The Need 4 Speed in Real-Time Dense Visual Tracking

ADARSH KOWDLE*, CHRISTOPH RHEMANN*, SEAN FANELLO*, ANDREA TAGLIASACCHI†, JONATHAN TAYLOR†, PHILIP DAVIDSON†, MINGSONG DOU†, KAIWEN GUO†, CEM KESKIN†, SAMEH KHAMIS†, DAVID KIM†, DANHANG TANG†, VLADIMIR TANKOVICH†, JULIEN VALENTIN†, and SHAHRAM IZADI†,
Google Inc.

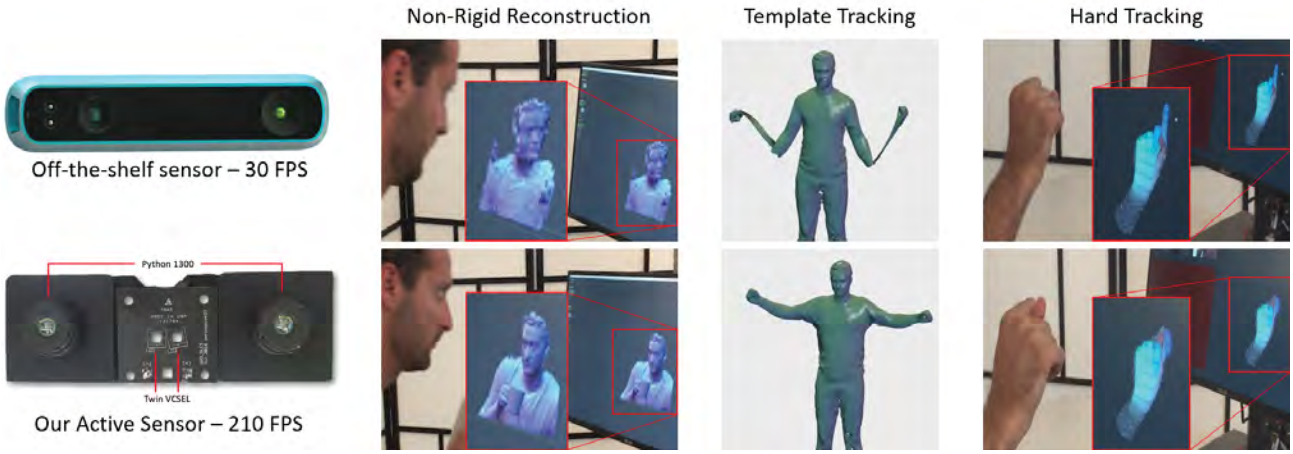


Fig. 1. Our high speed sensor can be exploited for efficient, low latency, and high quality computer vision algorithms. As illustrated, our system allows unprecedented robustness to tracking and frame-to-frame correspondence tasks, where commercially available depth cameras would typically fail.

The advent of consumer depth cameras has incited the development of a new cohort of algorithms tackling challenging computer vision problems. The primary reason is that depth provides direct geometric information that is largely invariant to texture and illumination. As such, substantial progress has been made in human and object pose estimation, 3D reconstruction and simultaneous localization and mapping. Most of these algorithms naturally benefit from the ability to accurately track the pose of an object or scene of interest from one frame to the next. However, commercially available depth sensors (typically running at 30fps) can allow for large inter-frame motions to occur that make such tracking problematic. A high frame rate depth camera would thus greatly ameliorate these issues, and further increase the tractability of these computer vision problems. Nonetheless, the depth accuracy of recent systems for high-speed depth estimation [Fanello et al. 2017b] can degrade at high frame rates. This is because the active illumination employed produces a low SNR and thus a high exposure time is required to obtain a dense accurate depth image. Furthermore in the presence of rapid motion, longer exposure times produce artifacts due to motion blur, and necessitates a lower frame rate that introduces large inter-frame motion that often yield tracking failures. In contrast, this paper proposes a novel combination of hardware and software components that avoids the need to compromise between a dense accurate depth map and a high frame rate. We document the creation of a full 3D capture system for high speed and quality depth estimation, and demonstrate its advantages in a variety of tracking and reconstruction tasks. We extend the state of the art active stereo algorithm presented in Fanello et al. [2017b] by adding a space-time feature in the matching phase. We also propose a machine learning based depth refinement step that is an order of magnitude faster than traditional post-processing methods. We quantitatively and qualitatively demonstrate the benefits of the proposed algorithms in the acquisition of geometry in motion. Our pipeline executes in 1.1ms leveraging modern GPUs and off-the-shelf cameras and illumination components. We show how the sensor can be employed in many different applications, from [non-]rigid reconstructions to hand/face tracking. Further, we show many advantages over existing state of the art depth camera technologies beyond framerate, including latency, motion artifacts, multi-path errors, and multi-sensor interference.

CCS Concepts: • **Computing methodologies** → **Computer vision**; **Image and video acquisition**;

Additional Key Words and Phrases: depth sensor, high framerate tracking.

ACM Reference Format:
Adarsh Kowdle, Christoph Rhemann, Sean Fanello, Andrea Tagliasacchi, Jonathan Taylor, Philip Davidson, Mingsong Dou, Kaiwen Guo, Cem Keskin, Sameh Khamis, David Kim, Danhang Tang, Vladimir Tankovich, Julien Valentin, and Shahram Izadi. 2018. The Need 4 Speed in Real-Time Dense Visual Tracking. *ACM Trans. Graph.* 37, 6, Article 220 (November 2018), 14 pages. https://doi.org/10.1145/3272127.3275062

*Authors equally contributed to this work.
†Authors equally contributed to this work.
This work was conducted at perceptiveIO.

Authors' address: Adarsh Kowdle; Christoph Rhemann; Sean Fanello; Andrea Tagliasacchi; Jonathan Taylor; Philip Davidson; Mingsong Dou; Kaiwen Guo; Cem Keskin; Sameh Khamis; David Kim; Danhang Tang; Vladimir Tankovich; Julien Valentin; Shahram Izadi Google Inc.

0730-0301/2018/11-ART220
https://doi.org/10.1145/3272127.3275062

1 INTRODUCTION

Computer vision tracking problems span a wide range of applications, from tracking simple parametric models [Nakabo et al. 2000; Stuhmer et al. 2015] to more complex non-rigid shapes such as human bodies [Dou et al. 2017]. However, when the tracking problem involves many degrees of freedom (e.g. non-rigid reconstruction of body parts [Dou et al. 2017, 2016; Zollhöfer et al. 2014]) depth sensors offer crucial advantages over RGB cameras or IMUs. In particular, depth images provide richer geometric data while simultaneously achieving invariance to surface texture and lighting conditions. The importance of high framerate (fps) for tracking was demonstrated by the research community (see e.g. [Handa et al. 2012; Kim et al. 2016; Nakabo et al. 2000; Stuhmer et al. 2015]), as well as by the widespread adoption of higher speed RGB cameras and/or inertial measurement units (IMUs) in commercial 3D rigid pose estimation and tracking systems for VR/AR (e.g. Oculus, HTC Vive, Microsoft HoloLens, and Google Tango). High framerate cameras are also used in commercial MOCAP systems (e.g. OptiTrack or VICON) for the tracking of a *sparse* set of optical markers at framerates higher than 1kHz. In our work, we desire to bridge the gap between MOCAP systems and depth sensors towards enabling the tracking of *dense* geometry in *fast* motion.

Depth revolution. The recent availability of low-cost commodity 3D capture systems such as the Microsoft Kinect has revolutionized our ability to tackle challenging computer vision and human computer interaction problems such as body part classification [Shotton et al. 2011; Sridhar et al. 2015], hand pose estimation [Keskin et al. 2012; Taylor et al. 2016; Tkach et al. 2016], action recognition [Fanello et al. 2013a,b], 3D scanning [Innmann et al. 2016; Izadi et al. 2011] and 3D scene understanding [Bleyer et al. 2012; Li et al. 2015; Valentin et al. 2015]. Depth sensors are also widely used in robotics [Ciliberto et al. 2012; Fanello et al. 2014; Gori et al. 2013] as well as in virtual and augmented reality [Orts-Escolano et al. 2016]. Despite their widespread use, depth sensors typically operate at 30fps – a framerate optimized for color cameras in the motion picture industry. However, this low framerate poses considerable challenges to computer vision applications because significant motion can occur between consecutive temporal frames.

Coping with insufficient framerate. Many computer vision algorithms explicitly assume small frame-to-frame motion (e.g. optical flow [Horn and Schunck 1981]) or implicitly by assuming that the initial solution is close to the expected global optimum and can be found by gradient based local optimization (e.g. ICP [Chen and Medioni 1992]). When the small motion assumption is violated suboptimal solutions are produced. As a result, recent methods design complex and compute-intensive pipelines in order to cope with high frame-to-frame variations. For instance, recent parametric models for non-rigid tracking [Taylor et al. 2016], use sophisticated reinitialization strategies to avoid getting stuck in local minima. Other dynamic reconstruction pipelines [Dou et al. 2016; Orts-Escolano et al. 2016] rely on fast frame-to-frame *semantic* correspondences [Wang et al. 2016], increasing the overall compute. Despite these efforts, these tracking systems still struggle to cope with the huge search space as well as motion artifacts and appearance changes that occur at lower framerate capture.

High-speed capture. Since the displacement of objects in an image is linearly dependent on their speed and the sampling frequency of the camera, large frame-to-frame motion problems can be attacked by reducing the time interval between consecutive frames. Hence, we note that employing high frame-rate depth cameras make numerous computer vision problems easier to solve. The advent of high speed RGB sensors and IMUs in mobile phones and consumer cameras have already paved the way towards robust 3D pose estimation; e.g. recently Kiani Galoogahi et al. [2017] acquired high framerate datasets for 2D object tracking. Very recent work [Fanello et al. 2017b] has shown a fast 3D capture camera that processes depth maps in 2.5ms, however the method employs a single-mode laser coupled with a Diffractive Optical Element (DOE) as active illumination, similar to the one used in Kinect V1. These active illuminators have a very low SNR and therefore they struggle with low reflectivity areas and large distances. As a consequence, without a fairly high exposure time of the camera (i.e. 30 ms), the quality of depth significantly decreases (see Figure 14). However, with such a high exposure time motion artifacts are more easily introduced. Thus, in practice, this system cannot produce a high frame rate depth stream without sacrificing depth quality.

In this paper we address these issues: our 3D capture technology is based on depth-from-stereo with active structured light infrared illumination that leverages Vertical Cavity Surface Emitting Laser (VCSEL) technology [Moench et al. 2016]. This allows for a strong SNR even in low exposure settings, i.e. $1 - 2$ms, enabling very high framerates without sacrificing depth quality (see Figure 14). As result, we drastically reduce the difficulty of the non-rigid tracking problems by leveraging high frame rate depth estimation. The overall computational budget required for high-framerate tracking itself is *comparable* to tracking at low speeds (due to significantly lower per frame computation costs), but with the benefit of *significantly more accurate* tracking results.

Contributions. We propose a high speed, fast exposure, and low latency dense capture pipeline for geometry in fast motion that runs in 1.1ms end-to-end on an Nvidia TitanX. Our contributions are:

- We detail all off-the-shelf hardware components needed to build a high speed and high quality 3D capture sensor.
- We contribute a hardware solution to effectively eliminate *depth bias* inherent in active illumination stereo systems.
- We provide a practical solution to avoid *lens flare* induced by the active illumination in *multi-sensor* setups.
- We extend Fanello et al. [2017a] to leverage temporal coherency with a complexity that is *independent* of patch size.
- We introduce a novel temporal prior to compute disparity.
- We devise a machine learning approach to refine the disparities by learning to efficiently invalidate outliers.
- We demonstrate the effectiveness of the system in many high-level tracking applications.

Numerous quantitative and qualitative experiments prove the effectiveness of the proposed applications for many challenging capture and tracking scenarios.

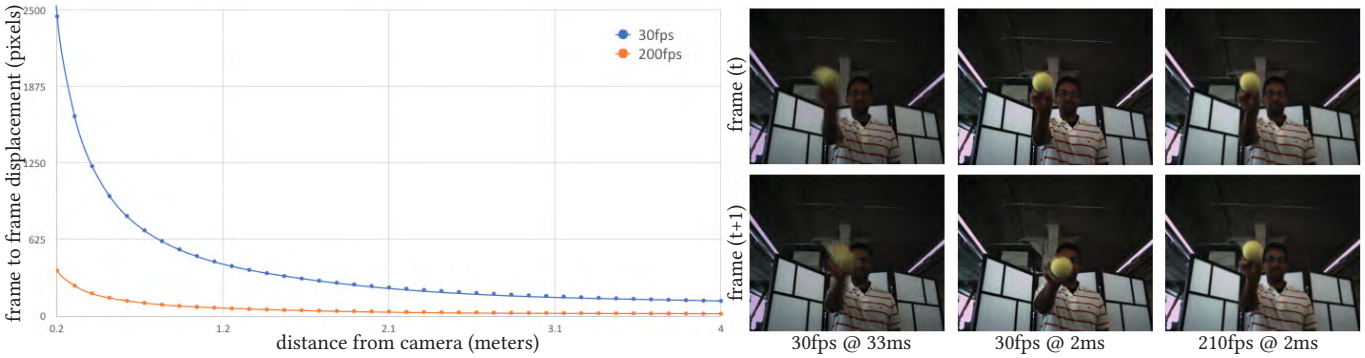


Fig. 2. (left) The screen space displacement (in pixels) between consecutive frames induced by a target object moving 12 m/s with respect to framerate and distance from the camera. (right) We display consecutive RGB frames captured at different frame-rates and exposures for a subject moving a tennis ball at a 1.5m distance. Our high frame-rate and low exposure camera results in (1) small motions and (2) limited motion blur.

2 RELATED WORK

Solving general tracking problems is one of the most active areas in computer vision. Generally speaking, tracking applications can be categorized by the degrees of freedom (DOFs) of the problem. For instance, in RGB images we may be interested in tracking the 2D position of a particular object using a known model or a template [Nakabo et al. 2000], or in a depth image we may want to infer the 3D position of the object with respect to the camera [Stuhmer et al. 2015]. Increasing the degrees of freedom also increases the complexity of the problem, for example in camera localization tasks the goal is to infer both position and orientation leading to a 6DOF estimation problem [Forster et al. 2014; Izadi et al. 2011; Newcombe et al. 2011]. Research has shown that, given a sufficiently high SNR, high frame rates greatly simplifies tracking problems [Handa 2013; Newcombe 2012]. For example, commercially available systems such as VR headsets, successfully solve the camera pose problem with a combination of higher speed RGB (IR) cameras and extremely fast IMUs [RoadToVR 2016]. For tracking high-speed rigid motion without motion blur, high-framerate pan/tilt cameras such as the one proposed by Okumura et al. [2011] can be employed, but these do not generalize to the generic tracking scenarios we consider.

Event cameras. Recently, event cameras have also shown very promising results for localization and tracking problems [Kim et al. 2016; Rebecq et al. 2017; Reinbacher et al. 2017], showing again the importance of high frame-rates for tracking purposes. However, when the degrees of freedom of the problem increase, RGB sensors are not sufficient. For these applications, 3D sensors have become the standard approach since they provide the additional depth information that helps to resolve ambiguous cases.

Table 1. Depth sensors commercially available compared with the proposed solution. Notice how we can achieve the fastest frame rate without sacrificing resolution.

Sensor	Max Res @ FPS	Min Res @ FPS
Kinect v1	$640 \times 480@30FPS$	$640 \times 480@30FPS$
Kinect v2	$512 \times 424@30FPS$	$512 \times 424@30FPS$
DUO MLX	$752 \times 480@45FPS$	$320 \times 120@320FPS$
ZED	$4416 \times 1242@15FPS$	$1344 \times 376@100FPS$
Intel SR300	$640 \times 480@60FPS$	$640 \times 480@60FPS$
Intel D435	$1280 \times 720@30FPS$	$640 \times 480@90FPS$
Proposed	**1280 × 1024@210FPS**	**640 × 512@800FPS**

High framerate systems. Some systems for high framerate depth capture have been proposed in the literature [Gong and Zhang 2010; Höfling et al. 2015; Hyun et al. 2017; Zhang et al. 2010; Zuo et al. 2013], but the required hardware is bulky and prohibitively expensive as they rely on dynamic patterns from high-speed projection technology. Conversely, our system could be realized with off-the-shelf hardware such as a $5 VSCEL and a $3 Sony IMX camera typically used in mobile phones for slow-motion videos. The hybrid system recently proposed by Lu et al. [2017] combines a low framerate depth sensors such as Kinect with a high speed RGB camera, producing 500fps depth with 20ms latency. The authors show promising results for simple tracking applications, but the generated depth maps exhibit high levels of noise, which is not acceptable for precise tracking. Although higher speed consumer depth sensors are available on the market, such as the *DUO MLX* and the *ZED*, these use passive illumination, thus they produce high speed depth at a *much* lower quality than our active sensor. Moreover passive sensors do not work in low light conditions. Further, our system captures images with 34× more pixels than the DUO MLX, and 2.5× more pixels than the ZED; see Table 1 for a comparison with commercially available sensors. Recent works in depth estimation [Fanello et al. 2016, 2017b; Keselman et al. 2017] show that triangulation systems can achieve high quality results with very low compute. In particular, Fanello et al. [2017b] presented a 210Hz depth camera, but this system suffers significant limitations when capturing *fast motion*; see Section 6. To cope with these shortcomings, we propose a novel hardware solution that uses VCSEL technology as opposed to traditional single-mode lasers and DOE systems (such as in StructureIO). In contrast to single mode lasers, we use VCSELs which are more efficient in terms of power consumption. We demonstrate better SNR performance using this hardware allowing for a lower exposure time and a reduction in generated motion artifacts. On the algorithmic side, we extend [Fanello et al. 2017b] with a novel space-time stereo matching scheme.

Space-time stereo matching. Previous work [Davis et al. 2005; Zhang et al. 2003] cast the task of depth estimation as a space-time

stereo matching task. The general idea consists of aggregating the matching cost across both space and time. These methods focus on high quality reconstruction, therefore they pay little consideration to the overall running time. As a consequence, they use an exhaustive disparity search across both the spatial and temporal domain, making them intractable for real-time performances. In addition, these methods fail for fast motion, which can result in inconsistent observations in the left and right cameras. In contrast to previous work [Davis et al. 2005; Zhang et al. 2003], our computational requirements do not increase when employing a temporal window. Indeed, our matching cost is independent of the window size. Moreover, given the high speed of the cameras and the capability of processing all the frames in real-time, we are more robust to fast motion. Finally, disparity optimization is carried out in parallel, as opposed to the sequential PatchMatch-like search of [Fanello et al. 2017b].

3 OVERVIEW

In the following sections, we show how to build the first complete system, from *hardware* design (Sec. 4) to *software* implementation (Sec. 5), capable of capturing depth images at high *framerate*, *resolution* and *quality* without artifacts typically caused by interference between multiple sensors being operated simultaneously. We qualitatively and quantitatively evaluate these improvements (Sec. 6), highlight how they significantly impact a number of real-time dense tracking applications (Sec. 7), and present the applicability of our cameras to low-latency passthrough in mixed reality scenarios (Sec. 8).

4 SENSOR BLUEPRINT – HARDWARE

Commercially successful commodity depth sensors based on *active* illumination mostly fall into these categories: Time-of-Flight (TOF), Temporal Structured Light (TSL), Spatial Structured Light (SSL), and Active Stereo (AS); see Appendix A for a detailed discussion. Due to temporal integration, TOF and TSL cameras either require *extremely* high (2000fps) framerate projector/cameras, or they result in systematic artifacts which researchers have recently been attempting to resolve [Gupta et al. 2015; O'Toole et al. 2014]. SSL sensors can theoretically provide high-frame rates, but they severely interfere with each other, hence limiting their applicability. AS sensors do not suffer any of these shortcomings, and recent work has substantially reduced their computational burden [Fanello et al. 2017b]. For these reasons, we selected Active Stereo as our sensor architecture.

Camera choice → OnSemi Python 1300. The ability to successfully track a fast moving object is heavily influenced by the resulting motion displacement between image frames as well as induced motion blur. Large displacements and motion blur make tracking more unstable, and in some cases intractable. The amount of displacement and motion blur can be controlled by the camera framerate and exposure time, respectively. Ideally, one would desire having both *high frame-rates* and *low exposure times*, but these values are constrained by capabilities of commodity cameras and illuminators. Thus, our camera module should be a (1) commercially available off-the-shelf camera, (2) be easily interfaced with, (3) be capable of achieving high frame rates, and (4) be a high resolution global shutter sensor with good quantum efficiency in the infrared (IR) spectrum.

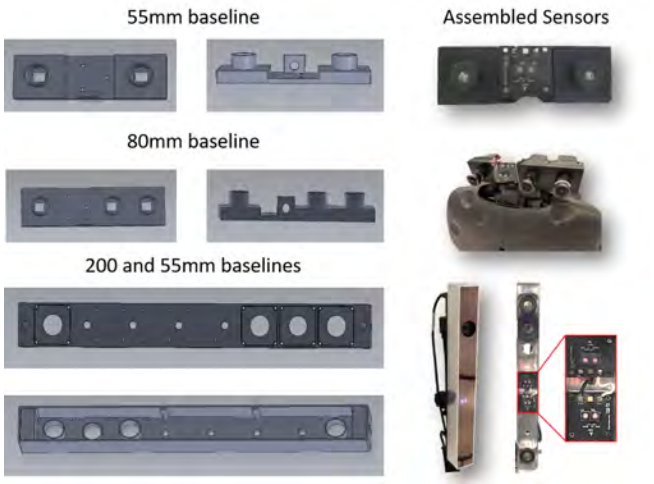


Fig. 3. (left) In the **additional materials** we provide CAD fabrication models for our active stereo camera systems at different baselines. (right) The 55mm baseline for exo-centric short range interaction (depth only), 80mm baseline for ego-centric VR headsets (depth+RGB) and hybrid 200mm/55mm for long range (full body, depth+RGB) capture.

In order to determine a sufficiently high frame rate and exposure time, we work backwards by considering the extremely high speed motions that occur in full body sports scenarios and from rapid hand movement. In particular, we consider a punch from a boxer as well as the snap of a finger. These motion trajectories correspond roughly to a velocity of 12 m/s. We consider a camera with a field of view of 65 degrees, SXGA resolution that captures at 30fps with an exposure time of 33ms – typical values in commercial sensors like the Kinect v1. When the subject is 1m away from the camera, the induced motion blur and screen-space displacement is about 500 pixels, with 500 pixels of motion blur between the two consecutive frames, ultimately making frame-to-frame tracking very difficult; see Figure 2. Taking these aspects into consideration, we considered the spectrum of widely available off-the-shelf high speed cameras and selected the *OnSemi Python1300* packaged as a USB3 module by Ximea. This infrared sensor is capable of achieving a framerate of 210fps at a spatial resolution of 1280×1024 with a 1ms exposure time[1]. In the previously detailed scenario, this results in an 80 pixels displacement and only 16 pixels of motion blur; see Figure 2 and Section 7. While the Ximea scientific camera module is relatively expensive at around $\approx$ \$800, we could also leverage mobile sensors such as the Sony IMX. This sensor has a better SNR/Quantum efficiency and framerate than the Python1300, and a cost of only $\approx$ \$3 (excluding lenses). However, there are still no devkits available for these cameras, therefore the final users are required to write their own custom drivers and build appropriate data transfer interfaces. We note that the choice of the sensor in this work is purely a function of the availability of off-the-shelf hardware. When designing for an actual product these would have to be considered

[1]Note that our current software stack and algorithms support future releases of the sensors up to 1000 fps at SXGA resolution.

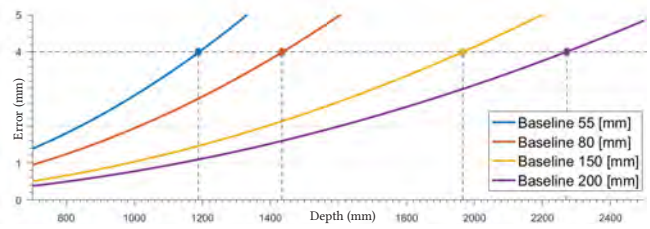
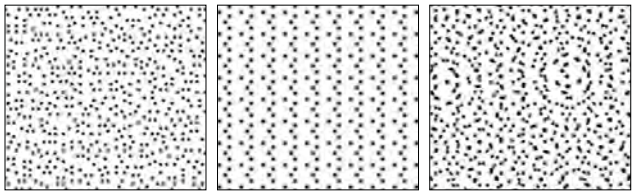


Fig. 4. We target an average expected error of 4mm to select our baselines (constrained by cameras form factors). Our hybrid 50mm/200mm capture pod delivers an excellent tradeoff of range vs. precision.



Microsoft Kinect v1 Single VCSEL Twin VCSEL (ours)

Fig. 5. Illumination patterns used by active stereo and structured light system. The Kinect uses a DOE pseudo-random dot pattern, while we use a VCSEL. This can result in spatial repetitions in the pattern, which we avoid by employing two illuminators slightly rotated with respect to each other.

ground up. For instance, we were bound by the USB bandwidth for the data transfer, but there are sensors that are available such as the Sony IMX range of sensors that are capable of upwards of 250fps via a MIPI interface that would be more amenable when designing such a depth sensor from ground up. Note that such sensors are already mass market, and are available on most recent mobile devices to allow for slow-motion video capture. One of the most recent sensors on the Samsung Galaxy S9 range of mobile devices boasts of a ultra high speed 12 Megapixel sensor capable of 720p at 960fps exemplifying the availability of high speed sensors on mobile platforms.

Stereo module → baseline and resolution. We mount two Python 1300 sensors in a rigid machined aluminum housing; see Figure 3-(left). The depth precision of a stereo system is governed by the baseline, camera resolution and the focal length. Due to our sensor choice we fixed the maximum resolution at 1280×1024 pixels and focus on optimizing the baseline; see Figure 4. In Section 4.3 we detail the choices of lenses that in turn affect the system resolution. In stereo systems, a larger baseline delivers more precise measurements but results in larger occlusions and foreshortening, which makes the matching process harder for objects close to the camera. Therefore, depending on the tracking scenario we use different camera baselines. Our specific choices of the baselines were driven by the sizes of the modules. In the case of a desktop scenario, for applications such as hand tracking, we chose a baseline similar to products such as Leap Motion while ensuring that we can pack the cameras in a tight configuration, which gave us a baseline of 55mm. In the case of a head-mounted scenario where we would want to capture close interactions such as the hands of the person wearing the headset while being able to reconstruct the room the user is in, we designed the baseline to closely match those of room scale stereo systems, such as Kinect and StructureIO, while keeping the cameras as close to each other as possible, resulting in a baseline of 80mm. Lastly, we built a 200mm baseline pod that allows us to capture the full-body of a person in the center of a multi-sensor rig; see Fig. 3. This baseline allows us to reduce the depth error further away from the camera.

Illuminator → Infrared VCSEL. Effective stereo matching requires the scene to be textured with a *locally unique* pattern. One choice for the illuminator is a low power single mode laser coupled with a *Diffractive Optical Element* (DOE) and replicator stack that can generate a pseudo random pattern. This approach is used in commercial depth sensors such as Microsoft Kinect V1, StructureIO, and in [Fanello et al. 2017b; Orts-Escolano et al. 2016]. However, for short exposure times (1 - 5 ms) such an illuminator does not offer enough SNR for low reflective surfaces and hence results in fairly sparse depth maps. Unfortunately, higher power lasers cannot be used together with a DOE due to eye safety reasons [OSHA 2017]. Therefore, we leverage the recent success in VCSEL-based IR illumination used in structured light projectors such as the Google Tango tablet and Mantis Vision scanners. A VCSEL emits the light from a much bigger surface than a DOE and therefore can project much more light while still being eye-safe. VCSELs have also been shown to be more efficient than lasers [D'Asaro et al. 2016] and can emit structured light when coupled with a suitable mask; e.g. see Google Tango and Mantis Vision. Conversely, we propose a simpler solution: we use a *pair* of VCSELs – each of them generates a regular grid of dots. One of the VCSELs is slightly rotated with respect to the other so that the combination of the two patterns results in the locally unique pattern shown in Figure 5. In order to maximize the SNR captured within the short exposure time we additionally *pulse* the illuminator in sync with the camera in such a way to reduce contamination from ambient light. By pulsing the illuminator we can reduce the exposure time of the camera to 2ms, as well as the energy consumption due to a reduced duty cycle.

4.1 Multi-sensor and multi-illuminator setups

Multiview tracking scenarios require multiple depth cameras to work together without interference. Our active stereo system is naturally robust as the only assumption that we make about the projected pattern is that it is locally unique. As the combination of two unique patterns results in another unique pattern, multiple depth cameras can actively illuminate the same surface without causing interference. However, two depth cameras that directly face each other can result in saturation in parts of the image due to *lens flare* – scattering of light from one illuminator in the lens of the other camera. In order to mitigate this effect, we pulse the VCSEL projectors. Recall that due to pulsing each illuminator and camera is only active for 2ms. If the cameras run at 210fps we will have a gap of 4.75ms between two consecutive frames. This means that we can temporally offset the exposures of cameras such that they do not interfere, while still maintaining the target frame rate of 210fps.

static pattern dynamic pattern

Fig. 6. (left) A constant structured light pattern results in visible depth bias for static scenes. (right) A temporally dynamic pattern *combined* with spatio-temporal matching windows corrects these artifacts.

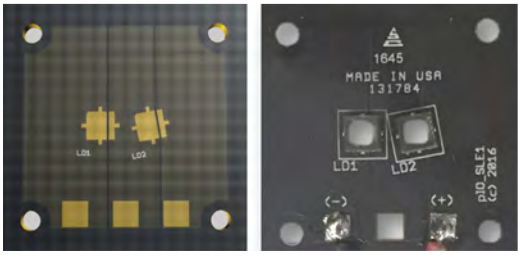


Fig. 7. (left) Underlying PCB. (right) VCSEL illuminator pair where two illuminator modules can be soldered in series and driven using one constant current LED driver.

4.2 Dynamic pattern – bias correction

Triangulation methods that make use of spatial active illumination suffer from bias due to the particular structure of the projected dot pattern; see Figure 6-(left). Even when we aggregate multiple frames over time, the structured nature of the noise generates errors that cannot be mitigated via simple averaging schemes. Butler et al. [2012] showed that bias and interference in standard structured light can be reduced through small motion of the illuminator. We implement a temporally varying pattern by using a four VCSEL setup, where each is mounted at a slightly different angle, and randomly flashing $\binom{4}{2}$ + $\binom{4}{3}$ combinations in succession; We implemented this solution in our 200mm baseline pod, which has enough space to accommodate four VCSEL illuminators; see Figure 3-(bottom right). The results of this process are showed in Figure 6 where one should notice the absence of artifacts on the body, and the high frequency details of the face. Also note that the reconstruction is more complete, especially in low SNR regions such as hair.

4.3 Hardware components

Lenses. The camera lens affects the field of view and hence the stereo system resolution. Sourcing lenses with exact characteristics for F number, focal length, field of view etc. is a cumbersome task due to limited availability of options. While the right approach is to build custom lenses designed for the specific scenario, building such custom lenses turns into a very expensive endeavor. In our depth camera, all the options we built have an M12 or an S-mount lens holder. We therefore resort to off-the-shelf M12 lenses from companies such as Megapixel Lenses, Uxcell and Lensation. We provide three different solutions, that in our experience, cover most of the high-level computer vision and HCI applications we developed. For short-range scenarios we found that a 60 deg field of view lens is satisfactory. This includes applications like gesture recognition in front of a laptop and object scanning. We used a 6mm lens from Megapixel Lenses for this scenario (model number 130620MP). In ego-centric scenarios, such as in VR applications, wide field of view lenses are preferred, therefore we selected a 3.6mm Uxcell lens (model number US-SA-AJD-69585) that can cover up to a 90 deg field of view. Finally for full body capture applications [Dou et al. 2017] we use a wide field of view (80 deg) lens with low distortion which provides high quality capture results. For this case we used a Lensation lens (model number BSM6016S12).

Illuminators. The choice of the type of illuminator used for the active stereo sensor is also an important one that defines the quality of depth obtained. There are a number of options when deciding the illuminator to use. Our initial experiments were using a commonly used illuminator that involves a DOE and an edge emitting laser (as in Kinect v1, StructureIO). However, our observation was that at very low exposure times, as required by a high speed depth camera the SNR of such illuminators was very poor. In addition, these illuminators cannot be driven at high currents as this takes it away from an eye-safe IR margin. In our case we use a VCSEL based illuminator by Heptagon called Lima. This consists of an array of VCSELs operating at a wavelength of 850nm and a microlens array above it that produces a regular grid of dots as shown in Figure 5. In contrast to the DOE based solutions (Kinect v1 and StructureIO) the dots produced by the microlens array has a much larger pixel footprint, specifically the spot size from the Lima is about 2 to 3 times wider than that of a Kinect v1; see Figure 5. The larger spot size however provides more signal for matching in the active stereo setup. The field of illumination of the Lima illuminator is around 85 degree diagonal that closely matches the field of view of the lenses. As discussed, we reduce the ambiguity in stereo matches by using two Lima modules rotated with respect to each other. These are soldered onto PCB shown in Figure 7 such that the illuminators are connected in series. The illuminator pair is then driven by a 1000mA LuxDrive BuckBlock, which is a constant current LED driver with a dimmer that allows us to pulse the illuminator synchronized with the camera exposure. We note that more recently illuminators that use a VCSEL along with a DOE have been proposed, such as the Apple iPhoneX, however these are not available off-the-shelf. These could, however, be viable alternatives. At high frame rates, thermal and power considerations are important. In all our experiments, we power the illuminator in sync with the short exposure time of the camera, thus ensuring that the duty cycle is low and the power consumption is kept in check. In practice, the scenario the sensor is being applied to, defines the form-factor and power limitations of the system. For instance, a room scale sensor could have a larger form factor and incorporate large heat sinks if the system needs to be driven at a high power, whereas for a mobile form-factor we have seen a viable solution evident in the iPhoneX.

IR Filters. Given the lens and the illuminators above, the last part for the IR optical path is the filter. We need a filter that has

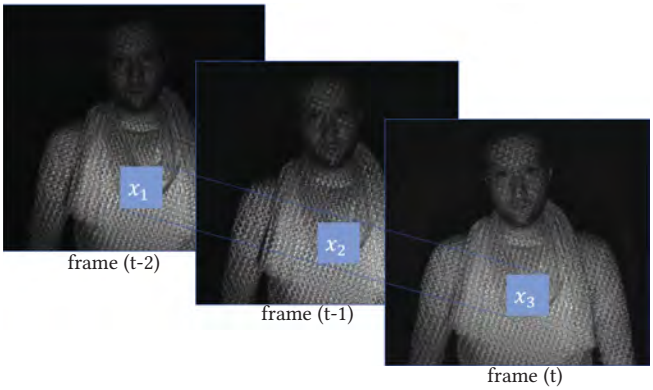


Fig. 8. Each pixel in a sequence of IR stereo images, a binary descriptor is computed from a spatio-temporal neighborhood $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3]$.

a narrow bandwidth centered around the primary wavelength to ensure we suppress as much ambient light as possible, ensuring a high SNR at the specific IR wavelength. In this case, we employed IR filters with a thickness of around 1mm centered at 850nm with a bandwidth of around 15nm manufactured by Omega Filters. These were purchased as square filters cut to match the hole at the back of our CAD models, and were glued onto the back such that they lie between the lens and the sensor.

5 SENSOR BLUEPRINT – SOFTWARE

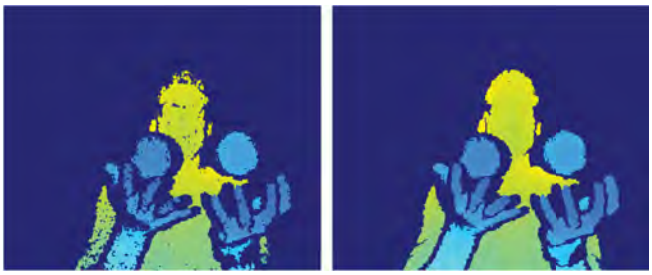
We now describe the software pipeline for the active stereo depth sensor described in the previous section. As detailed in Scharstein and Szeliski [2002], a general stereo matching pipeline comprises three main components:

- *Matching cost – Section 5.1:* defines the distance or similarity between two image patches.
- *Disparity optimization – Section 5.2:* searches corresponding patches in the two images minimizing the matching costs.
- *Disparity refinement – Section 5.3:* postprocess disparities in order to achieve subpixel precision and reject outliers.

We contribute to each stage of the stereo pipeline. First, we design a space-time matching solution that leverages the high framerate stream. Then, we exploit the temporal coherency in the disparity optimization stage. Finally, we we address outlier rejection problem using a computationally efficient machine learning model.

5.1 Matching cost computation

Given an image patch $\mathbf{x}_L$ in the left image and an image patch $\mathbf{x}_R$ in the right image, we want to compute a matching score. The patch size needs to be big enough to uniquely identify a pixel based on the texture in its surrounding area. Typical patch sizes for active stereo are 11×11 windows for 1.3 Megapixel images. Traditional stereo correlation functions compute differences between *every* pixel of the two patches, and produce a score that is used for disparity optimization. The computational cost of these methods increases with the patch size and therefore are not well suited if many disparity hypotheses need to be evaluated; see Section 5.2.



spatial matching spatio-temporal matching

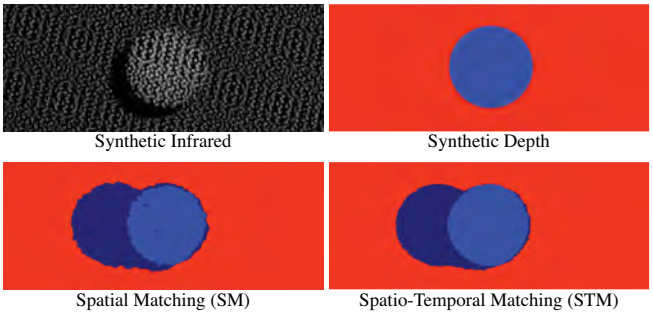
Fig. 9. Using only spatial information for matching results in missing data. Using temporal information gives more complete results because more information is embedded in the spatio-temporal domain.

Learnt matching space. Applying Fanello et al. [2017a] to our problem, each spatio-temporal image patch $\mathbf{x}$ is converted to a 32-dimensional binary code as $\mathbf{b} = sign(\mathbf{xW})$, where $\mathbf{W} \in \mathrm{R}^{n \times k}$ has been learnt so that every column contains at most $b = 4$ non-zeros. We then define the cost function between two image patches $\mathbf{x}^L$ and $\mathbf{x}^R$ as the Hamming distance between the codes $\mathbf{b}^L$ and $\mathbf{b}^R$. This results in a complexity that is independent from n, the number of pixels in a patch/window.

Exploiting temporal coherency. In high speed sensors we can safely assume that the motion between subsequent frames is very small, hence we can extend the approach above to mapping of spatio-temporal patches. As the shape remains roughly constant over small periods of time, we can use a straight spatio-temporal image volume $\mathbf{x}$ with dimensions $n = P \times P \times F$, where P is the spatial window size and F is the number of frames in the temporal buffer; see Figure 8. Note how, since $\mathbf{W}$ is learnt to be b-sparse, our spatio-temporal mapping is also independent of F. In order to take advantage of temporal windows, the appearance of the patch needs to change over time to be able to ensure the information added across multiple frames is not redundant. We achieve this by dynamically changing the projected pattern over time as detailed in Section 4.2. The benefit of matching with a spatio-temporal window is threefold: (1) It reduces noise in the matching; see Figure 9 and Figure 10. (2) It allows for smaller spatial windows that ensure reduced edge fattening, resulting in better performance along depth discontinuities; see the edges of the sphere in Figure 10. (3) It removes bias artifacts caused by active stereo matching (for static objects); see Figure 6.

5.2 Disparity optimization

Disparity optimization is the most expensive part of the stereo matching pipeline. Given the mapping of each pixel $\mathbf{p}_i$ in both images to a binary code $\mathbf{b}_i$, our optimization aims at finding the best disparity assignment for each pixel without evaluating all possible disparity labels d (respectively 256 or 512 labels for short and wide baselines in our implementation). While many recent works [Fanello et al. 2017b; Orts-Escolano et al. 2016; Pradeep et al. 2013] optimize for disparity via a PatchMatch search [Bleyer et al. 2011], in this paper we investigate the applicability of the more computationally efficient parallel inference technique presented in [Fanello et al. 2017a]. These changes allowed us to achieve a *two-fold* increase in the performance of disparity optimization from 500fps to 1000fps.



	SM	STM 0-5	STM 6-20	STM 21-40	STM 41-60
Error (mm)	1.08	**0.76**	1.35	1.48	1.8
Valid (%)	71	**89**	78	72	67

Fig. 10. Quantitative evaluation on synthetic data. We move a sphere 3.5m away from the camera horizontally in front of a wall (1.5m distance). We synthesize two IR views using our projected pattern, as well as ground truth disparity in Blender. With small displacements STM performs favorably to SM, while the error does not degrade significantly with larger displacements. Most importantly, note how the number of valid pixels is much larger than for SM, and this holds even at very large displacements (21-40 pixels range). Assuming an objects at 1.5 distance from the camera, a 20 pixel displacement at 210 fps means a speed of 5m per second. For the considered applications (i.e. performance capture and tracking of humans) this motion is very large.

We initialize the image by testing 32 random disparities for each pixel, and then selecting the one with the smallest Hamming distance in the binary space. Each pixel p_i has now associated a certain disparity d_i. To perform the actual optimization, we now test all disparity labels in a 3×3 neighborhood $\mathcal{N}_p$ of the pixel p_i and select the one with the best cost. The cost optimization problem is defined as:

$$\arg\min_{d\in\mathcal{N}_p} \underbrace{|\mathbf{b}_p^L - \mathbf{b}_{p+d}^R|}_{\text{distance}} + \sum_{d_k\in\mathcal{N}_p} \underbrace{\max(\tau, |d_k - d|)}_{\text{smoothness}} \qquad (1)$$

where the first term is the Hamming distance between the codes at the pixel p in the left image and the codes computed at the location $p+d$ in the right image. Note here that we use a simplified notation referring only to the pixels on the same scan-line. Therefore a pixel p is defined only by its x component and $p+d$ is a shift along that dimension. The second term enforces smoothness among neighboring pixels. Since we are considering $d \in \mathcal{N}_p$ potential solutions, we can solve Eq. 1 efficiently on the GPU by enumerating all the elements as $|\mathcal{N}_p| = 3\times3$ and select the best one. To allow for disparities to get propagated in large areas of the image, we run multiple iterations to solve Eq. 1 in the 3×3 neighborhood. We empirically noticed that 4 iterations are enough to reach convergence.

Exploiting temporal coherency. We also extend this optimization scheme to exploit high frame rate data in the initialization step. In particular, for each pixel p at time t, we test its previous disparity at time $t-1$. If the Hamming distance is lower than all the 32 random disparities we keep the previous values and use that to initialize the iterative optimization. We find that typically, given a 210fps sensor, most of the pixels will have the same disparity between two consecutive frames, and thus this considerably increases accuracy.

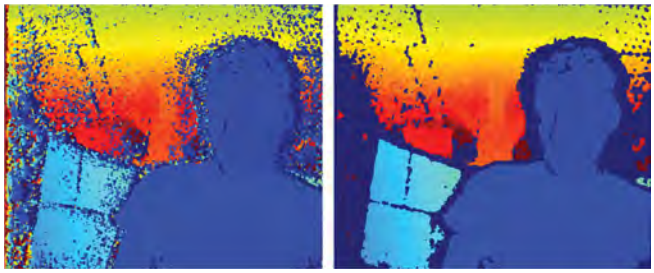


Fig. 11. We remove false positives from the disparity map using a data driven approach. Given a pixel in the disparity map, a decision tree sparsely samples its spatial neighborhood to predict whether the pixel should be invalidated. The overall running time is $100\mu s$ per frame. Notice also that the temporal initialization scheme proposed in Section 5.1 automatically corrects some of the wrong estimates in the disparity maps. As consequence, the rejected pixels have some chances of being correctly recovered in the next frame and the disparity map appears more complete.

Our results are visualized in Figure 9, notice how our approach has on average 15% more valid pixels than the standard matching scheme without temporal information (see also Figure 10).

5.3 Disparity refinement

Subpixel refinement. To achieve subpixel precision, we use a standard parabola interpolation. Given a pixel p with a disparity d we fit a parabola by considering the disparities $d-1$ and $d+1$. We compute the Hamming distances of the binary codes for the disparities d, $d-1$ and $d+1$ and fit a quadratic function. The best disparity lies at the global minimum of this function, therefore we pick this as optimal value. We couple this step in the optimization scheme and repeat the fitting at the end of each iteration and for every pixel.

Outlier rejection. The traditional approach in stereo vision community is to apply a cross-check to detect and remove outliers [Scharstein and Szeliski 2002] followed by a weighted median filtering using an RGB image. This process requires the computation of two disparity maps for each frame, which is prohibitively expensive. In this paper, we propose a novel method that learns the invalidation function directly from data. We collected training data by recording about 10000 disparity maps of arbitrary indoor scenes, and 5000 images for testing. We generated ground-truth data by running traditional outlier rejection methods such as left-right disparities cross-check, as well as a weighted median. To compute the weighted median we mount an RGB sensor next to our active stereo sensor that we synchronized and calibrated to our system (the RGB sensor is only needed at training time and not at runtime). For training we mark each pixel as either *valid* or *invalid*, depending on the result of the cross-check and median filter. Given this ground truth, we learn a function that decides to either invalidate or accept a given disparity. In order to keep the computation low and independent of image resolution, we perform this outlier rejection using a decision forest [Shotton et al. 2013]. A node in our decision tree contains two learned pixel offsets $\mathbf{u} = (\Delta x, \Delta y)$ and $\mathbf{v} = (\Delta x', \Delta y')$ and a threshold value τ. When evaluating a pixel at position $\mathbf{p} = (x, y)$, the tree decides where to route a particular example based on the

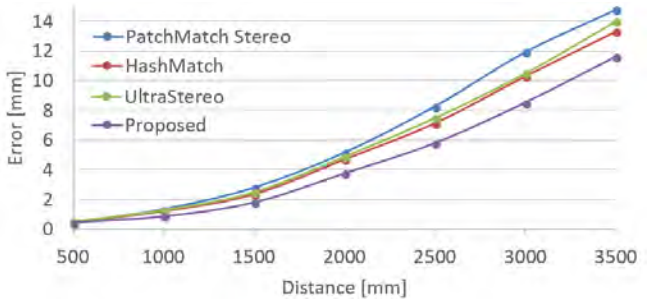


Fig. 12. Depth precision of proposed hardware versus competing methods. We compare our method against PatchMatch Stereo [Bleyer et al. 2011], HashMatch [Fanello et al. 2017a] and UltraStereo [Fanello et al. 2017b].

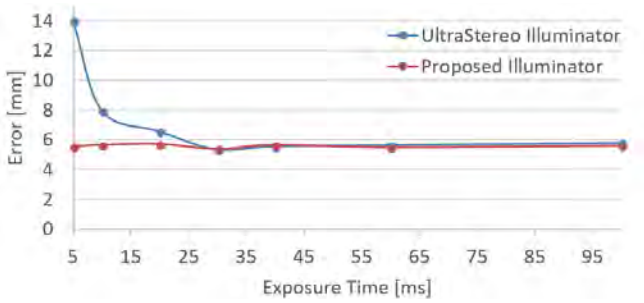


Fig. 14. We compare different illuminators (UltraStereo [Fanello et al. 2017b] vs. the one we propose) while keeping the depth algorithm fixed (our algorithm). Notice how for very low exposure time (i.e. low SNR) our solution outperforms DOE illuminators.

sign of $I(\mathbf{p}+\mathbf{u}) - I(\mathbf{p}+\mathbf{v}) > \tau$, where $I(\mathbf{p})$ is the intensity value of a pixel $\mathbf{p}$. At training time we randomly sample 500 possible split parameters $\delta = (\mathbf{u}, \mathbf{v}, \tau)$ for the current node. Each δ induces a split on the set S of the data into left $S_L(\delta)$ and right $S_R(\delta)$ child sets. We select the set of parameters δ that maximizes the *Information Gain*:

$$IG(\delta) = E(S) - \sum_{d \in L,R} \frac{|S_d(\delta)|}{|S|} E(S_d(\delta)) \tag{2}$$

where $E(S)$ is the Shannon entropy of the empirical distribution $p(valid|S)$ of the class label "valid" in S. Each leaf node contains a probability $p(valid|\mathbf{p}, I)$ and we invalidate pixels when this quantity is less than 0.5. In Figure 11 we show an example of our invalidation strategy based on a decision tree. A single tree with 12 level is enough to reach 98.25% accuracy with a runtime speed of 100μs per frame; see Figure 11 for an example. In comparison, a traditional min-region check executes in 1ms.

6 COMPARISON WITH THE STATE-OF-THE-ART

Precision of depth estimation – Figure 12 and Figure 13. We first evaluate the precision of the depth algorithm. We use the proposed hardware and collect multiple images of a flat wall at various distances ranging from .5m to 3.5m. We robustly compute ground truth depth by fitting a plane to the depth data and calculate the error averaging across multiple frames. In Figure 12, we report the results for various competitors – notice how the proposed algorithm achieves the lowest error. Additionally, to better highlight the improvements of our system, we show qualitative results in Figure 13. Here we compare the method with [Fanello et al. 2017b], as well as with a modified version where we exploit the initialization scheme described in Sec. 5.2: in this way we can separately evaluate the contribution of the temporal initialization and the spatial-temporal matching scheme we propose. The proposed solution produces smoother disparity maps, with less holes in low SNR areas like hair and slanted surfaces. Further, the computational complexity of our method is *half* the one proposed in [Fanello et al. 2017b].

Fig. 13. Qualitative comparisons with UltraStereo [Fanello et al. 2017b], and a variant of this method including the initialization from Sec. 5.2. Notice how the proposed space-time stereo provides more complete disparity maps in low SNR regions (e.g. hair) as well as in slanted regions.

Impact of different illuminators – Figure 14. In our second experiment, we evaluate the benefits of the proposed pattern/illuminator, compared to the one used in [Fanello et al. 2017b]. In particular, we recorded a flat wall at 2.5m varying the exposure time of the camera from 5ms to 100ms. We recorded data using the proposed illuminator, as well as the one used in [Fanello et al. 2017b]. For both the illuminator we use the same amount of power ($\approx$ 220mW) and we fix the depth algorithm to be the one proposed in this paper. To assess the impact of the illuminators, we compute the average error on a single frame at different exposure time. Notice how our pattern exhibits very low error at 5ms exposure time, whereas the DOE used in [Fanello et al. 2017b] reaches a similar behavior at 30ms.

7 APPLICATIONS TO REAL-TIME TRACKING

We now consider multiple tracking applications as to qualitatively and quantitatively evaluate our system. In particular, we show the effectiveness of high framerate streams for all of these applications. In a general tracking problem, we assume a camera sensor providing a stream of depth images $\{\mathcal{D}_t\}$. The goal of a tracking method is to find the pose parameters $\theta_t \in \mathbb{R}^d$ that describe the pose of the object of interest at the time t according to the current data $\mathcal{D}_t$. Often such problems are cast as an energy minimization problem:

$$\arg\min_{\theta} \quad E_{data}(\theta) + \lambda_R E_{reg}(\theta) \tag{3}$$

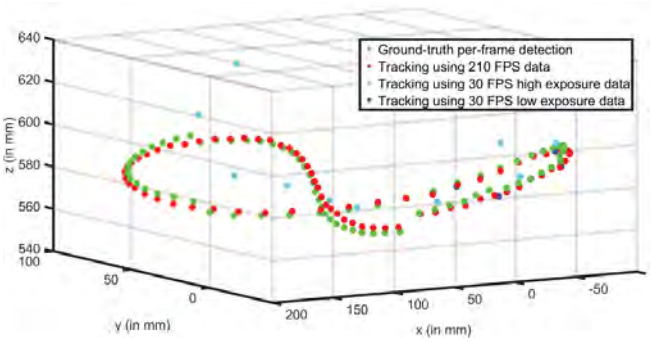
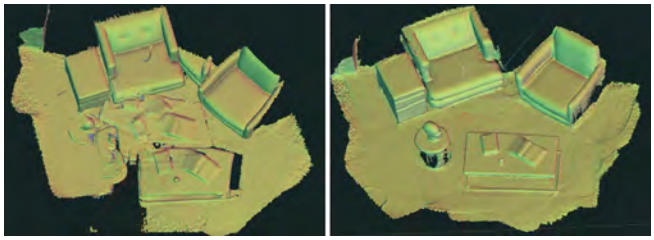


Fig. 15. We track a tennis ball over time and plot the tracked positions over time; see Figure 2. Tracking with high framerate data is closer to the GT than using low framerate data.

The data term $E_{data}(\theta)$ describes how well the solution describes the current observations. The regularization term $E_{reg}(\theta)$ helps to choose a sensible solution in under-constrained problems. The form of these two energy terms strongly depends on the considered application. A common approach is to define smooth energy functions suitable for gradient based local optimizers such as Levenberg-Marquardt (LM). In these optimization schemes, the initialization of θ plays a crucial role and typically the pose θ_{t-1} in the previous frame is employed as an initialization. However, when the object of interest moves at high speed, the pose θ_t at time t can deviate largely from that of the previous frame making it increasingly likely that this initialization is not in the basin of convergence of a good local minimum (*i.e.* a local minimum that corresponds to the true pose). State-of-the-art systems try to cope with this by either designing complex reinitialization systems [Taylor et al. 2016], or by computing additional correspondences via SIFT features [Innmann et al. 2016], learnt hash functions [Dou et al. 2016] or spectral embeddings [Dou et al. 2017]. Conversely, we provide a capture technology capable of producing a real-time 210fps stream of high quality depth images. In this setting interframe motion is drastically reduced, hence increasing the chance that the optimizer starts in the basin of convergence of a good local minimum; *i.e.* that tracking succeeds. The capabilities of our high-framerate sensor, and its advantages across a number of applicative domains, can be better appreciated in our **supplemental video**.

Tracking evaluation design. We recorded very challenging sequences where subjects or objects were moving with high speed. The raw data consists of 210fps IR images and RGB (used only for visualization purposes) recorded with 2*ms* exposure time. We then subsampled the raw data to generate 30fps data with the same exposure time, and finally we simulated a standard 30fps with 33ms exposure time by averaging multiple frames; this approximation holds when the SNR is sufficiently high, which is true with our pulsed illumination. We tuned our depth algorithm to achieve the best results in each setting. We also tried a GPU version of PatchMatch Stereo [Bleyer et al. 2011], running at 60Hz, but we did not notice substantial changes in the depth quality. Indeed, in active stereo, local methods have proved to achieve high accuracy at par with more expensive global methods. In the following, we process all



30fps 210fps

Fig. 16. A fast moving camera scanning the room: notice how at 30fps the tracker loses its position and duplicate surfaces appear.

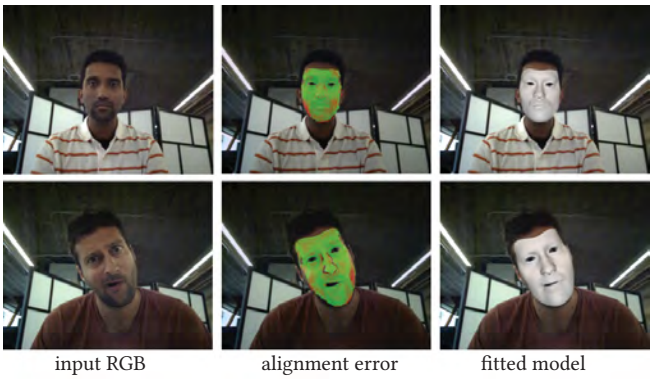
three data streams ("210fps - low exposure", "30fps - low exposure" and "30fps - high exposure") in the various tracking pipelines with exactly the *same* computational budget. This means that we can only run 1 iteration of Levenberg-Marquardt for the 210 fps data, whereas for the 30 fps data we can run 7 iterations. In the following, we show that high framerate data is needed to achieve convergence of the algorithm, and that with standard 30fps data we cannot infer the correct solution even though many more iterations of the solver are used.

7.1 Parametric tracking – [Nakabo et al. 2000] – Fig. 15

In parametric tracking, the model of the tracked geometry is known and the six dimensional transformation of the model is to be inferred from the input data. We recorded a 2000 frame long sequence of a fast moving ball as shown in Fig. 2-(right). We initialize the tracking for the first frame using the RGB information and then use the known geometry of the sphere to track its motion over time. We

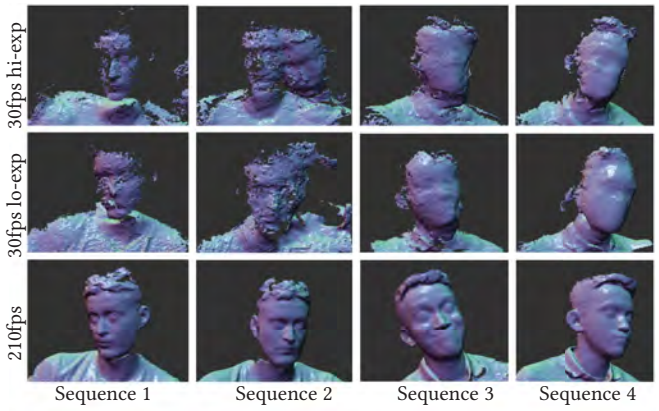
Type of data	30fps high exp	30fps low exp	210fps
Fitting error (in mm)	2.89 ± 0.63	2.76 ± 0.65	**2.37 ± 0.46**

Fig. 17. Each column shows a raw IR image on top, and the tracked hand rendered on top of the depth map on the bottom. While the changes in average fitting error of 3000 frames might seem small, tracking at a high framerate results in remarkably more robust hand tracking.



Type of data	30fps high exp	30fps low exp	210fps
% frames < 3mm error	69.5%	70.1%	**74.5%**

Fig. 18. We color-code per-pixel error in the range [0, 10]mm from green to red. Our quantitative analysis shows how using a higher framerate performs better although less compute is spent.



Type of data	30fps high exp	30fps low exp	210fps
Fitting error	$2.7 * 10^5$	$2.1 * 10^5$	$\mathbf{1.3 * 10^4}$

Fig. 19. In each column we report the non-rigid reconstruction at the last frame of different sequences (≈ 3000 frames). Notice how the 210fps data achieves an *order of magnitude* lower fitting error than the 30fps data.

use a combination of color and geometry information to generate reliable ground truth for this sequence. In Figure 15, we plot the ground truth data along with results on the three test data streams. When using the "30fps - high exposure" we have significant amount of motion blur; see Figure 2. Interestingly, the motion blur actually helps the tracker to not fail, but the tracking error is much higher. When "30fps - low exposure" data is used, the tracker fails to follow the ball after few frames due to the large displacement in the image. For the "210fps" data (red dots) we are able to recover the full motion.

7.2 Rigid fusion – [Newcombe et al. 2011] – Fig. 16

In rigid fusion our target is to determine the camera position within the scene, and then fuse the depth data into a non-parametric reference surface representation. To compare high and low framerate data for rigid tracking we recorded a sequence of a fast moving camera in a standard indoor scene and we reimplemented KinectFusion [Newcombe et al. 2011] to perform a rigid reconstruction of the environment. We show qualitative results in Fig. 16: for both "30fps - low exposure" and "30fps - high exposure" data the tracking is lost and the algorithm creates duplicate surfaces in the scene. Conversely, for the high speed data the scene is reconstructed correctly with no substantial errors.

7.3 Hand tracking – [Taylor et al. 2017] – Fig. 17

In articulated hand tracking our task is to estimating the pose parameters of a human hand. We use [Taylor et al. 2017], specifically designed to run with high frame rate depth streams, to track the hand pose, and report the fitting error in Figure 17-(bottom). The high frame rate 210fps data achieves lower fitting error across all the data in comparison to the 30 fps data. Note that while this average error might seem small, the differences in regressed poses are *dramatic* as illustrated by the failure cases in Fig. 17-(top).

7.4 Face tracking – [Thies et al. 2016] – Fig. 18

In our facial tracking test, we regress the rigid transformation as well as shape and pose parameters encoded in a low-dimensional blendshape representation [Blanz and Vetter 1999]. We recorded sequences of subjects performing arbitrary facial expressions, and quantitatively evaluated the data fitting error; see Fig. 18. Notice that although we use 7 iterations for both "30fps - low exposure" and "30fps - high exposure" data, the retrieved solution is not able to match the precision of the high speed stream.

7.5 Non-rigid fusion – [Dou et al. 2017] – Fig. 19

In non-rigid fusion our task is to progressively reconstruct a non-parametric model of the object being observed [Newcombe et al. 2011], while simultaneously determining the parameters of a non-rigid deformation model [Sumner et al. 2007]. In Figure 19, we use our reimplementation of [Dou et al. 2017] and show how fast motions *cannot* be handled at low framerate. Notice how the tracking error completely breaks the reconstruction or generates additional surfaces, a phenomenon that is clearly measured by an *order of magnitude* lower error when executing on data acquired at 210fps. Similarly to face tracking, we only need 1× LM iteration at 210fps, while 7× iterations are necessary at 30fps.

8 LOW LATENCY VIRTUAL REALITY

Depth information plays a crucial role in perceiving the world in virtual reality. The use of traditional RGB passthrough techniques would lead to distorted depth perception [Rolland et al. 1995; Takgi et al. 2000]. To enable precise 3D information of the real world while wearing a VR device, we use our camera in a first person view configuration by attaching the sensor to a 6-DoF tracked VR headset (HTC Vive), see Figure 3, second row. The camera and the VR environment coordinate systems are aligned using the hand-eye calibration method described in [Tsai and Lenz 1988]. We back-project the colored depth map into the 3D environment, essentially

providing the user with a mixed reality environment in which real objects in front of the user coexist in the artificial environment, similar to Intel Project Alloy. The user benefits from being able to interact both with real and virtual objects and can confidently walk around in the physical space.

Latency and out-of-body feeling. High frame rate sensing is one of many crucial aspects in reducing the lag between what is happening in the real world and what is conveyed into the virtual environment. High latency causes difficulties coordinating human-to-human interactions, such as passing, tossing and catching objects [Pan and Niemeyer 2017], shaking hands or performing a high five. Even when the user is only seeing their own body, high latency can cause spatial discrepancy between where the user feels their hand is (e.g. through proprioception and haptic feedback) and where it is being rendered. This can make hand-eye-coordination challenging and cause unintentional out-of-body experiences.

Hand-Eye coordination. Most of the experiences in VR, involve a certain degree of interaction with the virtual world. To enable the same interactivity with the real-world it is crucial to employ a depth sensor with very low latency. Although a straightforward solution consists of employing an off-the-shelf depth sensor, these cameras usually carry an unacceptable latency for the user, which can fail simple eye-hand coordination tasks such as catching a real ball while wearing a VR headset. To this end, we conducted an informal study evaluating users' performance at throwing and catching a ball from a distance of about 150cm. We observed a significant drop in catching performance at around 66.66 ms latency, which is the performance of a typical consumer depth camera [Webster and Celik 2014]. Using our capture system, on the other hand, enables these interactive scenarios with the real world; see **supplemental video**.

Latency in our system. We finally assess the overall latency of the proposed 3D sensor. There are many steps from capturing to displaying a depth map, each adding latency. Starting with the inherent delay caused by the exposure time, camera frame rate, data transfer, processing, rendering and display refresh rate. In order to keep the continuous capture of the physical world aligned with the virtual environment, we apply headset transformations from the past (the point at which the capture presumably happened) to back-project depth pixels into 3D space. We measured the total latency between a physical target's motion and the rendering on the headset with two synchronized high speed cameras capturing at 240fps (see Fig. 20). One camera captured the target attached to an HTC Vive controller. The other camera captured the output of the headset display. Our depth camera ran at 180fps, while the render loop ran at 100fps and the HTC Vive at 90fps. This results in a theoretical minimum latency of 26.66ms. We performed a frame-by-frame analysis of the motion trajectory of the target and the Vive controller around the point of lift from a surface (start of motion) and around the point of impact (abrupt stop of motion). We measured the total motion to photon latency of the target to be around 8 frames in a 240fps recording (i.e. 33ms). The VR controller utilizes a combination of absolute optical pose tracking and IM-based pose prediction. In our analysis of the controller's motion trajectory, we observed a motion delay of around 5 frames or 20.83ms with overshooting behavior and around 11 frames or 45.83ms delay at the point of absolute pose correction. Our depth capture system's latency is exactly at the midpoint of the VR controller's pose prediction and absolute pose correction delays and is only slightly above the theoretical minimum latency.

Fig. 20. Frame-by-frame motion trajectory and latency analysis. *Left: direct slow motion capture of a HTC Vive controller with a physical target. Right: slow motion capture of depth rendering through the HTC Vive VR headset.*

9 DISCUSSION AND LIMITATIONS

We presented a real-time high speed 3D capture system for visual tracking problems. We designed the full stack from hardware choices to the software implementation. We proposed multiple practical and algorithmic contributions ensuring that each step of the pipeline can be easily reproduced. We showed how to take advantage of the high fps to improve state of the art active stereo algorithms, and evaluated the framework in multiple applications. Our results show the importance of high speed sensors for a multitude of computer vision problems: from simple model tracking to sophisticated articulated pose estimation tasks.

Although the proposed system yields substantial improvements in many tracking related applications, there are still many cases where even very high speed sensors cannot completely guarantee tracking failures are avoided:

- In the non-rigid reconstruction case described in Section 7.5, when a single depth camera is used, the presence of occlusions cause catastrophic tracking failures, and even state of the art methods [Guo et al. 2018] are very unlikely to recover.
- At short exposure times the system can still struggle to obtain accurate depth measurements on very dark surfaces (e.g. an object painted blackt).
- Due to the effective resolution of our pattern and camera, obtaining depth measurements on very thin structures at large distances remains a challenge.
- Highly reflective surfaces, such as mirrors, would likely lead to incorrect 3D reconstructions.

We believe that recent advances in deep learning for depth estimation [Khamis et al. 2018; Zhang et al. 2018] can address most of these issues.

REFERENCES

A. Bhandari, A. Kadambi, R. Whyte, C. Barsi, M. Feigin, A.A. Dorrington, and R. Raskar. 2014. Resolving Multi-path Interference in Time-of-Flight Imaging via Modulation Frequency Diversity and Sparse Regularization. *CoRR* (2014).

Volker Blanz and Thomas Vetter. 1999. A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 187–194.

M. Bleyer, C. Rhemann, and C. Rother. 2011. PatchMatch Stereo - Stereo Matching with Slanted Support Windows. In *BMVC*.

Michael Bleyer, Christoph Rhemann, and Carsten Rother. 2012. Extracting 3D Scene-consistent Object Proposals and Depth from Stereo Images. In *ECCV*.

D. Alex Butler, Shahram Izadi, Otmar Hilliges, David Molyneaux, Steve Hodges, and David Kim. 2012. Shake'N'Sense: Reducing Interference for Overlapping Structured Light Depth Cameras. In *CHI*.

Yang Chen and Gérard Medioni. 1992. Object modelling by registration of multiple range images. *Image and vision computing* 10, 3 (1992), 145–155.

C. Ciliberto, S. R. Fanello, L. Natale, and G. Metta. 2012. A heteroscedastic approach to independent motion detection for actuated visual sensors. In *IROS*.

L. Arthur D'Asaro, Jean-Francois Seurin, and James D. Wynn. 2016. The VCSEL Advantage: Increased Power, Efficiency Bring New Applications. (2016).

James Davis, Diego Nehab, Ravi Ramamoorthi, and Szymon Rusinkiewicz. 2005. Space-time Stereo: A Unifying Framework for Depth from Triangulation. *PAMI* (2005).

Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, , and Shahram Izadi. 2017. Motion2Fusion: Real-time Volumetric Performance Capture. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)* (2017).

Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. 2016. Fusion4D: Real-time Performance Capture of Challenging Scenes. (2016).

S. R. Fanello, I. Gori, G. Metta, and F. Odone. 2013a. Keep it simple and sparse: Real-time action recognition. *JMLR* (2013).

Sean Ryan Fanello, Ilaria Gori, Giorgio Metta, and Francesca Odone. 2013b. One-Shot Learning for Real-Time Action Recognition. In *IbPRIA*.

Sean Ryan Fanello, Ugo Pattacini, Ilaria Gori, Vadim Tikhanoff, Marco Randazzo, Alessandro Roncone, Francesca Odone, and Giorgio Metta. 2014. 3D Stereo Estimation and Fully Automated Learning of Eye-Hand Coordination in Humanoid Robots. In *IEEE-RAS International Conference on Humanoid Robots*.

Sean Ryan Fanello, Christoph Rhemann, Vladimir Tankovich, A Kowdle, S Orts Escolano, D Kim, and S Izadi. 2016. Hyperdepth: Learning depth from structured light without matching. *CVPR* (2016).

Sean Ryan Fanello, Julien Valentin, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, Carlo Ciliberto, Philip Davidson, and Shahram Izadi. 2017a. Low Compute and Fully Parallel Computer Vision with HashMatch. *Proc. of ICCV* (2017).

Sean Ryan Fanello, Julien Valentin, Christoph Rhemann, Adarsh Kowdle, Vladimir Tankovich, and Shahram Izadi. 2017b. UltraStereo: Efficient Learning-based Matching for Active Stereo Systems. *CVPR* (2017).

Christian Forster, Matia Pizzoli, and Davide Scaramuzza. 2014. SVO: Fast semi-direct monocular visual odometry. In *ICRA*.

D. Freedman, E. Krupka, Y. Smolin, I. Leichter, and M. Schmidt. 2014. SRA: Fast Removal of General Multipath for ToF Sensors. *ECCV* (2014).

Jason Geng. 2011. Structured-light 3D surface imaging: a tutorial. *Advances in Optics and Photonics* 3, 2 (2011), 128–160.

Yuanzheng Gong and Song Zhang. 2010. Ultrafast 3-D shape measurement with an off-the-shelf DLP projector. *Optics express* (2010).

I. Gori, U. Pattacini, V. Tikhanoff, and G. Metta. 2013. Ranking the Good Points: A Comprehensive Method for Humanoid Robots to Grasp Unknown Objects. In *IEEE ICAR*.

Kaiwen Guo, Jonathan Taylor, Sean Fanello, Andrea Tagliasacchi, Mingsong Dou, Philip Davidson, Adarsh Kowdle, and Shahram Izadi. 2018. TwinFusion: High Framerate Non-Rigid Fusion through Fast Correspondence Tracking. In *3DV*.

Mohit Gupta, Shree K Nayar, Matthias B Hullin, and Jaime Martin. 2015. Phasor imaging: A generalization of correlation-based time-of-flight imaging. *ACM TOG* (2015).

Ankur Handa. 2013. *Analysing high frame-rate camera tracking*. Ph.D. Dissertation. Imperial College London.

Ankur Handa, Richard A. Newcombe, Adrien Angeli, and Andrew J. Davison. 2012. Real-Time Camera Tracking: When is High Frame-rate Best?. In *ECCV*.

Roland Höfling, Petra Aswendt, Frank Leischnig, and Matthias Förster. 2015. Characteristics of digital micromirror projection for 3D shape measurement at extreme speed. In *SPIE OPTO*. International Society for Optics and Photonics.

Berthold KP Horn and Brian G Schunck. 1981. Determining optical flow. *Artificial intelligence* 17, 1-3 (1981), 185–203.

Asmaa Hosni, Christoph Rhemann, Michael Bleyer, Carsten Rother, and Margrit Gelautz. 2013. Fast Cost-Volume Filtering for Visual Correspondence and Beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 35, 2 (2013), 504 – 511.

Jae-Sang Hyun, Beiwen Li, and Song Zhang. 2017. High-speed high-accuracy three-dimensional shape measurement using digital binary defocusing method versus sinusoidal method. *Optical Engineering* (2017).

Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. 2016. VolumeDeform: Real-time volumetric non-rigid reconstruction. In *Proc. of ECCV*. 362–379.

S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. 2011. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera.

D. Jimenez, D. Pizarro, M. Mazo, and S. Palazuelos. 2012. Modelling and correction of multipath interference in time of flight cameras. In *CVPR*.

L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik. 2017. Intel RealSense Stereoscopic Depth Cameras. *CVPR Workshops* (2017).

C. Keskin, F. Kıraç, Y.E. Kara, and L. Akarun. 2012. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *ECCV*.

Sameh Khamis, Sean Ryan Fanello, Christoph Rhemann, Julien Valentin, Adarsh Kowdle, and Shahram Izadi. 2018. StereoNet: Guided Hierarchical Refinement for Real-Time Edge-Aware Depth Prediction. *ECCV* (2018).

Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. 2017. Need for Speed: A Benchmark for Higher Frame Rate Object Tracking. In *The IEEE International Conference on Computer Vision (ICCV)*.

Hanme Kim, Stefan Leutenegger, and Andrew J. Davison. 2016. *Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera*.

Yangyan Li, Angela Dai, Leonidas Guibas, and Matthias Nießner. 2015. Database-Assisted Object Retrieval for Real-Time 3D Reconstruction. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library.

Jiajun Lu, Hrvoje Benko, and Andrew D. Wilson. 2017. Hybrid HFR Depth: Fusing Commodity Depth and Color Cameras to Achieve High Frame Rate, Low Latency Depth Camera Interactions. In *CHI*.

Holger Moench, Mark Carpaij, Philipp Gerlach, Stephan Gronenborn, Ralph Gudde, Jochen Hellmig, Johanna Kolb, and Alexander van der Lee. 2016. VCSEL-based sensors for distance and velocity. In *Proc. International Society for Optics and Photonics*.

N. Naik, A. Kadambi, C. Rhemann, S. Izadi, R. Raskar, and S.B. Kang. 2015. A Light Transport Model for Mitigating Multipath Interference in TOF Sensors. *CVPR* (2015).

Yoshihiro Nakabo, Masatoshi Ishikawa, Haruyoshi Toyoda, and Seiichiro Mizuno. 2000. 1ms Column Parallel Vision System and Its Application of High Speed Target Tracking. In *ICRA*.

Richard Newcombe. 2012. *Dense visual SLAM*. Ph.D. Dissertation. Imperial College London, UK.

Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 127–136.

Kohei Okumura, Hiromasa Oku, and Masatoshi Ishikawa. 2011. High-speed gaze controller for millisecond-order pan/tilt camera. In *Proc. of ICRA*. 6186–6191.

Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. 2016. Holoportation: Virtual 3D Teleportation in Real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, 741–754.

OSHA. 2017. *OSHA Technical Manual (OTM) by United States. Occupational Safety and Health Administration. Office of Science and Technology Assessment.*

Matthew O'Toole, Felix Heide, Lei Xiao, Matthias B Hullin, Wolfgang Heidrich, and Kiriakos N Kutulakos. 2014. Temporal frequency probing for 5D transient analysis of global light transport. *ACM TOG* (2014).

Matthew KXJ Pan and Günter Niemeyer. 2017. Catching a real ball in virtual reality. In *Virtual Reality (VR), 2017 IEEE*. IEEE, 269–270.

V. Pradeep, C. Rhemann, S.i Izad, C. Zach, M. Bleyer, and S. Bathiche. 2013. MonoFusion: Real-time 3D Reconstruction of Small Scenes with a Single Web Camera.

H. Rebecq, T. Horstschaefer, G. Gallego, and D. Scaramuzza. 2017. EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time. *IEEE Robotics and Automation Letters* (2017).

Christian Reinbacher, Gottfried Munda, and Thomas Pock. 2017. Real-Time Panoramic Tracking for Event Cameras. *arXiv preprint arXiv:1703.05161* (2017).

RoadToVR. 2016. Analysis of Valve's 'Lighthouse' Tracking System Reveals Accuracy. http://www.roadtovr.com/analysis-of-valves-lighthouse-tracking-system-reveals-accuracy/. (2016).

Jannick P. Rolland, Richard L. Holloway, and Henry Fuchs. 1995. Comparison of optical and video see-through, head-mounted displays. (1995).

Joaquim Salvi, Sergio Fernandez, Tomislav Pribanic, and Xavier Llado. 2010. A state of the art in structured light patterns for surface profilometry. *Pattern recognition* 43, 8 (2010), 2666–2680.

D. Scharstein and R. Szeliski. 2002. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *IJCV* 47, 1-3 (April 2002).

J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. 2011. Real-time Human Pose Recognition in Parts from Single Depth Images. In *CVPR*.

J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. 2013. Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images. In *Proc.*

of CVPR.
Srinath Sridhar, Franziska Mueller, Antti Oulasvirta, and Christian Theobalt. 2015. Fast and Robust Hand Tracking Using Detection-Guided Optimization. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 9. http://handtracker.mpi-inf.mpg.de/projects/FastHandTracker/
Jan Stuhmer, Sebastian Nowozin, Andrew Fitzgibbon, Richard Szeliski, Travis Perry, Sunil Acharya, Daniel Cremers, and Jamie Shotton. 2015. Model-Based Tracking at 300Hz Using Raw Time-of-Flight Observations. In *ICCV*.
Robert W Sumner, Johannes Schmid, and Mark Pauly. 2007. Embedded deformation for shape manipulation. *ACM TOG* 26, 3 (2007), 80.
A. Takgi, S. Yamazaki, and H. Fuchs. 2000. Development of a stereo video see-through HMD for AR systems. (2000).
J. Taylor, L. Bordeaux, T. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. Valentin, B. Luff, A. Topalian, E. Wood, S. Khamis, P. Kohli, S. Izadi, R. Banks, A. Fitzgibbon, and J. Shotton. 2016. Efficient and Precise Interactive Hand Tracking Through Joint, Continuous Optimization of Pose and Correspondences. *SIGGRAPH* (2016).
Jonathan Taylor, Vladimir Tankovich, Danhang Tang, Cem Keskin, David Kim, Philip Davidson, Adarsh Kowdle, and Shahram Izadi. 2017. Articulated Distance Fields for Ultra-Fast Tracking of Hands Interacting. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)* (2017).
Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. 2016. FaceVR: Real-Time Facial Reenactment and Eye Gaze Control in Virtual Reality. *arXiv preprint arXiv:1610.03151* (2016).
Anastasia Tkach, Mark Pauly, and Andrea Tagliasacchi. 2016. Sphere-Meshes for Real-Time Hand Modeling and Tracking. *ACM Transaction on Graphics (Proc. SIGGRAPH Asia)* (2016).
R. Y. Tsai and R. K. Lenz. 1988. Real time versatile robotics hand/eye calibration using 3D machine vision. In *ICRA*.
Julien Valentin, Vibhav Vineet, Ming-Ming Cheng, David Kim, Jamie Shotton, Pushmeet Kohli, Matthias Nießner, Antonio Criminisi, Shahram Izadi, and Philip Torr. 2015. SemanticPaint: Interactive 3d labeling and learning at your fingertips. *ACM Transactions on Graphics (TOG)* (2015).
Shenlong Wang, Sean Ryan Fanello, Christoph Rhemann, Shahram Izadi, and Pushmeet Kohli. 2016. The Global Patch Collider. *CVPR* (2016).
D. Webster and O. Celik. 2014. Experimental evaluation of Microsoft Kinect's accuracy and capture rate for stroke rehabilitation applications. In *Haptics Symposium (HAPTICS), 2014 IEEE*. IEEE, 455–460. https://doi.org/10.1109/haptics.2014.6775498
Li Zhang, Brian Curless, and Steven M. Seitz. 2003. Spacetime Stereo: Shape Recovery for Dynamic Scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 367–374.
Song Zhang, Daniel Van Der Weide, and James Oliver. 2010. Superfast phase-shifting method for 3-D shape measurement. *Optics express* (2010).
Yinda Zhang, Sameh Khamis, Christoph Rhemann, Julien Valentin, Adarsh Kowdle, Vladimir Tankovich, Michael Schoenberg, Shahram Izadi, Thomas Funkhouser, and Sean Fanello. 2018. ActiveStereoNet: End-to-End Self-Supervised Learning for Active Stereo Systems. *ECCV* (2018).
Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rhemann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, and Marc Stamminger. 2014. Real-time Non-rigid Reconstruction using an RGB-D Camera. *ACM Transactions on Graphics (TOG)* 33, 4 (2014).
Chao Zuo, Qian Chen, Guohua Gu, Shijie Feng, Fangxiaoyu Feng, Rubin Li, and Guochen Shen. 2013. High-speed three-dimensional shape measurement for dynamic scenes using bi-frequency tripolar pulse-width-modulation fringe projection. *Optics and Lasers in Engineering* (2013).

A COMMERCIALLY AVAILABLE DEPTH SENSORS

Depth sensing is an active research field for decades and in recent years a wide range of consumer depth sensors have emerged, ranging from the Microsoft Kinect to the most recent Intel RealSense technologies. In the research community, *passive* depth estimation method based on stereo [Bleyer et al. 2011; Hosni et al. 2013] or motion [Pradeep et al. 2013] have been proposed. These methods fail at delivering precise depth measurements due to the *lack of texture* in many scenes. Therefore, most commercially successful commodity depth sensors are based on *active* illumination and mostly fall into the categories: time of flight (TOF), temporal structured light (TSL), spatial structured light (SSL) or active stereo (AS).

Spatial Structured Light (SSL) – e.g. Microsoft Kinect v1. SSL systems [Geng 2011; Salvi et al. 2010] project a spatial pattern into the scene that enables *uniquely* identifying each pixel by looking at a spatial neighborhood around that pixel. If the spatial code can be recognized in the observed camera image, a depth value can be estimated via triangulation. Since this approach estimates depth from a single image, it is *theoretically* capable of delivering very high frame rates. However, depth estimation in these systems is computationally expensive and hence commercial products have sacrificed accuracy over speed and resolution (e.g. quantization artifacts in Kinect V1). Recently proposed efficient variants such as [Fanello et al. 2016] require expensive per-camera training/calibration step, and fails to provide high quality depth data in multi-sensor applications due to interference.

Time of Flight (TOF) – e.g. Microsoft Kinect v2. Time of flight cameras such as the Microsoft Kinect v2 have recently gained popularity. A drawback of TOF is that *multiple frames* need to be captured in order to produce a single depth map. For instance, the Kinect v2 sensor captures raw infrared frames at 300fps in order to produce depth maps at 30fps [Stuhmer et al. 2015]. This means that in order to achieve our desired output framerate of 210fps we would need a camera and depth estimation algorithm that is capable of recording and processing infrared images at 2000fps. This is simply unfeasible with commodity hardware. In addition to framerate challenges, the quality of depth maps generated by TOF cameras is affected by *multipath interference* (MPI), which occurs when the emitted light is reflected from multiple surfaces in the scene before traveling back to the sensor. Despite significant efforts [Bhandari et al. 2014; Freedman et al. 2014; Gupta et al. 2015; Jimenez et al. 2012; Naik et al. 2015; O'Toole et al. 2014], there is no broadly accepted solution.

Temporal Structured Light (TSL) – e.g. Intel SR300. The temporal counterpart to SSL is TSL, which project a *temporal sequence* of coded patterns into the scene. This temporal code makes it possible for each projector pixel to be uniquely identified within a calibrated camera field of view, that observes this pattern over time. After locating pixel pairs in the projector and camera space that have the same temporal code, depth can be computed via triangulation. Although this approach is computationally very efficient it suffers from the same limitation as TOFs: multiple frames are required for a single depth prediction. In general, the bigger the depth range and the higher precision required, the more frames are necessary. This causes significant problems for high speed scenarios. Such approaches are prone to motion artifacts and, similar to TOF, pose significant challenges in terms feasibility of camera hardware.

Active Stereo (AS) – e.g. Intel RealSense D400 series. Active stereo depth sensing uses a calibrated pair of infrared (IR) cameras together with an (IR) illuminator that projects a texture into the scene. Depth is then estimated via triangulation of corresponding points identified in the two images. The role of the active illumination is to project a spatially unique texture that helps in the search of good correspondences. Active stereo has the potential to run at high frame rates because it generates depth on a *per-frame* basis. Stereo matching algorithms are historically computationally expensive, but recent work has demonstrated how to lift this limitation [Fanello et al. 2017b]. Most importantly, multiple AS sensors can work together without interference, hence our choice for this technology.