Polarimetric Dense Monocular SLAM

Luwei Yang[1,*] Feitong Tan[1,*] Ao Li[1] Zhaopeng Cui[1,2] Yasutaka Furukawa[1] Ping Tan[1]

[1] Simon Fraser University [2] ETH Zurich

{luweiy, feitongt, leeaol, furukawa, pingtan}@sfu.ca, zhaopeng.cui@info.ethz.ch

Abstract

This paper presents a novel polarimetric dense monocular SLAM (PDMS) algorithm based on a polarization camera. The algorithm exploits both photometric and polarimetric light information to produce more accurate and complete geometry. The polarimetric information allows us to recover the azimuth angle of surface normals from each video frame to facilitate dense reconstruction, especially at textureless or specular regions. There are two challenges in our approach: 1) surface azimuth angles from the polarization camera are very noisy; and 2) we need a near real-time solution for SLAM. Previous successful methods on polarimetric multi-view stereo are offline and require manually pre-segmented object masks to suppress the effects of erroneous angle information along boundaries. Our fully automatic approach efficiently iterates azimuth-based depth propagations, two-view depth consistency check, and depth optimization to produce a depthmap in real-time, where all the algorithmic steps are carefully designed to enable a GPU implementation. To our knowledge, this paper is the first to propose a photometric and polarimetric method for dense SLAM. We have qualitatively and quantitatively evaluated our algorithm against a few of competing methods, demonstrating the superior performance on various indoor and outdoor scenes.

1. Introduction

Polarization is a natural characteristic of light waves, which conveys rich geometric cues of the surrounding environment, such as directions and shapes. While human vision has not evolved to exploit polarization, certain species of birds and insects[37] are known to sense polarization. Some shrimps [7] even roll their eyeballs (i.e., rotating around the gazing direction) to maximize the polarization contrast. A fundamental challenge in Computer Vision is to develop computational algorithms that exploit polarimetric as well as photometric properties of light transport.

A polarization camera, such as PolarCam [1], has an array of linear polarizer on the top of the CMOS sensor, just like the RGB Bayer filter [14]. The polarization camera can capture the scene under four polarization angles with a single shot, and recover the surface azimuth angle at every pixel in each video frame. An azimuth angle provides an iso-depth direction on the image domain, along which depth values can be propagated to fill-in and improve a depthmap. The challenge is that these azimuth angle estimations are ambiguous and noisy, and false depth propagations damage a depthmap quickly. Furthermore, we need a near real-time algorithm for SLAM (Simultaneous Localization and Mapping) applications. An existing polarimetric stereo solution by Cui et al. [5] requires manually prepared segmentation masks to prevent false propagations and is offline. This paper develops a fully automatic algorithm based on a polarization camera, which exploits both photometric and polarimetric information to produce more accurate and complete depthmaps in real-time.

To achieve this goal, we design efficient algorithms to resolve the azimuth angle ambiguities, and avoid propagating false depths at outlier points. Specifically, we use a rough depthmap to bootstrap the dis-ambiguity process, unlike the expensive graph optimization adopted in [5]. During the azimuth-based depth propagation, we design a two-view propagation and cross-validation approach to avoid propagating outlier points. Our algorithm efficiently iterates 1) two-view depth propagationsand validation, and 2) depth optimization to produce more accurate and complete depthmaps. All the algorithmic steps are carefully designed to enable GPU implementations.

We have evaluated the proposed system on indoor and outdoor scenes with a hand-held monocular polarization video camera. The qualitative and quantitative evaluations demonstrate our superior performance over the traditional methods. To our knowledge, this paper is the first to propose a photometric and polarimetric method for dense SLAM.

*These authors contributed equally to this work.

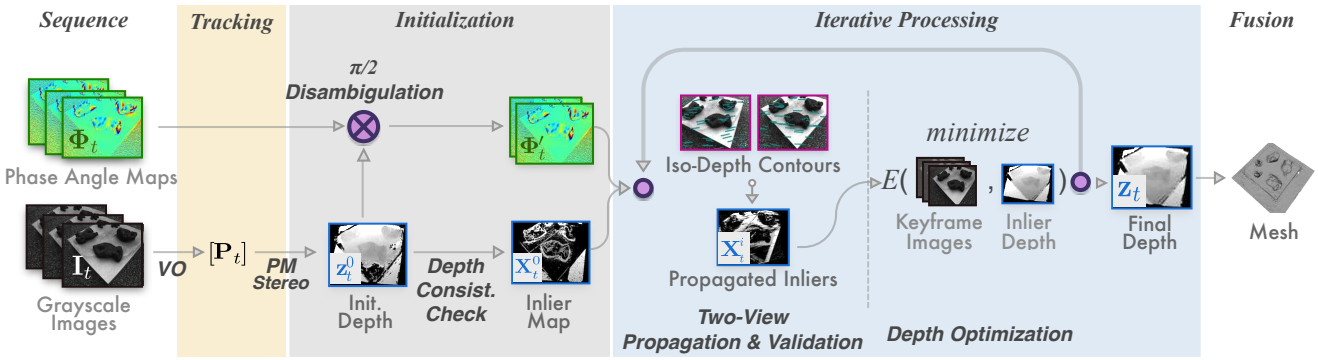


Figure 1: Pipeline of our system. Please refer to the main text for more detail.

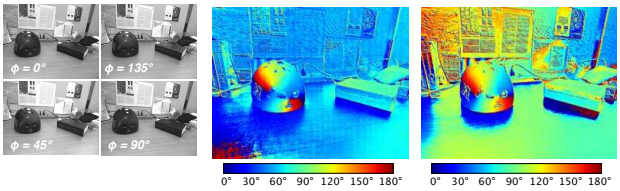
2. Related work

SLAM: Most visual SLAM algorithms [8, 20, 26] are dedicated to improve the camera pose accuracy or system robustness. They track a sparse set of image feature points to solve camera motion, and build a sparse 3D map from these tracked points (see a recent survey [3] for the full review of the SLAM literature).

Dense visual SLAM produces dense depthmaps, important in many applications such as object detection, recognition, obstacle avoidance, and augmented reality. Some methods [35, 38] apply dense stereo as a post-processing. For robustness and accuracy, fusion algorithms use RGB-D cameras [27, 6, 40], which however increase power consumptions and limit the operating capabilities to short-range indoor scenes. Semi-dense SLAM uses more image pixels, in particular along edges, to make tracking more robust and produce denser geometry [12, 11, 10]. However, their geometry still contains many holes. We apply the method in [10] to solve camera poses in realtime and compute a dense depthmap per keyframe for dense mapping.

Recently, unconventional cameras have been used with SLAM algorithms. Kim *et al*. used an event camera to cope with fast camera motions and low-light or high dynamic range scenes [19]. Jo *et al*. built a novel sensor "SpeDo", which solves 6 DOF ego-motions through speckle defocus imaging [16]. None of them builds dense 3D maps. This paper uses an unconventional sensor, polarization camera, to enhance the quality of 3D reconstruction.

Polarimetric 3D modeling: Light polarization encodes surface normal information (i.e., azimuth and zenith angles), which have been exploited in many 3D reconstruction algorithms. The polarimetric shape cues are ambiguous. Earlier methods [2, 24, 25] assumed smooth object surfaces. Recent methods employ shape-from-shading [21, 33, 36] or photometric stereo [29, 9] to deal with the ambiguities. Some algorithms integrate polarimetric constraints with multi-view stereo [32, 23, 5] or a RGB-D camera [17]. All existing methods are computationally expensive and designed for offline object-level 3D modeling. This paper proposes the first online polarimetric dense SLAM solution, which works for objects, indoors, and outdoors.



(a) (b) (c)

Figure 2: (a) four images with different polarization angles; (b) estimated azimuth angle map **without FlatField calibration** (bottom bar shows the phase angle); (c) angle map with our FlatField calibration.

3. Preliminaries

Polarization camera Our polarimetric dense monocoular SLAM (PDMS) system is based on a polarization camera, which has an array of four types of linear polarizers in front of its CMOS sensor and generates a four-channel image. Each channel measures the intensity of a light passing through one of the polarizers. From this polarization image, we can generate a "phase-map", which has two quantities per pixel: 1) the azimuth angle, equivalent to the direction of the surface normal vector projected onto an image plane (up to $\pm\pi/2$ ambiguity); 2) the degree of linear polarization (DOLP), which is a form of confidence metric in the azimuth estimation. We follow standard techniques in computing the phase-map, whose details can be found in our supplementary material.

Flat-field calibration A polarization camera requires extra careful flat-field calibration [22], which is to compensate the sensitivity and dark current variations across different pixels in the CMOS sensor. Typically, the flat-field can be calibrated by capturing an image of a uniform signal, where each pixel value is proportional to its sensitivity.

Digital cameras often have build-in flat-field calibration by their manufacturers before shipping, which is usually insufficient for a polarization camera, because the input signal should be unpolarized at the same time. Otherwise, the signal after going through the polarizer array is no longer uniform, violating the assumptions in the flat-field calibration. We have experimented several different light sources and setups and eventually find that a LCD monitor covered by 3-5 layers of A4 printing papers produces the designed uniform and *unpolarized* input. We then take 100 images and compute the average image to smooth out noises. This average image is the flat-field, encoding the relative sensitivity of different pixels. Denoting this average image as F, the multiplier factor map could be computed by: $\hat{F} = \frac{max(F)}{F}$. We calibrate the raw image as,

$$I_{corr} = I_{raw} \circ \hat{F}, \tag{1}$$

where the operator $\circ$ is an element-wise multiplication to correct the pixel sensitivities. Figure 2(c) shows the result of the flat-field calibration.

System overview Figure 1 shows the overview of our system architecture. The inputs to our program are the grayscale video frames and a phase angle map per frame captured by a polarization camera. We use an existing visual odometry (VO) algorithm [10] to solve the camera poses. In the next, we initialize a depthmap per keyframe by the PatchMatch stereo [13], which helps to resolve the $\pi/2$-ambiguities in the azimuth angles. The followed depth consistency check produces a sparse set of inlier 3D points with high confidence for each keyframe. Our system then iterates two steps, 1) two-view depth propagation and validation, and 2) depth optimization, to produce the final depthmap. Finally, we fuse those depthmaps at different keyframes to create an integrated triangle mesh.

Pose estimation Our system employs a state-of-the-art visual odometry software, DSO [10], while any other software can be used (e.g., ORB-SLAM [26] or LSD-SLAM [11]). We take the mean of the four channel intensities to generate a grayscale image as an input to DSO. DSO generates camera poses and key-frames.

4. Polarimetric dense mapping on GPU

Our polarimetric dense mapping consists of two major components, initialization and iterative processing, and both are designed to run entirely on GPU. The iterative processing further iterates two steps: 1) two-view depth propagation and validation; 2) depthmap optimization. We now explain the details of each step, where the overall algorithm including the initialization steps are given in Algorithm 1.

Algorithm 1: Depth Estimation on Keyframe K_t

Input : $\{\mathbf{I}_i, \mathbf{\Phi}_i, \mathbf{P}_i\}_{i\in 1,2,...,t}$
Input : $\mathbf{z}_{t-1}$
Output: $\mathbf{z}_t$

Initilization:
$\mathbf{z}_t^0 \leftarrow$ PatchMatchStereo($\mathbf{I}_t$, ...,$\mathbf{I}_{t-2}$)
$\mathbf{\Phi}_t' \leftarrow$ Disambiguity($\mathbf{\Phi}_t$)
$\mathbf{X}_t^0 \leftarrow$ DepthConsistCheck($\mathbf{z}_t^0$, $\mathbf{z}_{t-1}$)
$\mathbf{a}^0 \leftarrow \mathbf{z}_t^0$

Iterative Optimization:
for $i \leftarrow 1$ ***to*** it_{max} **do**
  $\mathcal{N}_t^i \leftarrow$ TraceDepthOnK_t($\mathbf{\Phi}_t'$, $\mathbf{z}_t^{inlier}$)
  $\mathcal{N}_r^i \leftarrow$ TraceDepthOnK_r($\mathbf{\Phi}_r'$, $\mathbf{z}_t^{inlier}$)
  $\mathbf{X}_t^i \leftarrow \mathbf{X}_t^{i-1} \cup$ PropagationCheck($\mathcal{N}_t^i$, $\mathcal{N}_r^i$)
  $\mathbf{z}_t^i \leftarrow$ OptimizeDataTerm($\mathbf{z}_t^{i-1}$, $\mathbf{a}_{t-1}$, $\mathbf{X}_t^i$)
  $\mathbf{a}_t^i \leftarrow$ OptimizeSmoothTerm($\mathbf{z}_t^i$, $\mathbf{a}_{t-1}$)
end

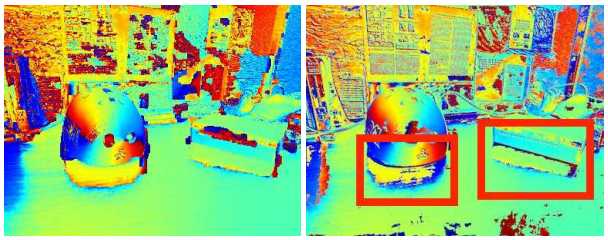


Figure 3: The disambiguated azimuth angle by [5] (shown in left) and by our method (shown in right). The original phase angle map is shown in Figure 2 (c).

4.1. Initialization

Depthmap initialization: Given a keyframe K_t, the system reconstructs an initial depthmap $\mathbf{z}_t^0$ by the GPU-accelerated PatchMatch stereo [13] with the regularizer introduced in [15]. The symbol t denotes a keyframe index. The depthmap is computed within a predefined depth range $[z_{min}, z_{max}]$. We remove spurious depth values in $\mathbf{z}_t^0$ via a simple consistency check with the depthmap $\mathbf{z}_{t-1}$ computed at the previous key-frame: we reproject $\mathbf{z}_{t-1}$ into K_t and filter out depth values where the depth difference is more than 1% of $(z_{max} - z_{min})$. The remaining points are considered as inliers whose depths will be propagated and optimized in the iterative processing. We denote this initial set of inliers as $\mathbf{X}_t^0$.

Azimuth angle dis-ambiguation The azimuth angle estimation inherently suffers from a $\pi/2$-ambiguity, due to two different types of polarized reflections: specular and diffuse. In many pixels, polarized specular reflection is dominant and this ambiguity can be ignored. In the other cases, the azimuth angle estimation needs to be corrected by $\pi/2$. Cui *et al*. proposed an effective graph-based solution [5],

but is computationally too expensive for realtime applications.

We found that the following process works well and runs efficiently on GPU. The idea is simple. Depth values would be constant along a direction perpendicular to the true azimuth angle. We trace iso-depth contours for the two possible azimuth angles, and simply pick the contour with the smaller variance in depth values (where available) based on the initial depthmap. The azimuth ambiguity occurs for diffuse dominant surfaces and the disambiguity process is applied to pixels whose degree of linear polarization (DOLP) is below 0.30. Note that this process fails where the depth values are completely missing, but works well where depth values are noisy. Figure 3 shows a comparison between our simple method and the graph optimization method in [5]. Our method correctly resolves the ambiguity for most of the pixels under the cast shadow of the halmet and book (see the highlighted region).

4.2. Two-view propagation and validation

The success of azimuth-based depth propagation critically depends on the accuracy of the seed points where the tracing starts. Depth discontinuity is one major failure mode illustrated in Figure 4. Note that previous works rely on object segmentation information as input to avoid propagating incorrect depth values at object boundaries [41, 5]. We need a fully automated realtime solution for SLAM. Our approach is to perform the azimuth-based depth propagation in the current keyframe and one well-separated reference keyframe simultaneously, and perform two-view consistency check to avoid propagating outliers.

Let K_t denote the current keyframe. We choose the reference keyframe K_r that is well-separated from K_t. More precisely, K_r is chosen to be the most recent key-frame whose rotation differs by more than 30 degrees from K_t.

Two-view depth propagation To ensure high quality propagation, in the i-th iteration, we propagate the inlier 3D points $\mathbf{X}_t^i$ in both K_t and K_r simultaneously. For a pixel x with unknown depth (either in K_r or K_s), we collect the inlier 3D points in $\mathbf{X}_t^i$ that are projected on its iso-depth contour, and compute a probability distribution for the depth of x. We estimate[30] a mixture of a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ and a uniform distribution between $[z_{min}, z_{max}]$ to model this depth distribution, where the uniform distribution accounts for random depth noisy. After that, we model the depth at x by the Gaussian component $\mathcal{N}(\mu, \sigma)$. Note this propagation can be easily parallelized for all pixels in K_t and K_r. But we only consider the propagated 3D points in K_t as the set of candidate inlier points $\Delta\mathbf{X}_t^i$.

Two-view consistency check Now, we use the propagated 3D points in the reference view K_r to cross validate the candidate points in $\Delta\mathbf{X}_t^i$. We project them to the keyframe K_t, and check the depth consistency at their projected positions. Suppose the pixel x_r in K_r is projected to x_t in K_t according to its mean depth μ_r. According to our propagation, the depth of x_t is modeled by a Gaussian distribution $\mathcal{N}_t(\mu_t, \sigma_t)$. We evaluate x_r's depth distribution in K_t and compare its consistency with $\mathcal{N}_t(\mu_t, \sigma_t)$ to decide if the propagated 3D point at x_t should be discarded.

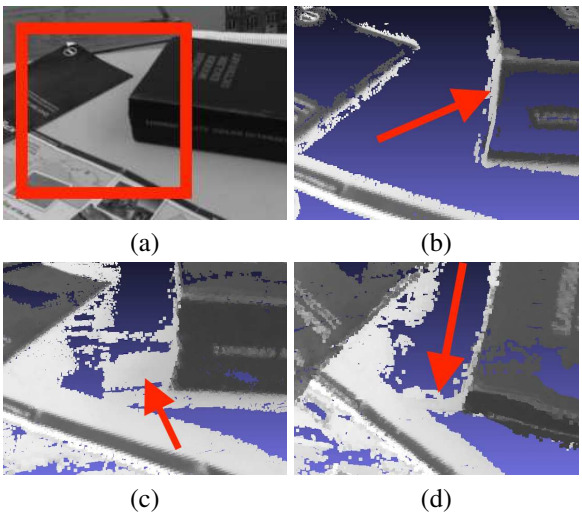


Figure 4: Naïve depth propagation magnifies 3D reconstruction errors. (a) is the input frame, and (b) shows the initial '*inlier*' 3D points $\mathbf{X}_t^0$ with some noisy 3D points (in white color) along the book's edge. (c) and (d) are the results by naïve depth propagation, visualized from two different viewpoints, where the noisy 3D points are incorrectly propagated along iso-depth contours.

Specifically, we sample x_r's depth distribution $\mathcal{N}_r(\mu_r, \sigma_r)$ in the reference view, and compute its depth in the keyframe K_t according to these sampled depths. In this way, we can compute x_r's depth distribution in the keyframe K_t, denoted as $\mathcal{N}_{r\to t}(\mu_{r\to t}, \sigma_{r\to t})$. Now, we compute the KL divergence between $\mathcal{N}_{r\to t}(\mu_{r\to t}, \sigma_{r\to t})$ and $\mathcal{N}_t(\mu_t, \sigma_t)$, and discard the propagated 3D point at x_t if the KL divergence is larger than 0.5 or if the absolute difference of $|\mu_t - \mu_{r\to t}|$ is larger than 1% of $|z_{max} - z_{min}|$. If a pixel x_t in K_t is not projected by any point x_r in K_r, we also discard its propagated depth.

After discarding inconsistent points in $\Delta\mathbf{X}_t^i$, we update the inlier set to $\mathbf{X}_t^{i+1} = \mathbf{X}_t^i \cup \Delta\mathbf{X}_t^i$ to move the depthmap optimization to finish an iteration.

4.3. Depthmap optimization

The last step of the iteration enforces the photometric, polarimetric, and spatio-temporal regularization constraints to optimize the depthmap. Specifically, we minimize an energy function with a data term and a smooth term over three neighboring keyframes K_t, K_{t-1}, K_{t-2},

$$E = \int_\Omega \lambda E_{data} + E_{smooth}. \tag{2}$$

iteration 0 iteration 3 iteration 6

Figure 5: More *inlier* pixels (marked as green) are reconstructed during the iterations.

Here, the weight λ balances the relative significance of the smoothness regularization term, and is set to 5.0 in all our experiments, and Ω is the image plane.

Data term Our data term is defined between two images, a source keyframe (K_t) and a reference keyframe (K_{t-1} or K_{t-2}). For denotation simplicity, we denote the source as $\mathbf{I}$ and the reference as $\mathbf{I}'$ in the following. The data term consists of a photometric error and a contour error, combined with a weight τ. On very pixel p, we have,

$$E_{data}(p) = (1-\tau(p))E_{photo}(d_p, \mathbf{n}_p) + \tau(p)E_{contour}(z_p). \tag{3}$$

The spatially variant weight $\tau(p)$ is defined as $\tau(p) = e^{-\zeta|\nabla I_p|^\eta}$, where ∇I_p is the image gradient at p. We fix the parameters $\eta = 0.8$, $\zeta = 3.1$ in all our experiments. This weight $\tau(p)$ is stronger on featureless areas and weaker nearby image edges, since the photometric term tends to produce better result at those well textured regions.

The photometric error with parameters of the surface normal $\mathbf{n}_p$ and the distance to the origin d_p follows the conventional approach [13] with the detailed definition in our supplementary file. The contour consistency error is evaluated as

$$E_{contour}(z_p) = \begin{cases} |z_p - \mu_p|, & \text{if } p \in \mathbf{X}_t^i; \\ c, & \text{otherwise.} \end{cases} \tag{4}$$

where the z_p is the depth at the pixel p (i.e. the distance from a 3D point to the camera image plane), μ_p is the mean of the Gaussian distribution of p's depth computed through propagation. The depth z_p at a pixel $p = (p_x, p_y)$ can be computed from the parameters $d_p, \mathbf{n}_p$ as,

$$z_p = \frac{-d_p f}{[p_x - c_x, p_y - c_y, f]\mathbf{n_p}}. \tag{5}$$

Here, f is the camera focal length and (c_x, c_y) is the principal point.

Smooth term The smoothness term also consists of two components, namely,

$$\begin{aligned} E_{smooth}(p) =& \tau(p)|\nabla z_p|_\epsilon \\ &+ \lambda_a \tau(p)|sin(\phi)\nabla_x z_p - cos(\phi)\nabla_y z_p|_\epsilon, \end{aligned} \tag{6}$$

Here, the $|\cdot|_\epsilon$ defined as Huber norm with threshold of ϵ. And the $\tau(p)$ is the gradient adaptive weight defined in Equation 3 which gives a strong penalization on featureless area. The weight λ_a balances these two terms, and is fixed to 0.4 in all our experiments. In second term, the operators ∇_x and ∇_y compute the derivative along on x-axis and y-axis respectively, and ϕ is the azimuth angle after disambiguation. This term is adopted from [5], which enforces the depthmap gradient to be consistent to the azimuth angle, i.e. $\tan(\phi_p) = \nabla_y z_p / \nabla_x z_p$.

Optimization method The energy function in Equation (2) is difficult to optimize, with a non-convex data term and a convex regularizer [34, 28, 15]. Similar to [28], we introduce an auxiliary variable a to simplify this problem, which turns the original energy function into,

$$E' = \int_\Omega \lambda E_{data}(z, \mathbf{n}) + E_{smooth}(a) + \frac{1}{2\theta}\|a - z\|_2^2 . \tag{7}$$

This auxiliary variable decouples the data term and smoothness term, so that they can be optimized separately. The last term $\frac{1}{2\theta}\|a-d\|_2^2$ is introduced to coupling the original and auxiliary variables together by gradually reducing θ during iterations.

After this decoupling, the data term becomes

$$\begin{aligned} E_{data}(p) =& (1-\tau_p)E_{photo}(d_p, \mathbf{n}_p) \\ &+ \tau_p E_{contour}(z_p) + \frac{1}{2\theta}\|a_p - z_p\|_2^2 , \end{aligned} \tag{8}$$

Note that the d_p can be converted from z_p by Equation 5. We adopted the PatchMatch variant [13] to minimize it, which performs a parallel updating scheme that could run efficiently on GPU architecture.

On the other hand, the smoothness term becomes,

$$\begin{aligned} E_{smooth} =& \tau_p|\nabla a_p|_\epsilon + \lambda_a \tau_p |sin(\phi)\nabla_x a_p - cos(\phi)\nabla_y a_p|_\epsilon \\ &+ \frac{1}{2\theta}\|a_p - z_p\|_2^2 , \end{aligned} \tag{9}$$

and we can solve d by the ROF method [4], which is easy to implement for parallel computation. At the first iteration, we set the parameter $\theta = 3.0$. We then iterate the optimization of E_{data} and E_{smooth} while reducing the coupling parameter θ with a factor of 1.5 over the iterations.

Figure 5 shows the propagated inlier 3D points in the 0th, 3rd, and 6th iterations, where the inlier pixels are colored in green. As we can see, more inlier points will be generated during the iterations, and featureless areas (e.g. the white table) are filled with propagated inliers.

Finally, depthmaps at different keyframes are fused into the final surface model by the InfiniTAM system [18].

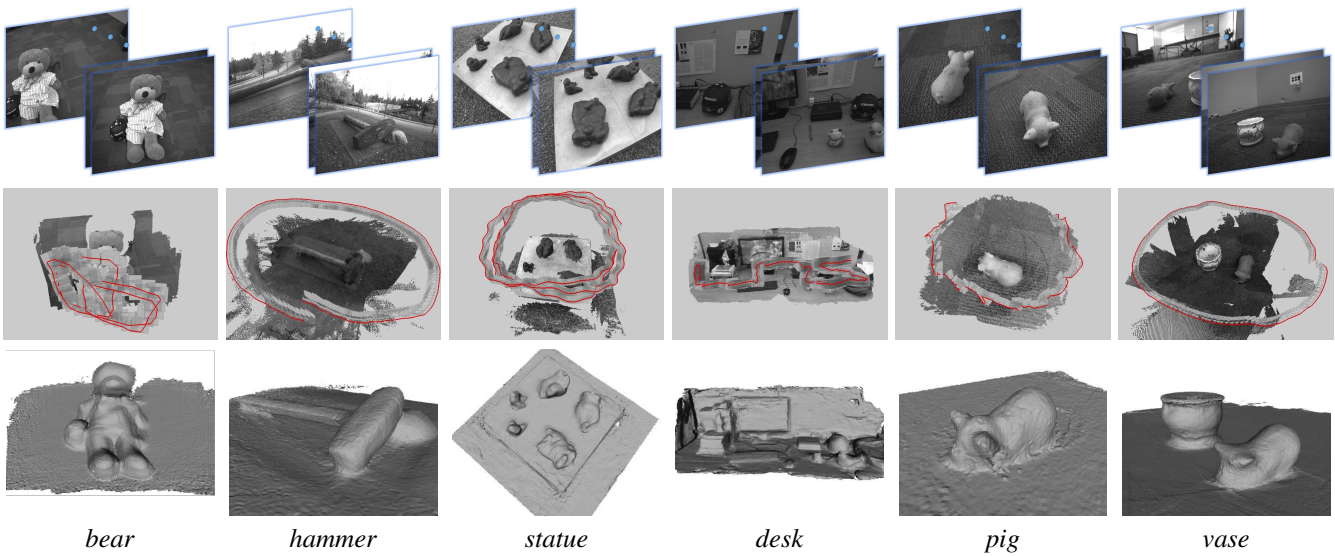


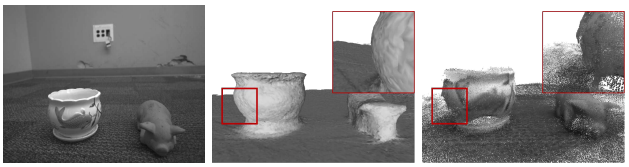
Figure 6: Our dataset captured by a PolarM[1] Camera. The top row shows some sample frames of the original sequence; the second row shows the camera motion trajectory; and the bottom row shows the mesh fused by our system.

5. Experiments

We evaluate our system with a polarization camera, the PolarM [1] camera, mounted with an 8-mm lens. The video resolution is 772×600 pixels, and the camera intrinsic parameters and lens distortions are calibrated beforehand with the calibration package [39]. All experiments are carried out on a computer with an Intel Core-i7 5820k CPU and 64G RAM and an NVIDIA GTX Titan X GPU running Ubuntu 14.04. Our polarimetric dense monocular SLAM (PDMS) takes 1.2 seconds to process a keyframe, including the depth initialization, dis-ambiguiation, and iterative processing. Each iteration of depth propagation, validation, and optimization takes 124 ms.

Typical results with surface fusion Figure 6 shows our results on various indoor and outdoor scenes, where the first row shows some sample input frames, and the second and third rows are the camera trajectories and fused dense mesh respectively. The examples *bear*, *desk*, *vase*, and *pig* are captured indoors under office light, while the examples *hammer* and *statue* are captured outdoors under nature illumination. Several examples contain large untextured areas (e.g. *statue*, *desk*, *pig*, *vase*), some examples contain objects with strong specular reflection (e.g. *statue*, *vase*). Our method consistently performs well on these various examples.

Compare with other dense reconstruction systems We compare our system with Remode[30] and MonoFusion[31]. We modified the code shared by the authors of Remode to use DSO [10] for camera tracking to ensure a fair comparison. The MonoFusion [31] does not have open source implementations, so we implement it by ourselves with the more recent PatchMatch stereo algorithm[13]. We further reinforce MonoFusion by incorporating a smoothness regularizer [15] in the PatchMatch stereo. All these three methods only use photometric information to solve a dense depthmap per keyframe. Comparison with them demonstrate the effectiveness of incorporating polarimetric information. Figure 9 shows the depthmaps computed at a single keyframe for the *bear*, *hammer*, *statue*, and *desk* examples. To make the comparison easier, for each example, we further provide a color-coded surface normals which is computed from the depthmap. Remode[30] generates much sparse reconstruction which leads to poorly fused mesh (see the supplementary video). The *bear* example is relatively easy with rich texture everywhere except the hat (highlighted in a red rectangle) to facilitate stereo matching. Our method produces smoother and denser point clouds, which can be better seen from the normal map. The *hammer* example is captured in outdoor under strong sunlight, where RGB-D sensors will not work. For this example, only our method captures the shape of the hammer faithfully. The PatchMatch stereo[13] in our implementation of MonoFusion generates noisy 3D points floating in space. The regularizer[15] can alleviate this problem, but still generates distorted reconstruction, especially at the handle (see the region highlighted by a red frame). These comparisons are most evident in the normal map. The *statue* is a challenging outdoor example, with some black metal statues sit on a white featureless base. Again, the PatchMatch stereo generates a large hole on the featureless



Scene Our method Method in [5]

Figure 7: Comparison with the method in [5]. We register all the depthmaps computed by [5] together to compare with our fused result.

base, even the regularizer will not solve this problem. In comparison, our method recovers iso-depth contours to propagate depth information into the hole from its boundary (more comparisons are in the supplmentary video). The *desk* examples is a small scale scene with multiple objects. Unlike the other methods, our method captures faithful shape details.

Comparison with [5] We further compare our system with [5] which is an offline polarimetric multi-view stereo method. The authors of [5] kindly run their system on our data with camera poses solved by our method. Figure 7 shows the comparison on an indoor scenes. The method in [5] produces noisy results on the *vase*, because it is designed for reconstructing a single object with manually prepared segmentation masks to avoid propagating outliers at occlusion boundaries. In comparison, the two-view propagation and validation strategy in our method successfully solved this problem. Furthermore, the method in [5] takes around 100 seconds to reconstruct a depthmap, while our method takes only 1.2 seconds, achieving a significant speedup by two magnitudes.

Quantitative Analysis In order to quantitatively analyze the improvement made by the polarimetric approach, we evaluate the reconstruction quality of a planar surface in the *statue* example. As shown in the lower right corner of Figure 8, we mark some pixels (as indicated in red) as the region of interest to compute the cumulative distribution function (CDF) of reconstruction errors of our method, MonoFusion, and the reinforced MonoFusion. We do not include Remode[30] because it tends to generate a sparse set of reliable points, which leads to poor fusion. To obtain a reference 'ground truth', we apply RANSAC to fit a plane from some manually picked 3D points on the base. For each reconstructed 3D point, we compute its shortest distance to this fitted plane as its reconstruction error. In this way, we can evaluate the reconstruction error for all the three methods. The CDFs of the three methods are shown in Figure 8, where the horizontal axis is the reconstruction error and the vertical axis is the percentage of pixels. In this chart, the unit of reconstruction error is set to the length of the white base under the statues. These curves tell us at each error level what percent of 3D points have accuracy higher than

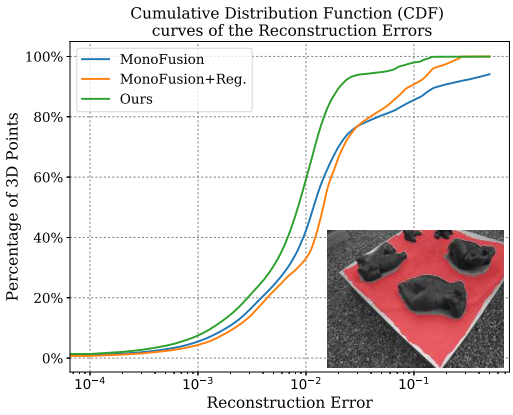


Figure 8: The cumulative distribution function (CDF) curves of the reconstruction errors from our method, the PatchMatch stereo [13] and the PatchMatch reinforced with a regularizer[15]. The image in the lower-right corner shows the region of interest in red.

that error level. It is clear that our polarimetric method produces more accurate result than both PatchMatch stereo[13] and the one reinforced with a regularizer[15]. For example, 60% of 3D points of our reconstructed points have a error smaller than 0.01, while that percentage drops to 40% and 32% for the reinforced MonoFusion and MonoFusion respectively.

6. Conclusion

This paper presents a polarimetric dense monocular SLAM (PDMS) algorithm that reconstructs a dense 3D depthmap in real-time for each keyframe of the input video. These individual depthmaps are further fused to produce an integrated triangle mesh to facilitate other applications. Our method exploits a novel camera, the polarization camera, to enhance 3D reconstruction at featureless and specular regions, which are long-standing difficulties in multi-view stereo vision. In particular, we design an iterative framework of two-view depth propagation, validation, and optimization to make the polarimetric multi-view stereo work without manually segmented object masks. To our knowledge, this is the first real-time polarimetric dense monocular SLAM algorithm.

Acknowledgements

This project is supported by the Canada NSERC Discovery project 611664 and Discovery Acceleration Supplement 611633.

References

[1] The polarm camera. https://www.4dtechnology.com/products/polarimeters/polarcam/.

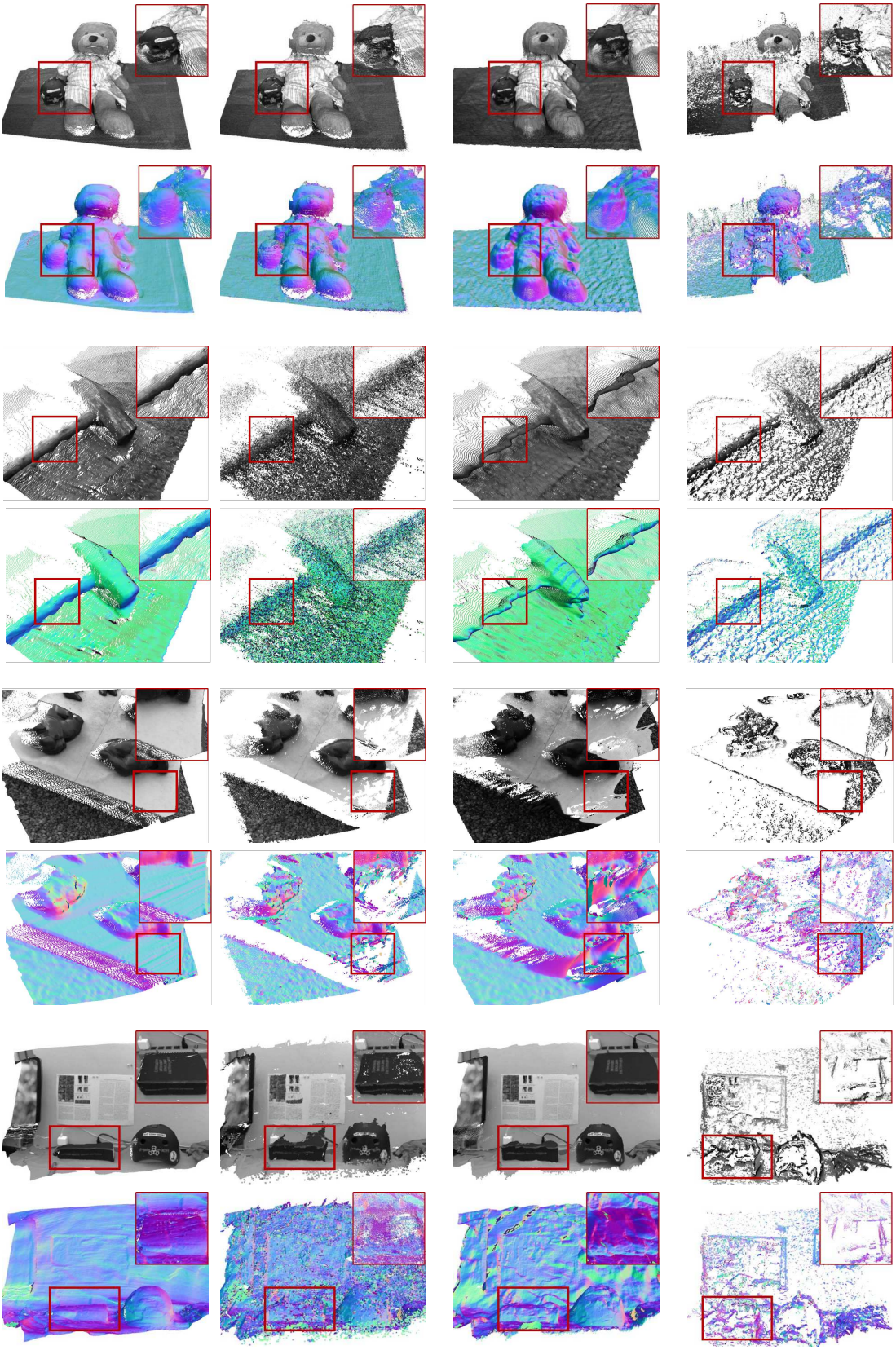


Figure 9: Comparison with Remode[30], MonoFusion[31], and our reinforced MonoFusion. Please refer to the main text for details.

[2] G. A. Atkinson and E. R. Hancock. Recovery of surface orientation from diffuse polarization. *IEEE Trans. on Image Processing (TIP)*, 15(6):1653–1664, 2006.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. on Robotics*, 32(6):1309–1332, 2016.

[4] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1):89–97, 2004.

[5] Z. Cui, J. Gu, B. Shi, P. Tan, and J. Kautz. Polarimetric multi-view stereo. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] A. Dai, M. Niesner, M. Zollhöfer, S. Izadi, and C. Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans. on Graphics (TOG)*, 36(3), 2017.

[7] I. M. Daly, M. J. How, J. C. Partridge, S. E. Temple, N. J. Marshall, T. W. Cronin, and N. W. Roberts. Dynamic polarization vision in mantis shrimps. *Nature Communications*.

[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6), 2007.

[9] O. Drbohlav and R. Šára. Unambiguous determination of shape from photometric stereo with unknown light sources. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, 2001.

[10] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*.

[11] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 834–849. Springer, 2014.

[12] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, pages 1449–1456, 2013.

[13] S. Galliani, K. Lasinger, and K. Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, pages 873–881, 2015.

[14] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., 2006.

[15] P. Heise, S. Klose, B. Jensen, and A. Knoll. Pm-huber: Patchmatch with huber regularization for stereo matching. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, pages 2360–2367, 2013.

[16] K. Jo, M. Gupta, and S. K. Nayar. Spedo: 6 dof ego-motion sensor using speckle defocus imaging. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, pages 4319–4327, 2015.

[17] A. Kadambi, V. Taamazyan, B. Shi, and R. Raskar. Polarized 3d: High-quality depth sensing with polarization cues. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, 2015.

[18] O. Kahler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S. Torr, and D. W. Murray. Very high frame rate volumetric integration of depth images on mobile device. *IEEE Trans. on Visualization and Computer Graphics (TVCG)*, 22(11), 2015.

[19] H. Kim, S. Leutenegger, and A. J. Davison. A new variational framework for multiview surface reconstruction. In *Proc. of European Conference on Computer Vision (ECCV)*. Springer, 2016.

[20] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Proc. of International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 225–234. IEEE, 2007.

[21] A. H. Mahmoud, M. T. El-Melegy, and A. A. Farag. Direct method for shape recovery from polarization and shading. In *International Conference on Image Processing (ICIP)*, 2012.

[22] J. Manfroid. On ccd standard stars and flat-field calibration. *Astronomy and Astrophysics Supplement Series*, 118(2):391–395, 1996.

[23] D. Miyazaki, M. Kagesawa, and K. Ikeuchi. Transparent surface modeling from a pair of polarization images. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 26(1):73–82, 2004.

[24] D. Miyazaki, R. T. Tan, K. Hara, and K. Ikeuchi. Polarization-based inverse rendering from a single view. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, 2003.

[25] O. Morel, F. Meriaudeau, C. Stolz, and P. GorriaK. Polarization imaging applied to 3D reconstruction of specular metallic surfaces. In *Proc. of Machine Vision Applications in Industrial Inspection*, 2005.

[26] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Trans. on Robotics*, 31(5):1147–1163, 2015.

[27] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc. of International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136. IEEE, 2011.

[28] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, pages 2320–2327. IEEE, 2011.

[29] T. T. Ngo, H. Nagahara, and R. Taniguchi. Shape and light directions from shading and polarization. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2015.

[30] M. Pizzoli, C. Forster, and D. Scaramuzza. Remode: Probabilistic, monocular dense reconstruction in real time. In *Proc. of International Conference on Robotics and Automation (ICRA)*, pages 2609–2616. IEEE, 2014.

[31] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche. Monofusion: Real-time 3d reconstruction of small scenes with a single web camera. In *Proc. of International Symposium on Mixed and Augmented Reality (ISMAR)*, 2013.

[32] S. Rahmann and N. Canterakis. Reconstruction of specular surfaces using polarization imaging. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2001.

[33] W. A. P. Smith, R. Ramamoorthi, and S. Tozza. Linear depth estimation from an uncalibrated, monocular polarisation image. In *Proc. of European Conference on Computer Vision (ECCV)*, 2016.

[34] F. Steinbrücker, T. Pock, and D. Cremers. Large displacement optical flow computation without warping. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, pages 1609–1614. IEEE, 2009.

[35] J. Stühmer, S. Gumhold, and D. Cremers. Real-time dense geometry from a handheld camera. In *Joint Pattern Recognition Symposium*, pages 11–20. Springer, 2010.

[36] S. Tozza, W. A. P. Smith, D. Zhu, R. Ramamoorthi, and E. R. Hancock. Linear differential constraints for photo-polarimetric height estimation. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, 2017.

[37] R. Wehner and M. Muller. The significance of direct sunlight and polarized skylight in the ants celestial system of navigation. *Proceedings of the National Academy of Science (PNAS)*, 103(33):12575–12579, 2006.

[38] A. Wendel, M. Maurer, G. Graber, T. Pock, and H. Bischof. Dense reconstruction on-the-fly. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 1450–1457. IEEE, 2012.

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

[40] Q.-Y. Zhou and V. Koltun. Depth camera tracking with contour cues. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 632–638, 2015.

[41] Z. Zhou, Z. Wu, and P. Tan. Multi-view photometric stereo with spatially varying isotropic materials. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2013.