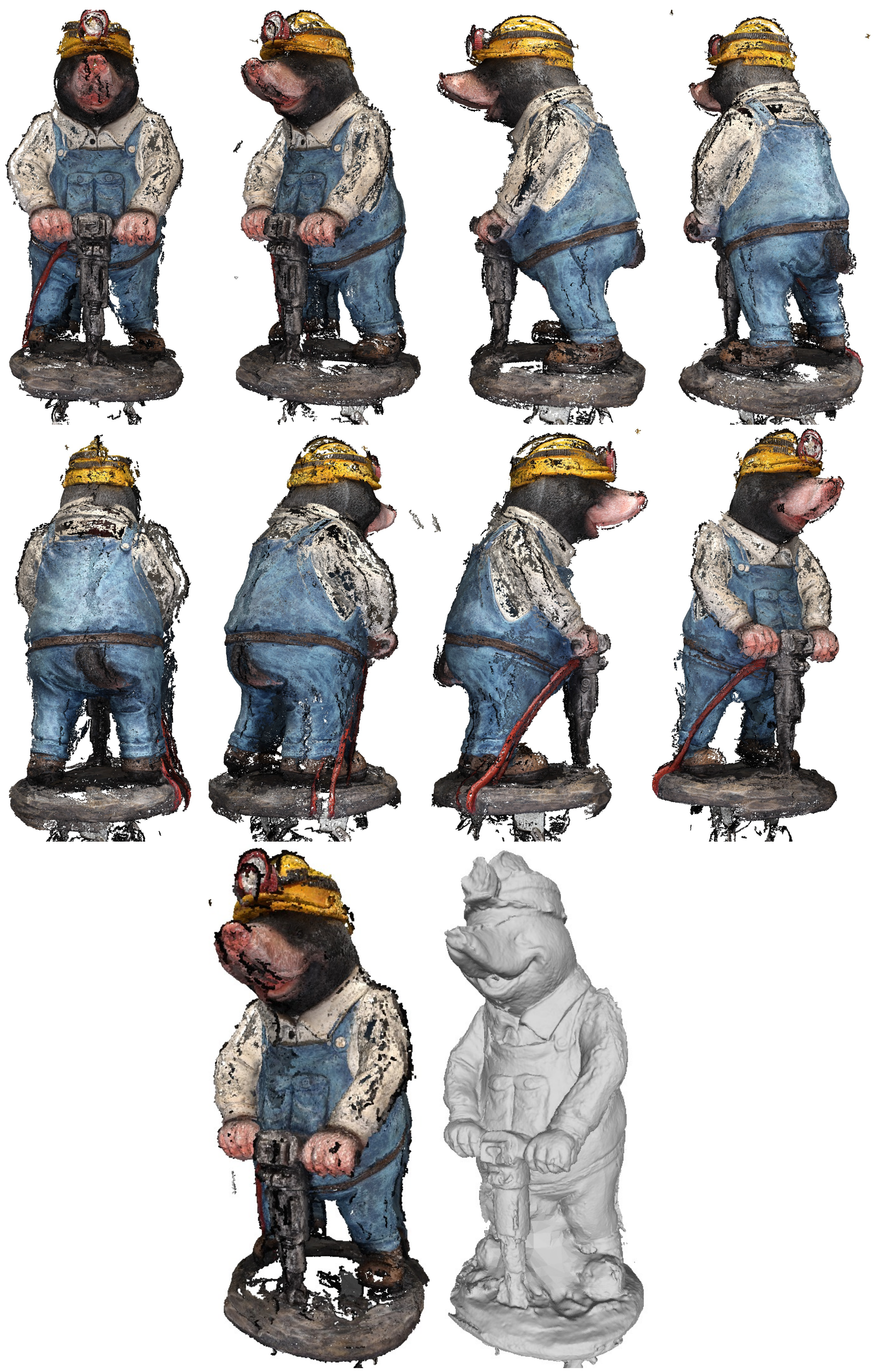


Figure 57: A full statue as a 360-degree point set, combined in Meshlab from four scans, and meshed. A slight misalignment between two of the clouds is visible. Misalignment errors could be addressed by more careful alignment and outlier removal, or a bigger rig with more cameras.

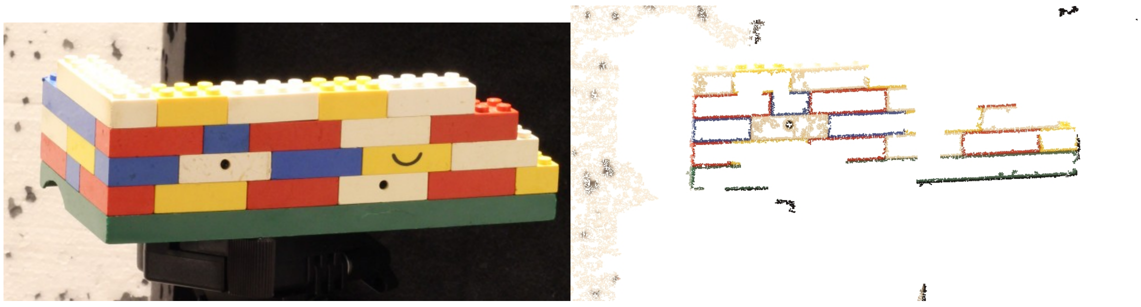


Figure 58: A lego block structure demonstrates that textureless surfaces reconstruct poorly, as was expected. The old and dirty block in the middle had enough texture to fill better than the others.

Step	PhotoScan	VisualSFM+Meshlab
Feature matching	75 s	17 s
SfM / Bundle adjustment	8 s	2 s
Undistortion		63 s
Dense reconstruction	40 min	20 min
Surface reconstruction	280 s	90 s
Texturing	115 s	15 s

Table 5: Typical durations of the major reconstruction steps in VisualSFM on an Intel i5-750 CPU and a NVIDIA GeForce GTS 250 GPU. The undistortion step is included in the dense reconstruction in PhotoScan.

in Table 5. Most of the time is taken by dense reconstruction. PhotoScan was configured to use CPU only; the dense reconstruction in PMVS is also performed on CPU. Different approaches remain to be tested; both have a similar working pipeline.

6.2.3 Video capture

Video capture was experimented with three 800 W Inaro Redhead Varibeam spot lights and no significant changes in camera setup, with key parameters listed in Table 6. As the cameras may not start recording at the same frame, a small utility was written to rename frames extracted from videos, given a manually set up list of temporal synchronization offsets. The tool is written in Python and can be found in the source code repository as *sync.py*.

The amount of light needed in video capture can easily be uncomfortable for the performing subject, in contrast to the momentary flash pulse in still photography. in this experiment, the lights were bright enough to be remarkably uncomfortable, and the subject used sunglasses for eye protection; less than three lights resulted in harsh shadows in parts of the face. Halogen spotlights were found to produce enough heat to make the subject sweat within minutes of continuous recording. A default "180 degree shutter" was used, i.e., an exposure time of half the time between

Camera baseline	30 +/- 5 cm
F number	f/14
Resolution	1920 x 1080
FPS	25
Exposure time	1/50 s
ISO sensitivity	800
Subject distance	90 +/- 10 cm
Lighting	total 2400 W halogen spot lights

Table 6: Key parameters in the video recording test.

frames. Motion blur was found to be an issue in some frames during quick motion; a shorter exposure time coupled with higher ISO sensitity would help. The video ISO sensitivity level of Canon EOS 700D ranges up to 6400, but the highest levels are usually avoided as they are extremely sensitive to noise.

The videos were shot with the larger 50 cm baseline which was later found to be intolerable for VisualSFM. Thus, the results were analyzed only in PhotoScan. In some frames, where the subject was moving too quickly, the resulting camera poses were incorrect. In similar challenging picture sets, VisualSFM typically did not reconstruct the parameters at all, due to lack of enough matching features. Placing a calibration object in the scene might have helped. With less motion, correct camera poses were found and results were comparable to the static reconstruction, with fewer points and polygons reconstructed because of the lower resolution (1920 x 1080). A typical result is shown in Figure 59; surface on the textureless and partly reflective sunglasses is incorrect, as was expected.

The camera uses the lossy h.264 video format and typically a bitrate of 5 MB/s. In comparison, the size taken by a single full-frame JPEG still photo is approximately 9 MB. The loss in resolution and compression quality results in poor description of surface details, as is depicted in Figure 60.

6.2.4 Final thoughts

Both PhotoScan and VisualSFM+Meshlab appeared to produce highly usable models. Both support similar output formats for the point cloud, mesh, and texture. Comparing the results, it can be concluded that PhotoScan appeared more robust to the shooting conditions and always gave results of some quality. Given carefully tuned conditions, the VisualSFM method gave more freedom in selecting different parameters and external programs, and might result in a better level of detail, given enough time for testing. The community visible in PhotoScan's forums [108] appears active, and the existing forum topics give a quick start to the use of the program and to reconstruction rigs.

While PhotoScan does not disclose the details of its algorithms, the algorithms used by VisualSFM and Meshlab are well specified. Meshlab's code is quite directly adapted from scientific papers, and being open source, it is easily modified. The fact that VisualSFM uses external applications for parts of the reconstruction process

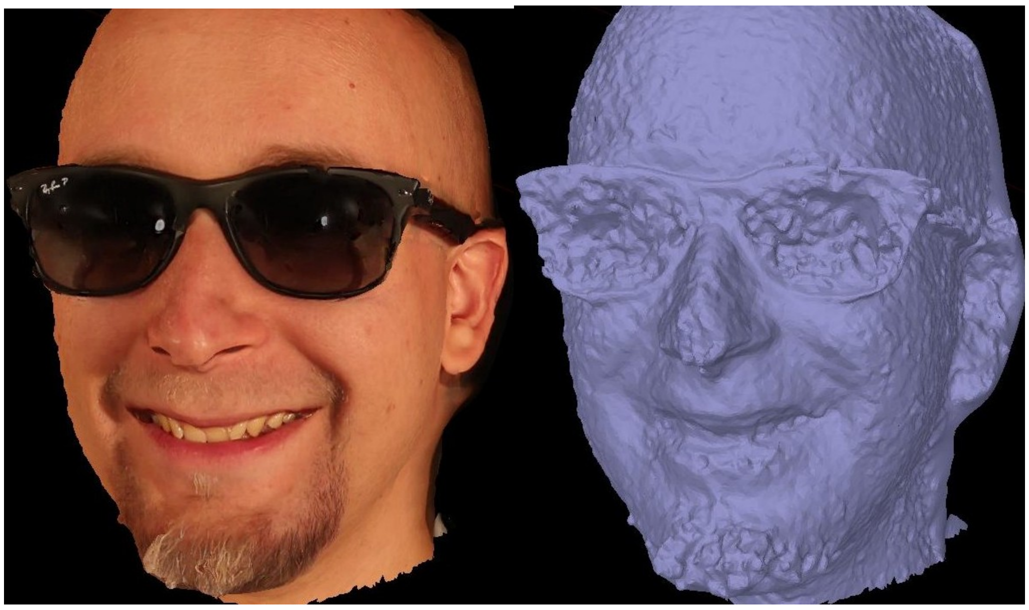


Figure 59: PhotoScan reconstruction from video frames, textured and solid. The solid rendering shows the relatively low vertex count and high amount of spatial noise, which are hidden in this particular textured demonstration view.

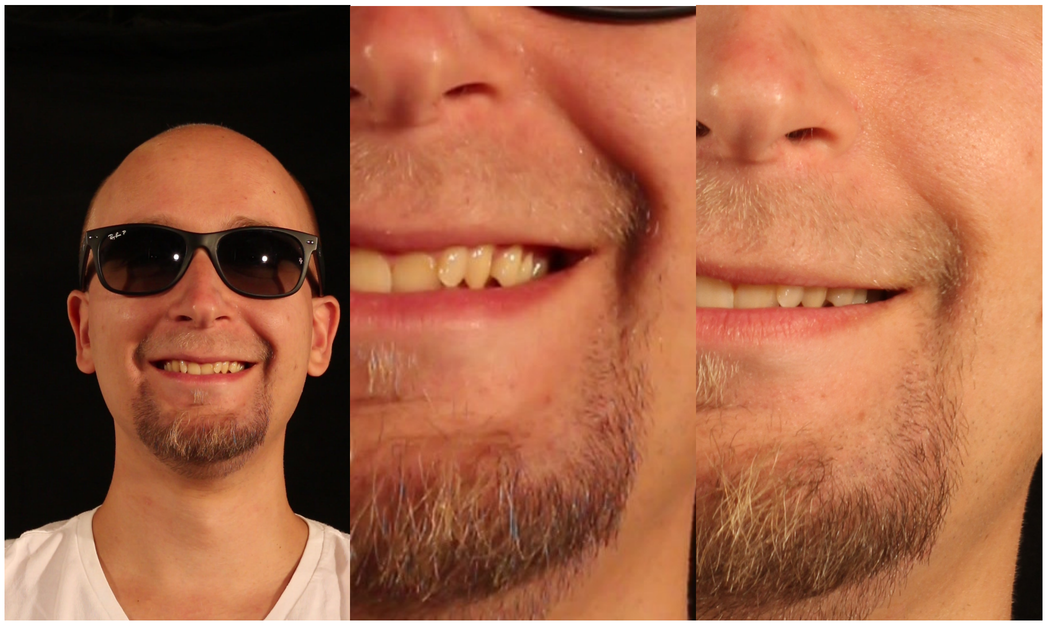


Figure 60: A frame of a single camera, a crop, and a crop from a still photo. Note the colored aliasing in the hair and the lack of surface detail in the skin.

and only performs camera alignment by itself, it can be readily used for trying other approaches on those steps performed externally by replacing the other applications.

7 Conclusions and future work

The purpose of this thesis was to build a rig for turning real life subjects into three-dimensional digital models, using a technique called *multi-view stereo reconstruction*. Modern reconstruction rigs are increasingly necessary for both content production and reconstruction research, as computing power increases and reconstruction methods advance. While the same methods are applicable for any number of digital cameras of any quality, the aim in this work was ease of use and high photographic quality. Based on the understanding built in the beginning of this thesis, requirements for the rig could be well specified and best settings could be chosen for the cameras and the photography environment. The end result is a rig of nine cameras tested to work for desired subjects using two different software pipelines, with potential to expand by adding more cameras.

First part of this thesis surveyed the image formation process in digital cameras and the 3D reconstruction principles that current algorithms build upon. Based on the camera technology study, the heart of the rig was selected to be an array of nine Canon EOS 700D digital single-lens reflex cameras using 50 mm lenses with an equivalent field of view of an 80 mm lens for 35 mm format, approx. 25 degrees. The cameras produce 18-megapixel images and the total footprint of the rig is less than 5m^2 for most setups including the subject. Software for jointly configuring the required manual settings in all cameras was surveyed and developed using the *gphoto2* camera control library. A custom electronic remote shutter trigger tool was built to synchronize the cameras for capturing still photographs in an accurate way even with a moving subject. For video capture, the cameras can still be remotely triggered, but as they lack accurate video synchronization, a maximum possible error of half a frame duration still exists. Artificially synchronizing consumer camera arrays is an interesting subject, as it is one of the biggest issues that separates consumer cameras from more expensive professional or industrial devices from the reconstruction point of view.

The selected cameras offer the necessary manual controls, can be remotely controlled with any standard personal computer and offer excellent image quality with a reasonable price. Hardware of the rig is fully flexible and can be modified for small and medium subjects that can be photographed indoors. Alternatives to the rig solution were found to be not configurable enough or overly expensive for the purpose.

Computer graphics has a clear connection to computer vision. Areas of computer graphics focus more on synthesizing content to build an image, while computer vision analyses images from real life. Interestingly, they both use similar principles, and vision can be used to create model data for rendered graphics. Similar ideas based on subject lighting can be employed in both fields; this thesis still barely touched the confluence of the two, and further study on state-of-the-art is suggested to better dive in the subject. The rig built as part of this thesis makes it possible to easily gather photographs and video from multiple viewpoints and suggests to study the state-of-the-art algorithms that analyze visual cues in the photographs, to reproduce the subject in a different lighting and to animate the subject in a new way.

Constructing a 3D scanning rig was a large topic on its own and future work remains in studying the data that can be produced with the machinery. Using the rig for content capture is practical once set up; new subjects can be scanned with the press of a few buttons. The field of 3D reconstruction needs content to evolve, even more so for content-specific improvements. Scanning the geometry and texture of real-life objects directly brings new practical opportunities on content usage and performance capture. The rig is to be used for digitizing real-life objects in higher resolution than what has been traditionally possible, and studying the process, helping to understand how different algorithms behave on different inputs. Also, identifying defects in current state-of-the-art is easier when high-resolution test data can be scanned for testing new hypotheses.

Suggestions for future work The example use cases and typical results presented a relatively careless use of the rig. Still, it can be deduced that the system is feasible for capturing many kinds of objects; even better quality would probably be achieved with tuning the physical setup and the algorithm parameters manually for each subject. Furthermore, the purposes of the rig – to study the reconstruction field, and to scan content for actual practical use – remain to be fulfilled. The user interface can be further improved from the current prototype phase to an actual product.

As seen in several photos, such as in the smile in Figure 59 and Figure 60, an even, diffuse lighting was not achieved using the current lighting and very dark shadows were introduced in certain self-occluded sections, making it difficult to scan the surface texture and resulting in low completeness of the scanned surface. It is suggested to experiment with devices used in professional photography, such as umbrellas and studio flash units inside soft boxes, possibly in combination with a large light tent where the system could be contained. Surrounding the subject and the rig with reflective white cover would help, either close to the subject requiring holes for the cameras on the cover, or completely using a large white tent and high-powered flash lights outside it. For example, the actors of the FIFA 14 video game were scanned inside a white tent with the Likeness Capture system [107, 164].

Arranging the cameras in pairs and aligning the resulting point clouds could provide more coverage with the limited number of cameras. Pairwise scanning has been done before successfully [60, 61, 164]; the multi-view SfM approach was used for testing because of its simplicity. Careful positioning within the camera frames is encouraged to take use of the resolution. However, requiring the subject to fill more of the frame restricts the space for movement, which may or may not be favorable. For a smaller section of a face, the cameras could be used as plain stereo pairs such that each camera would only see a part of the subject, thus using the camera resolution more effectively for more high density data, as presented by Bradley et al. [61]. More cameras would also provide more coverage.

A typical head scan results in a point cloud of 0.5-5 million points depending on the subject, lighting, and software used, eventually resulting in a triangular mesh with a number of triangles of the same order of magnitude. In rendering the models for actual use, fewer triangles should be used with geometry details not encoded as triangles but textures instead. The advanced rendering techniques employing the surface microgeometry as textures such as *bump maps* are beyond the scope of this thesis, but they present interesting future work in this field. The mesoscopic augmentation by Beeler et al. uses the same idea as high-detail rendering, where the tiny details that are occluded by surrounding geometry are rendered darker [60]. Ghosh et al. [32] photograph a same still subject in different illumination conditions by changing the lighting quickly, in order to scan for both diffuse and specular reflectance components of the surface. The current rig has possibilities for augmenting for such work.

Interestingly, the video performance capture based on anchor frames by Beeler et al. [84] is able to record and replay a temporally consistent animation (i.e., constant mesh topology) with mesoscopic details, using a final geometry of several orders of magnitude less than the raw point cloud density achieved with the presented demonstration subjects. As presented by Deng et al. [80], the Facial Action Coding System or similar could be used for animation to parameterize the face in a group of muscular actions for efficient replay. Such animations remain to be tested on this rig.

The camera array synchronization and rolling shutter compensation methods proposed by Bradley et al. [41] should be experimented on the system for preparing for computer animation, as both these issues are present in the current setup. Synchronization issues do not completely prohibit from using the rig, but the resulting quality may be suboptimal. However, generation of initial test content is feasible.

For small subjects such as a human head, the chosen 50 mm focal length in combination with the crop frame camera presented to be optimal; a wider field of view would require the cameras to be closer to each other, which is not physically possible. On the other hand, a smaller field of view would require the cameras to be unnecessarily far away from the subject, which becomes infeasible for larger subjects.

The rig captures the complex geometry and texture in preferably diffuse lighting; For rendering the object in a new lighting, the object's surface material may be captured with a different system. Methods for capturing the reflectance properties of a material typically keep the material sample still and vary the properties of the light [33, 165]. Combining the different approaches would produce both rich geometry and material data for complex digital reproduction.

Final thoughts The rig was found to be suitable for high-resolution capture of human faces, a target that is considered difficult due to the lack of well noticeable pattern and strong deformations. High resolution photographs recover detail that can be used for feature matching and successful 3D reconstruction. By tweaking reconstruction parameters, applying soft makeup and using more uniform lighting, the quality can be further improved; especially the surface color is challenging to recover uniformly because of specular highlights and self-occlusion, even though the reconstruction succeeds. The recent reconstruction algorithms base on many different methods; capturing content for studying the algorithms is a significant suggestion for future work.

References

[1] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice (2Nd Ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

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

[3] Fredrik Hollsten. The effect of image quality on the reconstruction of 3D geometry from photographs. Master's thesis, Aalto University School of Science, 2013.

[4] 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.

[5] 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. Theme Issue: Image Analysis and Image Engineering in Close Range Photogrammetry.

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

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

[8] Ioannis Pitas. *Digital image processing algorithms and applications*. John Wiley & Sons, 2000.

[9] Allen R Greenleaf. *Photographic optics*. Macmillan, 1950.

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

[11] Canon Camera Museum. EF50mm f/1.8 II. http://www.canon.com/camera-museum/lens/ef/data/standard/ef_50_18ii.html. Accessed 2015-05-09.

[12] Rudolf Kingslake. *Optics in photography*, volume 6. SPIE Press, 1992.

[13] Canon. EOS-1D X. http://cpn.canon-europe.com/content/education/technical/eos_1d_x_explained.do. Accessed 2014-11-08.

[14] Duane C Brown. Decentering distortion of lenses. *Photometric Engineering*, 32(3):444–462, 1966.

[15] Duane C Brown. Close-range camera calibration. *Photogramm. Eng*, 37:855–866, 1971.

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

[17] Jason P De Villiers, F Wilhelm Leuschner, and Ronelle Geldenhuys. Centi-pixel accurate real-time inverse distortion correction. In *International Symposium on Optomechatronic Technologies*, volume 7266/11, pages 1–8. International Society for Optics and Photonics, 2008.

[18] George Wolberg. *Digital image warping*, volume 10662. IEEE computer society press Los Alamitos, CA, 1990.

[19] Bob Caspe. How an electronic shutter works in a CMOS camera. http://caspegroup.com/HowanelectronicshutterworksinaCMOScamera.pdf. Accessed 2014-11-08.

[20] Kodak. Application note: Shutter operations for CCD and CMOS image sensors. http://www.isgchips.com/pdf/Shutter_Operations_Kodak_App_Note.pdf. Accessed 2014-11-08.

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

[22] Calvin Hass, ImpulseAdventure. Shutter lag and startup time comparison. http://www.impulseadventure.com/photo/shutter-lag.html. Accessed 2014-11-08.

[23] Junichi Nakamura. *Image sensors and signal processing for digital still cameras*. CRC Press, 2005.

[24] Abbas El Gamal and Helmy Eltoukhy. CMOS image sensors. *IEEE Circuits and Devices Magazine*, 21(3):6–20, 2005.

[25] Jong B. Park. Implementing image-processing pipelines in digital cameras. http://scien.stanford.edu/pages/labsite/2003/psych221/projects/03/piquant/index.htm, 2003. Accessed 2014-11-08.

[26] Stuart A Taylor et al. CCD and CMOS imaging array technologies: technology review. *Relatorio Tecnico EPC-1998-106, Cambridge Laboratory*, 1998.

[27] Dave Litwiller. CCD vs. CMOS. *Photonics Spectra*, 35(1):154–158, 2001.

[28] Shree K Nayar, Katsushi Ikeuchi, and Takeo Kanade. Surface reflection: physical and geometrical perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 13(7):611–634, 1991.

[29] Fred E Nicodemus. Directional reflectance and emissivity of an opaque surface. *Applied Optics*, 4(7):767–773, 1965.

[30] Berthold KP Horn. Determining lightness from an image. *Computer graphics and image processing*, 3(4):277–299, 1974.

[31] 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.

[32] Abhijeet Ghosh, Graham Fyffe, Borom Tunwattanapong, Jay Busch, Xueming Yu, and Paul Debevec. Multiview face capture using polarized spherical gradient illumination. *ACM Transactions on Graphics*, 30(6):129:1–129:10, December 2011.

[33] Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. Practical svbrdf capture in the frequency domain. *ACM Transactions on Graphics*, 32(4):110:1–110:12, July 2013.

[34] Michael Langford. *Basic photography*. Taylor & Francis, 2000.

[35] Alexander Hornberg. *Handbook of machine vision*. John Wiley & Sons, 2007.

[36] National Instruments. Choosing the right camera bus. http://www.ni.com/white-paper/5386/en/, 2013. Accessed 2014-11-08.

[37] Iain E Richardson. *H. 264 and MPEG-4 video compression: video coding for next-generation multimedia*. John Wiley & Sons, 2004.

[38] Magic Lantern. http://www.magiclantern.fm. Accessed 2014-10-22.

[39] Charles A Poynton. *A technical introduction to digital video*. J. Wiley, 1996.

[40] Robert B Musburger. *Single-camera video production*. Taylor & Francis, 2010.

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

[42] Sick Integrated Vision Products AB. Machine vision introduction. www.sick.com/uk/en-uk/home/products/product_portfolio/Documents/MachineVisionIntroduction2_2_web.pdf, 2006. Accessed 2014-11-09.

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

[44] Elan Dubrofsky. *Homography estimation*. PhD thesis, University of British Columbia, 2009.

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

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

[47] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[48] Jean-Yves Bouguet. Camera Calibration Toolbox for Matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`. Accessed 2014-04-18.

[49] Tomáš Svoboda, Daniel Martinec, and Tomáš Pajdla. A convenient multicamera self-calibration for virtual environments. *PRESENCE: teleoperators and virtual environments*, 14(4):407–422, 2005.

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

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

[52] Changchang Wu. SiftGPU: a GPU implementation of scale invariant feature transform (SIFT). *University of North Carolina at Chapel Hill. http://www.cs.unc.edu/ ccwu/siftgpu*, 2007.

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

[54] Marc Pollefeys. Visual 3D modeling from images. In *VMV*, page 3, 2004.

[55] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.

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

[57] Andrew W. Fitzgibbon, Geoff Cross, and Andrew Zisserman. Automatic 3D model construction for turn-table sequences. In *Proceedings of the European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, SMILE'98, pages 155–170, London, UK, UK, 1998. Springer-Verlag.

[58] Bernd Bickel, Mario Botsch, Roland Angst, Wojciech Matusik, Miguel Otaduy, Hanspeter Pfister, and Markus Gross. Multi-scale capture of facial geometry and motion. *ACM Transactions on Graphics*, 26(3), July 2007.

[59] 3dMD. 3D imaging systems and software. `http://www.3dmd.com`. Accessed 2014-11-09.

[60] Thabo Beeler, Bernd Bickel, Paul Beardsley, Bob Sumner, and Markus Gross. High-quality single-shot capture of facial geometry. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 40:1–40:9, New York, NY, USA, 2010. ACM.

[61] Thabo Beeler, Fabian Hahn, Derek Bradley, Bernd Bickel, Paul Beardsley, Craig Gotsman, Robert W. Sumner, and Markus Gross. High-quality passive facial performance capture using anchor frames. *ACM Transactions on Graphics*, 30(4):75:1–75:10, July 2011.

[62] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Transactions on Graphics*, 25(3):835–846, July 2006.

[63] R Toldo, F Fantinia, L Gionaa, S Fantonia, and A Fusiellob. Accurate multiview stereo reconstruction with fast visibility integration and tight disparity bounding. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1(1):243–249, 2013.

[64] Roberto Toldo. *Towards automatic acquisition of high-level 3D models from images.* PhD thesis, Verona University, 2013.

[65] Rodrigo L. Carceroni and Kiriakos N. Kutulakos. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3D motion, shape and reflectance. *International Journal of Computer Vision*, 49(2-3):175–214, September 2002.

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

[67] H-H Vu, Patrick Labatut, J-P Pons, and Renaud Keriven. High accuracy and visibility-consistent dense multiview stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):889–901, 2012.

[68] Ju Yong Chang, Haesol Park, In Kyu Park, Kyoung Mu Lee, and Sang Uk Lee. GPU-friendly multi-view stereo reconstruction using surfel representation and graph cuts. *Computer Vision and Image Understanding*, 115(5):620 – 634, 2011. Special issue on 3D Imaging and Modelling.

[69] Yasutaka Furukawa and Jean Ponce. Patch-based Multi-view Stereo Software (PMVS-Version 2). *PMVS2, University of Washington, Department of Computer Science and Engineering*, 14, 2012.

[70] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[71] P. Labatut, J.-P. Pons, and R. Keriven. Robust and efficient surface reconstruction from range data. *Computer Graphics Forum*, 28(8):2275–2290, 2009.

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

[73] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment – a modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 2000.

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

[75] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 163–169, New York, NY, USA, 1987. ACM.

[76] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):29:1–29:13, July 2013.

[77] Willow Garage and Open Perception. Point Cloud Library. http://pointclouds.org. Accessed 2014-04-18.

[78] Paul S Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, 1986.

[79] Lingyun Liu, Ioannis Stamos, Gene Yu, George Wolberg, and Siavash Zokai. Multiview geometry for texture mapping 2D images onto 3D range data. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2293–2300. IEEE, 2006.

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

[81] Keith Waters. A muscle model for animation three-dimensional facial expression. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 17–24, New York, NY, USA, 1987. ACM.

[82] Hao Li, Bart Adams, Leonidas J. Guibas, and Mark Pauly. Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics*, 28(5):175:1–175:10, December 2009.

[83] Li Zhang, Noah Snavely, Brian Curless, and Steven M Seitz. Spacetime faces: High-resolution capture for modeling and animation. In *Data-Driven 3D Facial Animation*, pages 248–276. Springer, 2007.

[84] Thabo Beeler, Fabian Hahn, Derek Bradley, Bernd Bickel, Paul Beardsley, Craig Gotsman, Robert W. Sumner, and Markus Gross. High-quality passive facial performance capture using anchor frames. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, pages 75:1–75:10, New York, NY, USA, 2011. ACM.

[85] Morten Bojsen-Hansen, Hao Li, and Chris Wojtan. Tracking surfaces with evolving topology. *ACM Transactions on Graphics*, 31(4):53:1–53:10, July 2012.

[86] Jonathan Starck and Adrian Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007.

[87] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics*, 23(3):399–405, August 2004.

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

[89] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, March 2001.

[90] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. *ACM Transactions on Graphics*, 27(3):99:1–99:9, August 2008.

[91] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[92] Carlo Tomasi and Takeo Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991.

[93] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012.

[94] Georg Nebehay and Roman Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *Winter Conference on Applications of Computer Vision*. IEEE, March 2014.

[95] James J Gibson. *The perception of the visual world*. Houghton Mifflin, 1950.

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

[97] Simon A. Eugster. slowmovideo: Slow-motion for video with optical flow. Master's thesis, ETH Zürich, 2011.

[98] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, October 1994.

[99] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. IEEE, 2001.

[100] J-P Pons, Renaud Keriven, and Olivier Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 822–827. IEEE, 2005.

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

[102] David W. Eggert, Andrew W. Fitzgibbon, and Robert B. Fisher. Simultaneous registration of multiple range views for use in reverse engineering of CAD models. *Computer Vision and Image Understanding*, 69(3):253 – 272, 1998.

[103] Daniel F Huber and Martial Hebert. Fully automatic registration of multiple 3D data sets. *Image and Vision Computing*, 21(7):637–650, 2003.

[104] Benedict J. Brown and Szymon Rusinkiewicz. Global non-rigid alignment of 3-D scans. *ACM Transactions on Graphics*, 26(3), July 2007.

[105] IR-Enterntainment Ltd. Infinite realities. http://ir-ltd.net. Accessed 2014-10-21.

[106] Ten24. Ten 24 3D scanning service. http://www.ten24.info. Accessed 2014-10-21.

[107] The Capture Lab Inc. Likeness capture. http://www.capturelab.com/likeness-capture/. Accessed 2014-10-21.

[108] Agisoft. Community Forum. http://www.agisoft.com/forum/. Accessed 2014-10-21.

[109] 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.

[110] T Al-Anezi, B Khambay, MJ Peng, E O'Leary, X Ju, and A Ayoub. A new method for automatic tracking of facial landmarks in 3D motion captured images (4D). *International journal of oral and maxillofacial surgery*, 42(1):9–18, 2013.

[111] George Borshukov, Dan Piponi, Oystein Larsen, J. P. Lewis, and Christina Tempelaar-Lietz. Universal capture: Image-based facial animation for "the matrix reloaded". In *ACM SIGGRAPH 2003 Sketches & Applications*, SIGGRAPH '03, pages 1–1, New York, NY, USA, 2003. ACM.

[112] Sony Corporation. HDWF900R. https://pro.sony.com/bbsc/ssr/product-HDWF900R/. Accessed 2014-11-09.

[113] Tom's Hardware. HDD charts 2013, write throughput average. http://www.tomshardware.com/charts/hdd-charts-2013/-04-Write-Throughput-Average-h2benchw-3.16,2904.html. Accessed 2014-11-08.

[114] Tom's Hardware. SSD charts 2013, AS-SSD sequential write. http://www.tomshardware.com/charts/ssd-charts-2013/AS-SSD-Sequential-Write,2783.html. Accessed 2014-11-08.

[115] Tom's Hardware. SD card sequential write benchmarks 2014. http://www.tomshardware.com/charts/sd-cards-2014/-2-Sequential-Write-MB-s,3473.html. Accessed 2014-11-08.

[116] Tom's Hardware. CompactFlash card sequential write benchmarks 2014. http://www.tomshardware.com/charts/compactflash-charts/-2-Sequential-Write-MB-s,3489.html. Accessed 2014-11-08.

[117] SD Association. SD standard overview: Bus speed. https://www.sdcard.org/developers/overview/bus_speed/. Accessed 2015-05-09.

[118] Magic Lantern Forum. http://www.magiclantern.fm/forum/. Accessed 2014-10-22.

[119] Thomann. https://www.thomann.de/. Accessed 2014-10-22.

[120] Manfrotto mini ball head 494. http://www.manfrotto.com/mini-ball-head-494. Accessed 2015-05-09.

[121] Manfrotto super clamp. http://www.manfrotto.com/product/0/035/_/Super_Clamp_without_Stud. Accessed 2015-05-09.

[122] Data Translation Inc. Digital I/O USB DAQ modules. http://www.datatranslation.com/products/dataacquisition/usb/digital-io/. Accessed 2014-11-09.

[123] PocketWizard. Wireless transmitters and receivers. http://www.pocketwizard.com/products/transmitter_receiver/. Accessed 2015-05-09.

[124] Lukasz Panek. DIY wired remote control for Canon EOS cameras. http://www.doc-diy.net/photo/eos_wired_remote/. Accessed 2014-10-22.

[125] Breeze Systems. Triggering multiple cameras. http://www.breezesys.com/MultiCamera/release.htm. Accessed 2014-11-09.

[126] Arduino. http://www.arduino.cc. Accessed 2015-05-09.

[127] STMicroelectronics. Nucleo-F401RE STM32 Nucleo development board. http://st.com/nucleoF401RE-pr. Accessed 2014-10-22.

[128] Toshiba. TLP621, TLP621-2, TLP621-4 optocouplers (datasheet). http://www.farnell.com/datasheets/57440.pdf. Accessed 2014-10-22.

[129] CadSoft Inc. EAGLE PCB Design Software. http://www.cadsoftusa.com/eagle-pcb-design-software/. Accessed 2014-10-22.

[130] Mbed. ST Nucleo F401RE. http://developer.mbed.org/platforms/ST-Nucleo-F401RE/. Accessed 2014-10-22.

[131] ARM. GNU tools for ARM Embedded Processors in Launchpad. https://launchpad.net/gcc-arm-embedded. Accessed 2014-11-09.

[132] Breeze Systems. DSLR Remote Pro Multi-Camera. http://www.breezesys.com/MultiCamera/index.htm. Accessed 2014-11-09.

[133] Kuvacode. Smart shooter. http://kuvacode.com/smart-shooter. Accessed 2014-11-09.

[134] gPhoto. http://www.gphoto.org. Accessed 2014-10-22.

[135] Canon Digital Imaging Developer Programme. ED-SDK. https://www.didp.canon-europa.com. Accessed 2014-10-22.

[136] Nikon Corporation. SDKs for digital imaging products. https://sdk.nikonimaging.com/apply/. Accessed 2014-11-09.

[137] Sony Corporation. Camera Remote API beta SDK. https://developer.sony.com/downloads/camera-file/sony-camera-remote-api-beta-sdk/. Accessed 2014-11-09.

[138] Olympus Imaging America Inc. Olympus announces software developer's kit to create applications for award-winning digital camera line. http://www.olympusamerica.com/cpg_section/cpg_pressDetails.asp?pressNo=105. Accessed 2014-11-09.

[139] Society for imaging science and technology. PTP standards. http://www.imaging.org/ist/resources/standards/ptp-standards.cfm, 2009. Accessed 2014-11-09.

[140] wxWidgets. https://www.wxwidgets.org. Accessed 2014-10-22.

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

[142] Andrea Vedaldi. SIFT++, a lightweight C++ implementation of SIFT, http://vision.ucla.edu/ vedaldi/code/siftpp/siftpp.html, 2011.

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

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

[145] Pierre Moulon, Alessandro Bezzi, et al. Python photogrammetry toolbox: A free solution for three-dimensional documentation, 2011.

[146] Autodesk. 123D Catch. http://www.123dapp.com/catch. Accessed 2014-10-22.

[147] Acute3D. Smart3DCapture. http://www.acute3d.com/smart3dcapture/. Accessed 2014-10-22.

[148] Agisoft. Photoscan. http://www.agisoft.com. Accessed 2014-10-22.

[149] Faceshift AG. Faceshift. http://www.faceshift.com/product/. Accessed 2014-10-22.

[150] 3DFlow. 3DF Zephyr Pro. http://www.3dflow.net/3df-zephyr-pro-3d-models-from-photos/. Accessed 2014-10-22.

[151] Pix4D. Pix4DMapper. http://pix4d.com/products/. Accessed 2014-10-22.

[152] The Pixel Farm Ltd. Pixel farm. http://www.thepixelfarm.co.uk/. Accessed 2014-10-22.

[153] Science-D-Visions. 3DEqualizer. http://www.sci-d-vis.com/index.php#?site=products&id=products_summary. Accessed 2014-10-22.

[154] Team Bondi. *L.A. Noire (video game)*. Rockstar Games, 2011.

[155] Pendulum Studios. Alter ego facial performance. http://www.studiopendulum.com/?page_id=204. Accessed 2014-10-22.

[156] FARO. Software solutions. http://www.faro.com/en-us/products/faro-software. Accessed 2014-10-22.

[157] Dimensional Imaging Ltd. Di4D. http://www.di4d.com/index.html. Accessed 2014-10-22.

[158] Vicon Motion Systems Inc. http://www.vicon.com. Accessed 2014-11-09.

[159] Geodetic Systems Inc. V-STARS. http://www.geodetic.com/products.aspx. Accessed 2014-11-09.

[160] Agisoft Community Forum. Topic: Algorithms used in photoscan. http://www.agisoft.com/forum/index.php?topic=89.0. Accessed 2014-10-21.

[161] Red Giant. PluralEyes. http://www.redgiant.com/products/all/pluraleyes/. Accessed 2014-10-22.

[162] Adobe. Synchronizing audio and video with merge clips. http://helpx.adobe.com/premiere-pro/using/synchronizing-audio-video-merge-clips.html. Accessed 2014-10-22.

[163] Structure from Motion for Unordered Image Collections. Frequently asked questions about Bundler. http://www.cs.cornell.edu/~snavely/bundler/faq.html. Accessed 2014-10-22.

[164] Techradar. How EA brings your FIFA 14 players to photo-realistic life. http://www.techradar.com/news/gaming/how-ea-brings-your-fifa-players-to-photo-realistic-life-1139060. Accessed 2014-10-21.

[165] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 145–156. ACM Press/Addison-Wesley Publishing Co., 2000.

A List of hardware

The hardware listed below is a comprehensive minimal list that is required for the system. Additionally, light sources should be used as necessary.

- 9 x Canon EOS 700D DSLR camera body
- 9 x Canon EF 50mm f/1.8 II lens
- 9 x Canon ACK-E8 Power adapter
- 9 x Transcend SDHC class 10 UHS-I 16GB memory card
- 9 x Manfrotto 494 ball head
- 9 x Manfrotto 035 Super Clamp, with Manfrotto 036-38 studs
- 3 x D-link DUB-H4 USB hub
- 4 x Millenium LST-310, a three-legged lighting stand
- Custom built remote trigger board using ST Nucleo-F401RE, see Section B
- Canon RC-6 wireless remote control for testing
- Auxiliary extension cords
- 2 x bags for the stands
- 2 x hard equipment cases with cut holes for all fragile parts of the rig

B Remote trigger

The designed remote trigger consists of a Nucleo-F401RE microcontroller platform by ST microelectronics [127], and supporting hardware for connecting the cameras. The microcontroller was programmed using the Mbed library [130]. Each camera connects to the trigger via a 2.5mm stereo plug, as the cameras have a standard 2.5mm jack for remote trigger. To drive the cameras individually, the remote wires are connected with opto-isolators (TLP621-2).

The ground signals of the isolators in the built circuit are connected together by mistake, so the cameras are not completely isolated; only their signal wires are. However, common grounding is generally not an issue. Furthermore, the USB connections of all cameras also connect to each other and to the trigger box via the computer they are connected to, which cannot be easily circumvented. The git repository contains the project files in a format for the Cadsoft EAGLE pcb design software [129]. The diagrams for schematic and PCB layout are given here for illustration only.

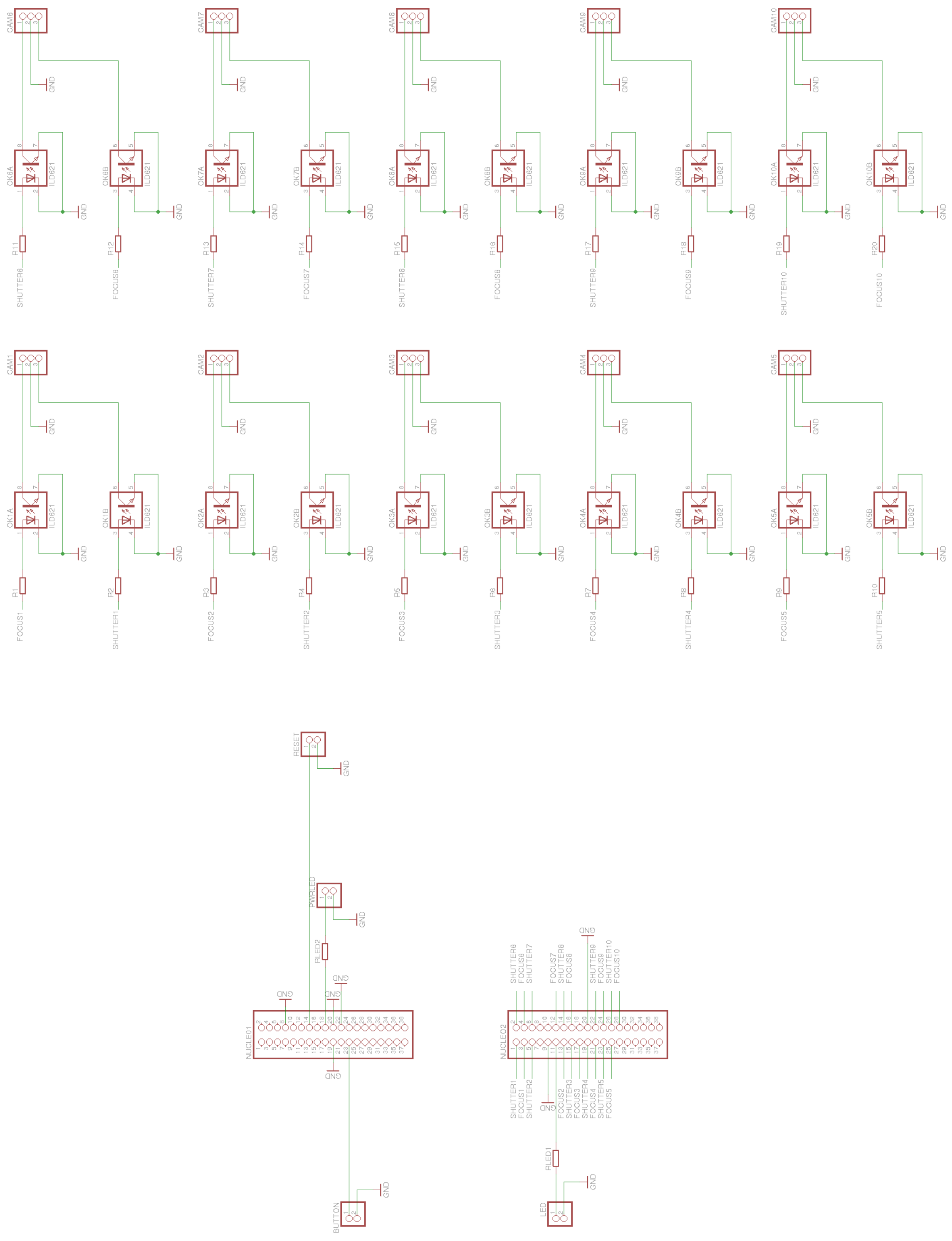


Figure 61: Remote trigger schematic

B.1 Schematic

See Figure 61 for a complete schematic. The two pin headers are those of the Nucleo board, and the order of the opto-isolators and connectors matches with the physical order of the connectors on the PCB. Additionally, the LEDs and buttons were routed to the side of the board to be accessible when installed in the enclosure.

B.2 Printed circuit board layout

The circuit board dimensions are 5 x 3.9 inches (approximately 127 x 99 mm) designed to fit in the given enclosure. All electrical connections can be realized with just one layer; another layer was used as a ground plane, but the ground signals are routed also along with the signals. The circuit is also simple enough to implement on a prototyping board with jumper wires. Figure 62 shows the board layout without a ground plane drawn.

B.3 Complete device

The PCB was engraved with a milling machine produced by LPKF Laser & Electronics AG. A recycled enclosure was used with custom drilled end caps for the 2.5 mm jacks for the camera connections in one end, and Nucleo board's micro-USB, LEDs and buttons on other end. For replicating the device identically, a similar box with inside dimensions of at least 127 by 99 mm for the PCB and 35 mm of height can be used. Of the two buttons, the red one is the user button, and black one is the reset button. The red LED shows that the power is on, and the yellow LED is on when the focus command has been received.

The completed device is shown in Figure 63, and Figure 64 shows the same with the case opened, exposing the circuit board, soldered parts and the connectors mounted on the end.

B.4 Parts list

- ST Microelectronics Nucleo-F401RE microcontroller platform
- Mini USB cable
- 10 x stereo 2.5 mm jack
- 10 x stereo 2.5 mm plug-plug 3 meter cable
- 10 x TLP621-2 opto-isolator; interchangeable with any with identical pinout
- 20 x 220 ohm resistor
- With an enclosure, two LEDs with current limiting resistors and two pushbuttons should be used

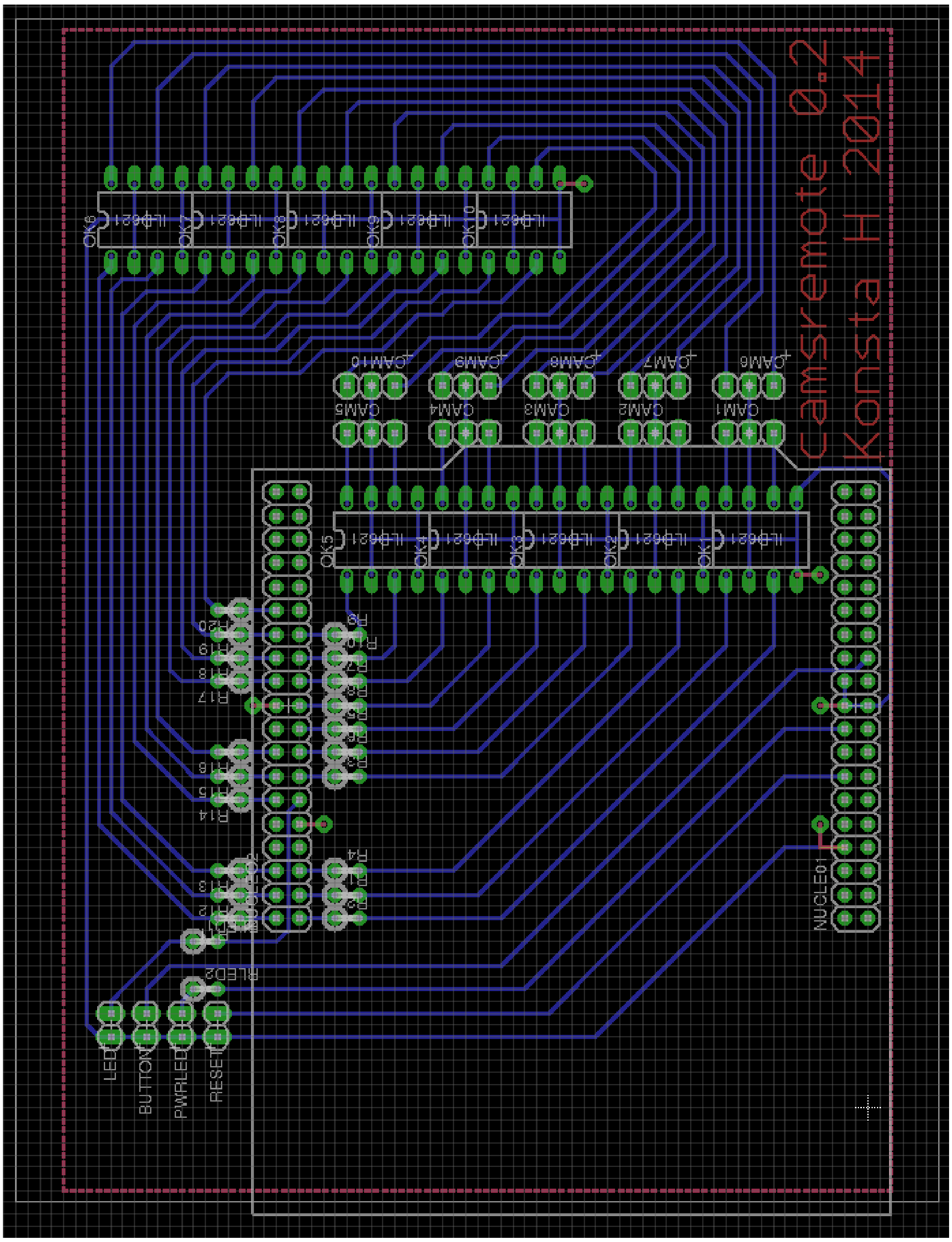


Figure 62: Remote trigger circuit board (PCB) layout

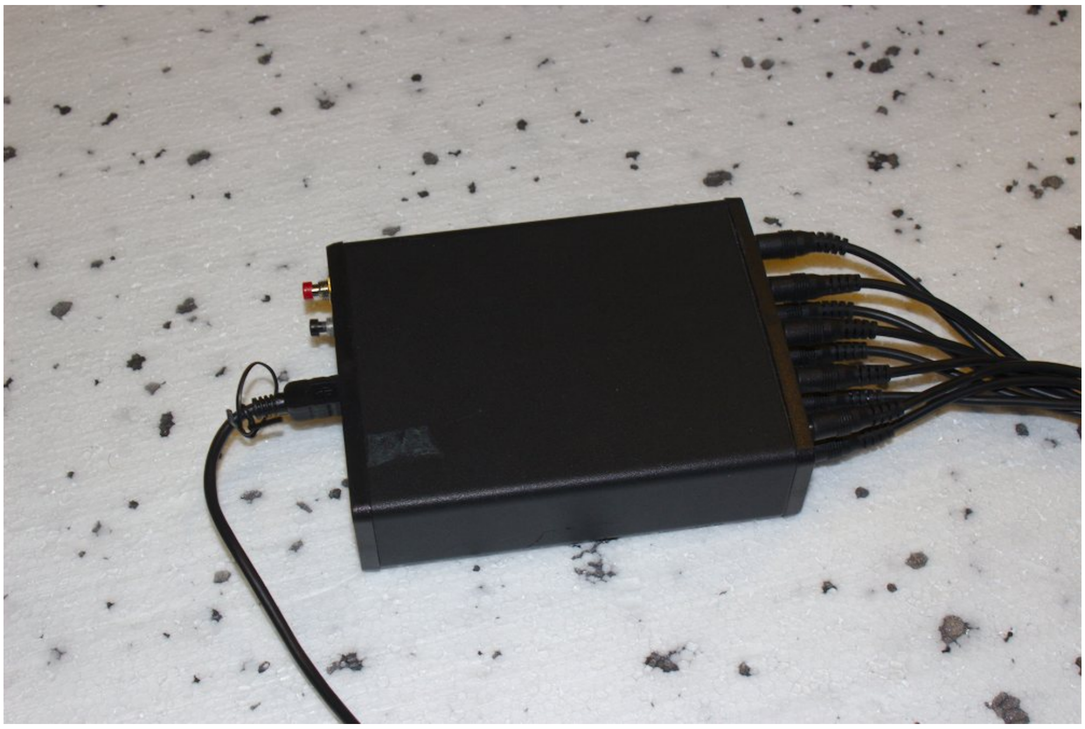


Figure 63: The remote tool enclosure with the lid closed and wires connected

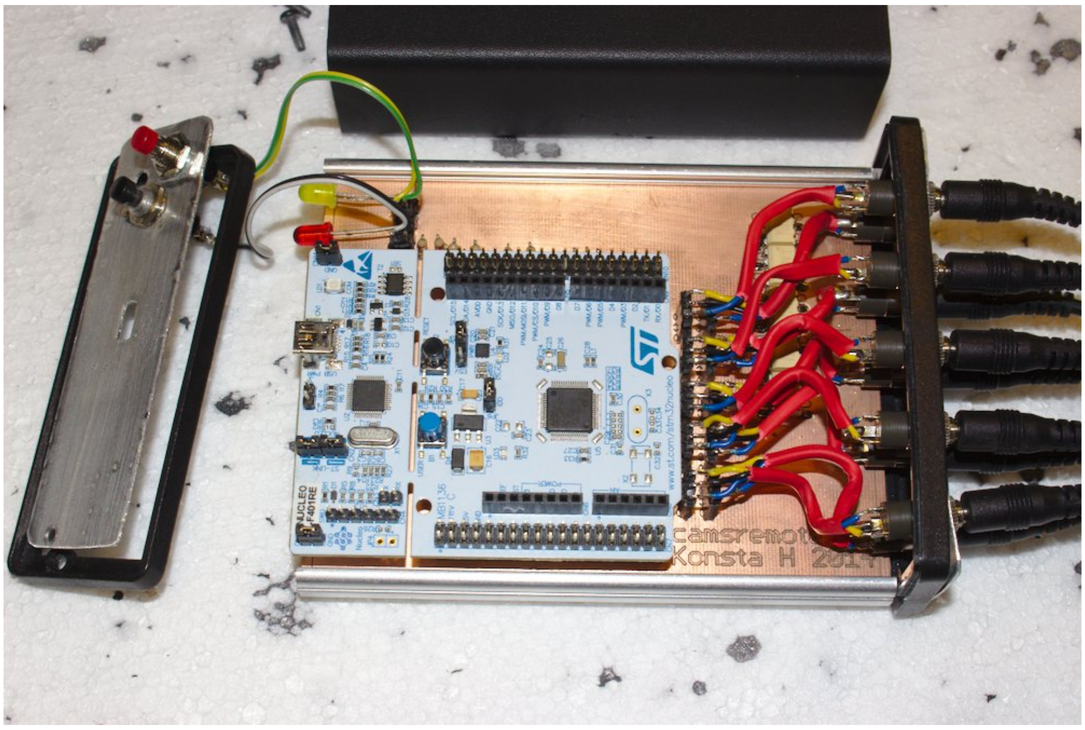


Figure 64: Components installed to the board and fitted in the enclosure