A Practical Stereo Depth System for Smart Glasses

Jialiang Wang[1*] Daniel Scharstein[1] Akash Bapat[1] Kevin Blackburn-Matzen[2†]
Matthew Yu[1] Jonathan Lehman[1] Suhib Alsisan[1] Yanghan Wang[1] Sam Tsai[1]
Jan-Michael Frahm[1] Zijian He[1] Peter Vajda[1] Michael F. Cohen[1] Matt Uyttendaele[1]
[1]Meta Platforms Inc. [2]Adobe

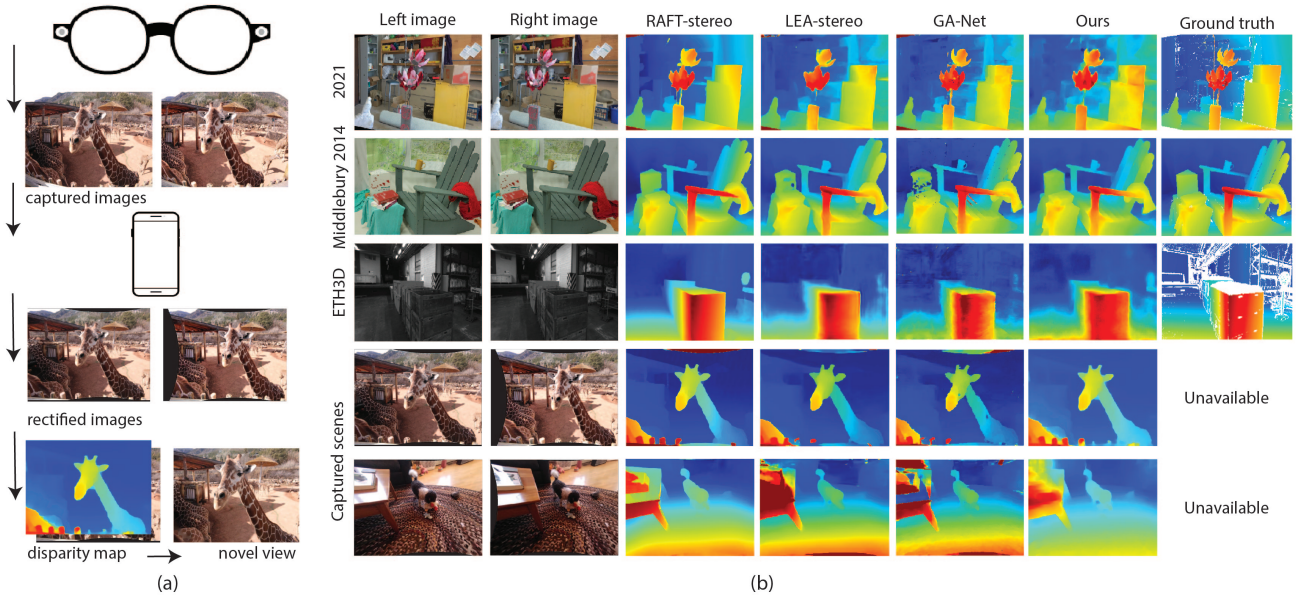

Figure 1. **(a) Overview.** The user captures a stereo pair with smart glasses. The images are sent to the user's phone for processing, including online rectification and fast disparity prediction via neural networks. The predicted disparity map is then used to generate visual effects by rendering novel views. **(b) Zero-shot accuracy.** Our stereo network, Argos, achieves high accuracy despite not being trained on these datasets, being quantized to 8 bits, and being significantly faster than SotA models such as RAFT-stereo [18], GA-Net [45] and LEA-Stereo [3].

Abstract

We present the design of a productionized end-to-end stereo depth sensing system that does pre-processing, online stereo rectification, and stereo depth estimation with a fallback to monocular depth estimation when rectification is unreliable. The output of our depth sensing system is then used in a novel view generation pipeline to create 3D computational photography effects using point-of-view images captured by smart glasses. All these steps are executed on-device on the stringent compute budget of a mobile phone, and because we expect the users can use a wide range of smartphones, our design needs to be general and cannot be dependent on a particular hardware or ML accelerator such as a smartphone GPU. Although each of these steps is well studied, a description of a practical system is still lacking. For such a system, all these steps need to work in tandem with one another and fallback gracefully on failures within the system or less than ideal input data. We show how we handle unforeseen changes to calibration, e.g., due to heat, robustly support depth estimation in the wild, and still abide by the memory and latency constraints required for a smooth user experience. We show that our trained models are fast, and run in less than 1s on a six-year-old Samsung Galaxy S8 phone's CPU. Our models generalize well to unseen data and achieve good results on Middlebury and in-the-wild images captured from the smart glasses.

*Email: jialiangw@meta.com
†work was done at Meta

1. Introduction

Stereo disparity estimation is one of the fundamental problems in computer vision, and it has a wide variety of applications in many different fields, such as AR/VR, computational photography, robotics, and autonomous driving. Researchers have made significant progress in using neural networks to achieve high accuracy in benchmarks such as KITTI [21], Middlebury [28] and ETH3D [30].

However, there are many practical challenges in using stereo in an end-to-end depth sensing system. We present a productionized system in this paper for smart glasses equipped with two front-facing stereo cameras. The smart glasses are paired with a mobile phone, so the main computation happens on the phone. The end-to-end system does pre-processing, online stereo rectification, and stereo depth estimation. In the event that the rectification fails, we fallback on monocular depth estimation. The output of the depth sensing system is then fed to a rendering pipeline to create three-dimensional computational photographic effects from a single user capture. To our knowledge there is limited existing literature discussing how to design such an end-to-end system running on a very limited computational budget.

The main goal of our system is to achieve the best user experience to create the 3D computational photography effects. We need to support any mainstream phone the user chooses to use and capture any type of surroundings. Therefore, the system needs to be robust and operate on a very limited computational budget. Nevertheless, we show that our trained model achieves on-par performance with state-of-the-art (SotA) networks such as RAFT-stereo [18], GA-Net [45], LEA-stereo [3] and MobileStereoNet on the task of zero-shot depth estimation on the Middlebury 2014 dataset [28] despite our network being orders of magnitude faster than these methods.

The main technical and system contributions are:

1. We describe an end-to-end stereo system with careful design choices and fallback plans. Our design strategies can be a baseline for other similar depth systems.
2. We introduce a novel online rectification algorithm that is fast and robust.
3. We introduce a novel strategy to co-design a stereo network and a monocular depth network to make both networks' output format similar.
4. We show that our quantized network achieves competitive accuracy on a tight compute budget.

2. Related work

The focus of our work is building a complete stereo system for flexible-frame smart glasses with robust and lightweight processing on a mobile device. This encompasses system design, online rectification, mono and stereo networks, and 3D effects. While there is much related work for each component, few papers focus on the entire system design, despite its importance. One example is the work by Kopf et al. [13], which presents an end-to-end system for one shot 3D photography, but the depth provider is a monocular depth network. Another example is the Google's work on Cinematic photos [11], which also uses an encoder-decoder monocular depth estimation network. As for stereo, most recent deep-learning-based papers assume the input to the stereo depth system is calibrated and rectified [3, 16, 18, 38, 41]. Barron et al. [1] proposed a bilateral approach for stereo to synthesize defocus that is used in Google's lens blur feature. Perhaps most relevant to our work is the work by Zhang et al. [46], which uses dual cameras and dual pixels to estimate depth for defocus blur and 3D photos. No paper, to the best of our knowledge, describes a stereo system with online rectification and failure handling when rectification fails. Below we briefly summarize related work for each component in our design.

Online stereo rectification: Since imperfect rectification is a problem for almost all practical stereo systems, a common approach is to allow stereo algorithm some vertical slack, e.g., by only utilizing horizontal gradients in the matching cost [10] or by training matching costs with samples that include small vertical disparities [36]. Given that misalignment is a global property that can be modeled with just a handful of parameters, recovering these parameters is clearly a cleaner approach. Online stereo calibration and rectification has been researched extensively [4, 8, 9, 14, 15, 17, 19, 25, 28]. Of specific interest are methods that robustly estimate rectification transforms for a single image pair from noisy feature correspondences [5, 47]. Our formulation differs slightly and provides a family of models with increasing number parameters for describing the relative and absolute orientation of the two cameras.

Stereo and monocular depth neural networks. A thorough review of the literature of stereo and monocular depth estimation neural networks is beyond the scope of this paper; see [22] for a recent survey.

Stereo. The first category of papers is to use deep learning to learn the features used for stereo matching to replace handcrafted ones such as census [44]. One notable paper in this category is MC-CNN [36]. However, in recent years, the community has shifted to study end-to-end deep learning stereo. There are three types of architectures: 2D CNNs, 3D CNNs, and RNNs. 2D CNNs, e.g., [34, 42], are typically faster than 3D CNNs, such as [2, 3]. More recently, RNN methods, e.g., RAFT-stereo [18] and CREstereo [16] have proven to show state-of-the-art performance [28], but they remain impractical to run on device. More recent works [31, 34, 43] try to make stereo networks run faster on device. There has also been recent interest in stereo ro-

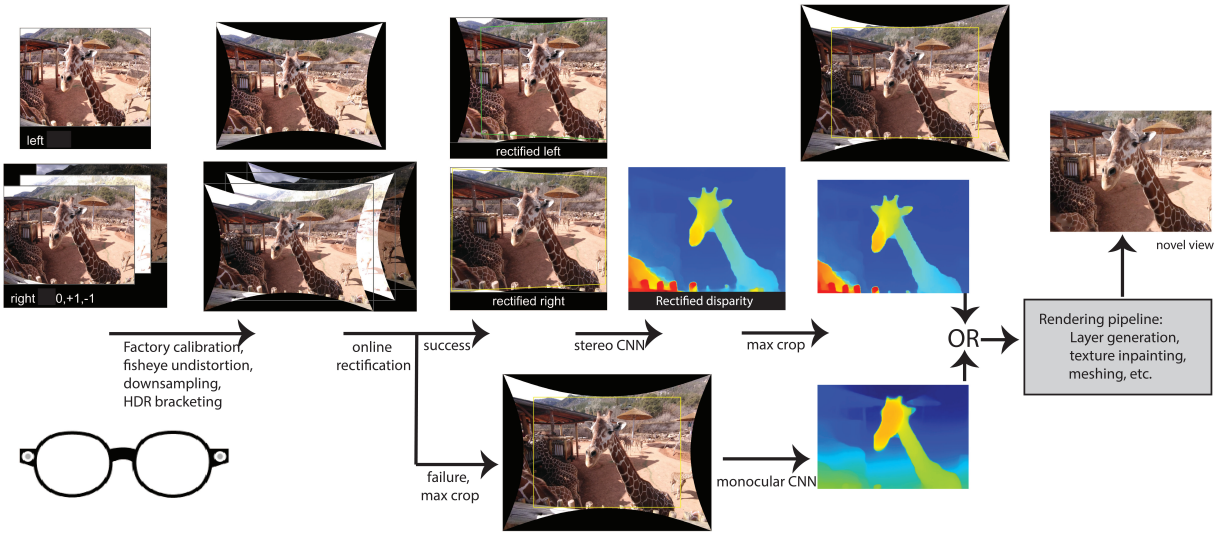

Figure 2. **Overview of the depth system.** The user captures a scene with smart glasses and saves the photo in their phone album. The HDR-bracketed right image is displayed in user's photo album. On the phone, the user can choose to make 3D effects from photos they took. When the user chooses to do so, we utilize the "hidden" left image to estimate stereo depth. The left-right image pair first goes through several pre-processing steps and then an online stereo rectification process to align the epipolar scanlines. We then either use a stereo matching network or a monocular depth estimation network depending on the status of rectification to obtain a relative disparity map. The cropped disparity map and the image are then used to render novel views to generate 3D photographic effects.

bustness [26, 37].

Monocular depth. There are supervised and unsupervised (self-supervised) monocular depth estimation papers using neural networks. In the supervised setting, researchers typically train an encoder-decoder network [13, 24] or a vision transformer [23]. The output is typically relative depth/disparity. In the self-supervised setting, rectified stereo pairs are typically used with photometric consistency loss [7], but the photometric consistency depth cue is absent in regions of stereo occlusion. There is also more recent work, such as [40], that combines the strengths of supervised and unsupervised methods.

3. System overview

The overall system is shown in Figure 2. Our smart glasses are equipped with a pair of hardware-synchronized fisheye cameras with resolution 2592×1944. Once the user captures a scene, the image pair is transferred to a mobile phone for further processing. In our system, the right image is used as the reference view for stereo matching and is used to render novel views.

On the user's smartphone, we first apply factory calibration to undistort the fisheye images to rectilinear and run HDR bracketing. We then downsample the images by $2\times$ to reduce computation for the next steps.

We then run our novel robust online calibration and rectification algorithm, described in detail in Section 4. Online calibration is needed since the glasses deform during normal use, and extrinsics can vary significantly, in particular as a function of the user's head size. Extrinsic and intrinsic parameters also change over time, especially with heat. Our algorithm estimates 3-5 orientation parameters as well as relative focal length correction from a set of precise feature correspondences. This allows accurate rectification of the images. Online calibration may become unreliable however, for instance when the left camera is obstructed, or due to an insufficient number of feature matches. In this case, we fall back on using only the right image and obtain depth with a monocular depth estimation network.

To enable identical downstream processing, it is essential that stereo and monocular networks are similar to each other in terms of architecture output format. Therefore, while a typical stereo network can output absolute disparities, here we output *relative disparities* normalized to 0..1 with both networks. Relative disparity is sufficient for our purpose of creating depth-based computational photography effects. Additionally, we share the encoder and decoder of the stereo network and the monocular network, with the stereo network having additional cost volume layers. We also train both networks with shared datasets, where the stereo dataset is created synthetically by rendering the second view using the monocular dataset. We discuss the networks and datasets in detail in Sections 5 and 7.

Once a disparity map is computed, we crop the maximal valid region to preserve the original aspect ratio. Finally, the predicted disparity for the right camera and the corre-

sponding color image is passed to a rendering pipeline to create the final three-dimensional effect.

4. Online rectification

Two-view stereo algorithms commonly assume a rectified image pair as input [29]. For fixed stereo rigs, rectification is possible from factory calibration. Our smart glasses, however, are made from material that can undergo significant bending ($> 10°$). This is further exacerbated by the variety of user's head sizes and changes of focal length with temperature. This almost always results in imperfect calibration parameters for stereo. Imperfect rectification can strongly affect performance, in particular in high-frequency regions and for near-horizontal image features [10, 28].

The goal of our online calibration method is thus to produce an accurately rectified stereo pair, given images with known lens distortion and estimates of camera intrinsics. We do this from a set of precise feature correspondences, computed in the original fisheye images, and converted to rectilinear (pinhole) coordinates. Figure 3 shows the results of applying our method to a real image pair.

The basic idea is to model this rectification problem as the estimation of correcting rotations $\boldsymbol{\omega}_0, \boldsymbol{\omega}_1$, for each of the two cameras, keeping the baseline fixed. This is a variant of the more common formulation of estimating extrinsics in the form of rotation $\mathbf{R}$ and translation $\mathbf{t}$. In both cases we need to estimate 5 parameters, since the absolute scale is unknown. Existing formulations fix the length of the baseline $\|\mathbf{t}\|$ and estimate its direction [28] or simply keep $\mathbf{t}_x$ constant [5, 47]. Here we adopt a simpler, symmetric approach — we keep the world coordinate system aligned with the stereo rig, both before and after re-rectification, define the length of the baseline to be $b = 1$, and instead compute rotational corrections to both cameras. We can hope to recover each camera's pan angles (ω_{y_0}, ω_{y_1}) and roll angles (ω_{z_0}, ω_{z_1}), but only relative pitch $\Delta\omega_x = \omega_{x_1} - \omega_{x_0}$ since absolute pitch (the angle of the rectifying plane rotating about the x axis) is a free parameter.

In addition to these 5 parameters, which determine the extrinsics (relative pose), we also estimate a 6th parameter Δf that models relative scale, i.e., a small change in focal length $f_0/f_1 = 1 + \Delta f$. In practice we find that compensating for relative scale is very important, as focal lengths change with temperature.

4.1. Projection model

A 3D point $P = (X, Y, Z)$ projects to pixel coordinates (u_0, v_0), (u_1, v_1) in the two images. Assuming that radial distortion has been corrected for and that intrinsics (focal lengths f_i and principal points (c_{x_i}, c_{y_i}) are known, we convert to normalized image coordinates $x_i = (u_i - c_{x_i})/f_i$, $y_i = (v_i - c_{y_i})/f_i$.

Figure 3. **Online rectification example.** Left: original image pair with matches overlaid, and scatterplot of match vectors, exhibiting large vertical disparities Δy. Right: image pair and matches after online rectification. More than 90% of $|\Delta y|$ are below 1.0 pixels, and points at infinity are stabilized at $\Delta x = 0$.

Under the above assumptions the cameras are located at $\mathbf{t}_0 = [0\ 0\ 0]^T$ and $\mathbf{t}_1 = [1\ 0\ 0]^T$, and their rotations are $\mathbf{R}_0 = \mathbf{R}(\boldsymbol{\omega}_0)$ and $\mathbf{R}_1 = \mathbf{R}(\boldsymbol{\omega}_1)$. We use a linear approximation for the rotations since we expect the rotational corrections to be small [33]: $\mathbf{R}(\boldsymbol{\omega}) \approx \mathbf{I} + [\boldsymbol{\omega}]_\times$. In normalized image coordinates, point P projects into the left camera at

$$[x_0\ y_0\ 1]^T \sim [\mathbf{I} + [\boldsymbol{\omega}_0]_\times][X\ Y\ Z]^T. \tag{1}$$

Parameterizing by inverse depth (*disparity*) $d = 1/Z$ we can "unproject" the point and project it into the right camera (see the supplementals for the derivation):

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \sim \begin{bmatrix} \\ \mathbf{I} + [\Delta\boldsymbol{\omega}]_\times \\ \\ \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} + d \begin{bmatrix} 1 \\ -\omega_{z_1} \\ \omega_{y_1} \end{bmatrix}, \tag{2}$$

where $\Delta\boldsymbol{\omega} = \boldsymbol{\omega}_1 - \boldsymbol{\omega}_0$ is the relative orientation of the two cameras. We can see that for $d = 0$ (a point at infinity), we can only recover the relative orientation $\Delta\boldsymbol{\omega}$. For closer points ($d < 0$) we also get a constraint for absolute roll and absolute pan, but not absolute pitch, as discussed earlier.

If we use $\Delta x = x_1 - x_0$ as our estimate for d and also introduce a scale correction $(1 + \Delta f)$, we can derive a constraint on $\Delta y = y_1 - y_0$ (see supplementals):

$$\begin{bmatrix} 1{+}y_0y_1 & -x_0y_1 & -x_0 & -\Delta x y_1 & -\Delta x & y_0 \end{bmatrix} \begin{bmatrix} \Delta\omega_x \\ \Delta\omega_y \\ \Delta\omega_z \\ \omega_{y_1} \\ \omega_{z_1} \\ \Delta f \end{bmatrix} = \Delta y. \tag{3}$$

4.2. Rectification algorithm

We use Harris corners as feature candidates and match them across images using a hierarchical subpixel ZSSD feature matcher with left-right consistency check. For each match we collect Eqn. (3) into an over-constrained system,

which we solve using robust least squares, iterating with decreasing inlier thresholds for robustness to outliers.

Furthermore, since our approximation of the true disparity d via measured disparity Δx only holds near the correct relative orientation, we first only solve for $\Delta\boldsymbol{\omega}$ and Δf, whose equations do not depend on d:

$$\begin{bmatrix} 1+y_0\,y_1 & -x_0\,y_1 & -x_0 & y_0 \end{bmatrix} \begin{bmatrix} \Delta\omega_x \\ \Delta\omega_y \\ \Delta\omega_z \\ \Delta f \end{bmatrix} = \Delta y. \qquad (4)$$

In practice, estimating these four parameters is very stable. Once estimated, we can then solve for one or both of the remaining parameters ω_{y_1} and ω_{z_1}, letting us recover absolute pan and roll for both cameras (since $\omega_{y_0} = \omega_{y_1} - \Delta\omega_y$ and $\omega_{z_0} = \omega_{z_1} - \Delta\omega_z$). However, these absolute angles can only be estimated reliably if the observed feature points are distributed over a sufficient depth range. In practice, we find that we usually get satisfactory rectification results for our low-resolution images with the 4-parameter model if we simply split the estimated relative angles evenly between left and right camera.

To declare online rectification successful, we require at least 100 matched feature points and an inlier rate of at least 60% matches with $|\Delta y| \leq 1.0$. In addition we check that the computed rectification angles stay within conservative bounds ($< 5^\circ$ absolute pitch and roll, and $< 22^\circ$ relative pan $|\Delta\omega_y|$). If any of these criteria is not met, we fall back on monocular depth estimation. In practice this happens for roughly 10% of user-taken images, typically for feature-poor indoor scenes or if one of the cameras is obstructed.

We based our rectification design decisions on experiments with 185 challenging test image pairs captured with prototypes, with manually established ground truth. Figure 7 shows a subset. Table 1 lists success and inlier rates for our 3 and 4-parameter models, and demonstrates that estimating Δf is important in practice.

Rectification model	Success rate	Median inlier rate	Error Δf
3-param: $\Delta\boldsymbol{\omega}$	142/185= 77%	80% (N=142)	0.083%
4-param: $\Delta\boldsymbol{\omega}$, Δf	**157**/185= **85%**	**85%** (N=157)	**0.024%**

Table 1. Performance increase due to estimating relative scale Δf.

5. Co-design of monocular and stereo networks

We describe a novel method to co-design stereo and monocular depth networks with the goal of having consistent output format (relative disparity), being light-weight, and being as accurate as possible.

5.1. Stereo network

We design a stereo network with these components inspired by both classical and deep stereo methods [33]:

1) An encoder to extract multi-resolution features f_0^l and feature f_1^l independently from the input stereo pair, where $l = 1 \ldots L$, for L-level feature pyramids.

2) Three-dimensional cost volumes that compare left and right feature distances via the cosine distance

$$C^l(x, y, d) = dot(f_0^l(x, y), f_1^l(x - d, y)), \qquad (5)$$

for $d \in [0, \ldots, d_{\text{max}}]$.

3) A number of middle layers that take the cost volumes and the image features of the reference image as inputs, and aggregate the disparity information. Because the middle layers obtain information from cost volumes and reference images directly, they can better leverage monocular depth cues when stereo matching cues are weak (e.g., in textureless regions) or absent (e.g., in half-occluded regions).

4) A coarse-to-fine decoder to predict an output disparity map. The output disparity map has the same resolution as the input right image. Each decoder module combines the output of the lower-resolution decoder module and the output of the same-resolution middle layers. Figure 4 shows the architecture diagram of our stereo network.

5.2. Monocular network

We design a monocular depth estimation network with three components: 1) An encoder to extract multi-resolution image features $f^{l=1\ldots L}$; 2) middle layers to aggregate the depth information; and 3) a coarse-to-fine decoder to predict a disparity map. Figure 5 shows the architecture diagram.

5.3. Shared network components

All three components of the monocular network also exist in the stereo network. The stereo network has additional cost volume modules to explicitly compare the similarity between the left and right local features. Therefore, we propose to share the exact same encoder, middle layers and decoder between the two networks. For stereo, the encoder runs twice to extract left and right image features independently (i.e., a Siamese encoder).

To train such networks with shared components, we train the monocular depth network first. Then we initialize the stereo network with the trained weights from the monocular network, and fine-tune with stereo training data. Both networks are trained with the same loss

$$L(d_*, d_{gt}) = smooth\_L1(d_*, d_{gt}) + \lambda \sum_{l=1}^{L} smooth\_L1(grad(d_*^l), grad(d_{gt}^l)), \qquad (6)$$

where

$$smooth\_L1(x, y) = \begin{cases} 0.5\,(x-y)^2, & \text{if } |x-y| < 1.0 \\ |x-y| - 0.5, & \text{otherwise.} \end{cases}$$

We use the median aligner proposed in MiDaS [24] to obtain aligned output disparity d_* with ground truth d_{gt}.

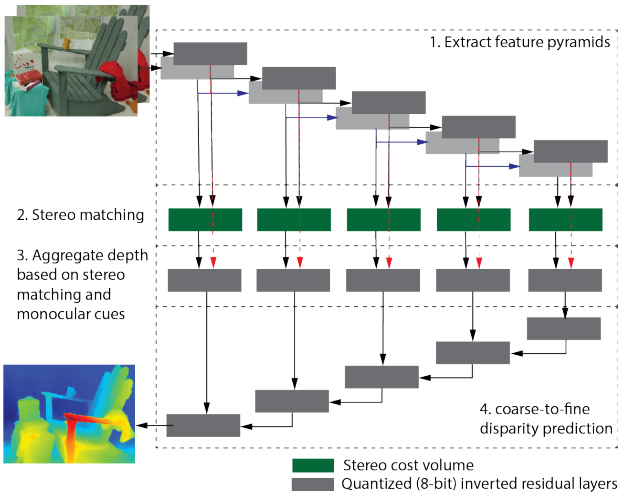

Figure 4. **Stereo network architecture.** We extract a feature pyramid for each input image, and then build 3D cost volumes. The 3D cost volumes and the features from the reference image are passed to middle layers to aggregate the disparity information. Finally, we design a coarse-to-fine decoder to compute a final disparity map. We use inverted residual blocks introduced in MobileNet v2 [27] and quantize to 8 bits to improve efficiency.

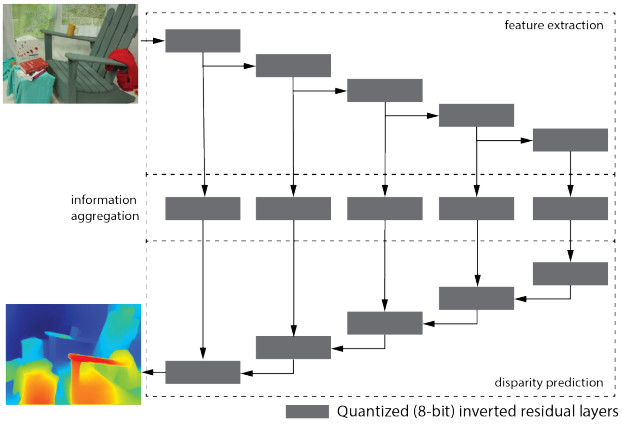

Figure 5. **Monocular depth network architecture.** We design an encoder-decoder architecture for our monocular depth estimation network. Similar to our stereo network, we use inverted residual blocks and 8-bit quantization to improve efficiency.

This design enables both the mono and stereo networks to output relative disparity maps and maximizes the consistency in predictions.

To improve efficiency, we use inverted residual modules proposed in MobileNetV2 [27] for all layers and quantize both weights and activation to 8 bits. Only the output layer and the layers that compute the cost volumes are not quantized. We keep the output layer 32 bits to get sub-pixel resolution and avoid quantization artifacts in the 3D effects.

5.4. Novel training datasets

We derive the stereo data entirely from an internal 4M monocular dataset captured with iPhone. Using the embedded depth map from the monocular dataset, we render the second view according to the known focal length and baseline of our smart glasses. The occluded regions are also inpainted. We use the LDI-based rendering method discussed in [13] which we will briefly discuss in Section 6. This makes our stereo dataset much more diverse than any existing training datasets. Additionally, both the stereo and the monocular networks can be trained with the same data to make sure their output and behavior are similar, which is one of the main design goals of our depth sensing system.

A similar technique was also described by Watson et al. [39], but they use a monocular depth estimation network such as MiDaS [24] to generate the depth.

To make the training data photo-realistic and challenging for the stereo network, we perform data augmentations for our rendered stereo dataset including brightness, contrast, hue, saturation, jpeg compression, reflection augmentation and image border augmentation. Reflection augmentation and image border augmentation are discussed in more details below. Other augmentations are standard.

Certain types of surfaces can reflect light, causing bright specular highlights in the images. This creates a difficult scenario for stereo matching because there are two textures at conflicting depths: one of the reflected light source and one of the reflecting surface. Some existing work models reflections in stereo explicitly [35]. We instead train our stereo network to ignore reflections of specular highlights via simple augmentation. We model a specular highlight as additive light following a Gaussian distribution with random size (σ_x, σ_y) and location (μ_x, μ_y) in the left image. We assume the light source is at random disparity d_l and render its corresponding highlight in the right image at $(\mu_x - d_l, \mu_y)$.

Finally, we also randomly add oval-shaped black image borders to mimic the invalid regions caused by warping and rectification in our depth pipeline (see Figure 2).

6. Novel view synthesis

We use the same LDI (layered depth image) based method described in [13] to create the stereo training dataset as well as to create our 3D effects. Here we briefly review this method. Given a image/depth pair, we lift it to a layered depth image, hallucinate the occluded geometry by doing LDI inpainting, and finally convert it to a textured mesh. For stereo dataset creation, we use the monocular ground-truth depth and the color image to create such a mesh and render the second view. For the 3D effect, we use the prediction from our stereo system and a pre-defined trajectory

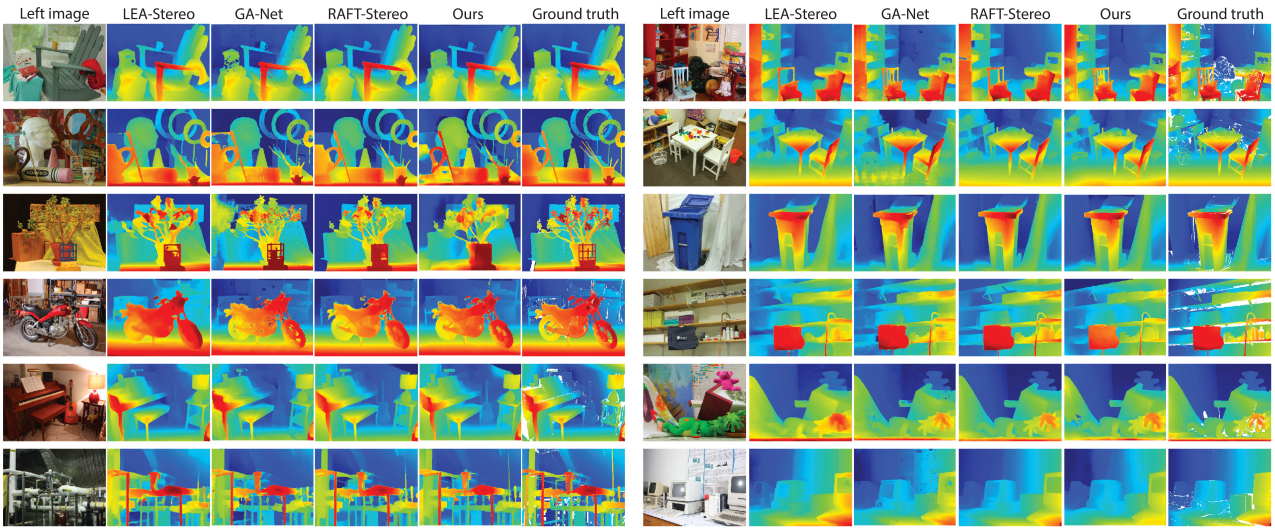

Figure 6. **Train on Sceneflow and test on Middlebury.** We achieve on par performance with SotA methods despite our model being much smaller and quantized. All methods are tested on 384×288 resolution.

	Train data	Quantized	Latency (s)	AbsRel↓	$\delta > 1.25$↑	$\delta > 1.25^2$↑	$\delta > 1.25^3$↑
RAFT-stereo [18]	Sceneflow	no	224.75	0.075	0.923	**0.973**	**0.987**
LEA-stereo [3]	Sceneflow	no	3.66	0.096	0.902	0.959	0.975
GA-Net [45]	Sceneflow	no	7.73	0.128	0.893	0.944	0.961
MobileStereoNet (2D) [31]	Sceneflow	no	12.30	0.180	0.818	0.920	0.956
MobileStereoNet (3D) [31]	Sceneflow	no	18.96	0.139	0.876	0.933	0.963
Argos (Ours)	Sceneflow	yes	0.167	0.107	0.897	0.953	0.971
Argos (Ours)	internal	yes	0.167	**0.075**	**0.931**	0.967	0.980
Tiefenrausch (mono) [13]	internal	yes	0.140	0.253	0.595	0.813	0.905

Table 2. **Middlebury results.** We report results on the 15 stereo pairs of the Middlebury 2014 training set. We use the pretrained PyTorch checkpoints in GitHub repos for baseline methods. For fair comparison, latency is measured at 384×288 resolution on a server without using GPU (Intel Xeon Gold 6138 CPU @ 2.00GHz). Note that GPUs can yield significant speedup for some methods (e.g. 1.04s for RAFT-stereo on a Tesla V100 GPU). Torchscripting and quantization may also bring speedup for some comparison methods.

to generate a smooth video of novel views.

7. Experiments

Neural network implementation details. We use the efficient monocular network *Tiefenrausch* presented in [13]. Details on network architecture can be found in the supplemental of [13]. Our stereo network, *Argos*, shares the same encoder, middle layers, and decoder with Tiefenrausch, but with added cost volume modules (Equation 5), which are is parameter free. They simply take feature vectors of patches of the same epipolar scanline and compare patch similarity. We add one more layer after each cost volume module with input size $d_{\max} + C(f_0^i)$ and output size $C(f_0^i)$, the dimension of the feature vector of the reference image.

For the production model, we first re-train Tiefenrausch with our 4M internal iPhone dataset. As in [13], we align the predicted disparity map with the "ground truth" disparity map with the MiDaS median aligner [24] to output relative disparity. When computing the loss (Eqn. 6), we compute the gradient loss at $L = 5$ levels by simply doing 2×2 subsampling the prediction and the ground truth at coarser levels. We then initialize the stereo network, Argos, with the Tiefenrausch weights. All but one layer (the added layer after cost volumes) can be initialized with the Tiefenrausch weights. We then finetune Argos with the rendered internal iPhone stereo dataset. During finetuning, we use Adam optimizer with base learning rate $4e-4$ and batch size of 24. We train a total of 640k iterations, and use quantization-aware-training (QAT) in PyTorch [6, 32], which is essential to achieve high accuracy. PTQ (Post-training quantization) leads to significant accuracy drop. We start QAT at the 2000^{th} training iteration, and use the FBGEMM backend [12]. All input images are resized to 384×288 resolution for both training and evaluation, as this is the output disparity resolution used in our system. To fairly com-

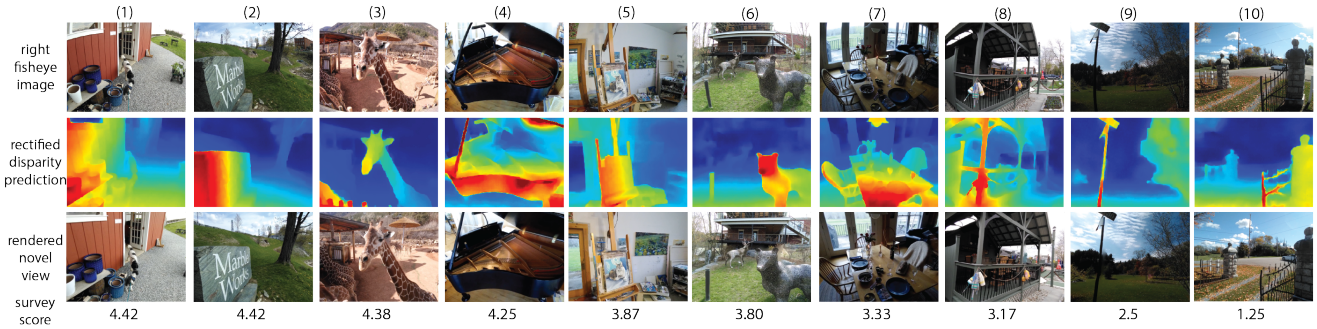

Figure 7. **Survey results.** 10 scenes captured from our smart glasses with original right images, rectified disparity maps, rendered novel views, and survey scores for rendered novel-view videos, with 1=poor and 5=excellent. Depth errors in thin structures often cause significant artifacts in rendered views. Depth errors in background or textureless regions typically do not lead to very noticeable artifacts.

pare our model with SotA models, we train another version of Argos with only the Sceneflow [20] datasets (FlyingThings3D, Driving and Monkaa), and compare our results with other SotA models that are also trained on Sceneflow.

Run time. We ran benchmarks of our pipeline on a Samsung Galaxy S8 CPU. The rectification pipeline takes 300-400ms, and the stereo network takes around 965ms. Other parts of the pipeline in total takes much lower latency than these two steps. Our model is optimized for mobile CPU, but converting SotA models to a mobile-friendly format is non-trivial and not very meaningful since they are not designed for mobile. As a compromise, we compare run time of all models on a computer server machine, with Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz, shown in Table 2.

Middlebury Results. Table 2 and Figure 6 show the quantitative comparison of our method with several SotA stereo methods on the Middlebury 2014 dataset [28]. When trained with Sceneflow dataset, our method achieves on-par performance with SotA methods despite being several orders of magnitude faster. Training with our rendered internal stereo dataset further improves our performance, and achieves the best Absolute Relative error among all tested models. We also include the results of Tiefenrausch [13] retrained on our internal dataset. Comparing with Tiefenrausch, our stereo model, Argos, dramatically reduces absolute relative error from 0.253 to 0.075, which shows the effectiveness of our novel design to render a stereo dataset from a monocular dataset to train our model.

Note is that our goal is to design an end-to-end depth system that works robustly in all scenarios, rather than to achieve the best performance on academic benchmarks like Middlebury [28]. Some of our design choices, e.g., using reflection augmentation, can cause the accuracy to drop on Middlebury, which is carefully curated and do not have any scenes with obvious reflections. But such artifacts are common in real life. We are also one of the first to present a quantized 8-bit stereo model whereas all comparison models use 32-bit weights and activation. All of the above give us disadvantages. Nevertheless, we achieve on-par performance with SotA methods, and are much faster.

3D photo quality survey. To better understand the quality of our depth system, we conducted a survey with 24 participants. We capture a number of scenes from the smart glasses and then render novel view videos with depth maps from our depth system. We ask the participates to rate the quality of the rendered video from 1 to 5, with 5 being best (no artifacts) and 1 worst (major artifacts). Out of all responses, the average scores were 3.44 for using stereo and 2.96 for using mono depth. Figure 7 shows 10 survey scenes, covering both success and failure cases. One interesting observation is that the depth map quality sometimes does not directly correlate with the quality of the rendered novel view videos. For example, the depth map of scene (4) is poor, but there are few artifacts in the rendered view, and the users gave high scores. In contrast, the depth map of (10) is mostly correct except for the thin structures of the fence. However, the rendered video has jarring artifacts near the fence, causing poor user ratings in the survey. If we use standard stereo metrics such as the percentage of bad pixels, (10) likely has lower error than (4). This highlights that solely comparing methods using standard metrics is not enough to evaluate a stereo method's performance in practice. Selected videos are attached in the supplement.

8. Conclusion

We present the system design of an end-to-end stereo depth sensing system that runs efficiently on smartphones. We describe a novel online rectification algorithm, a novel co-design of monocular and stereo disparity networks, and a novel method to derive large stereo datasets from monocular datasets. We present an 8-bit quantized stereo model that has competitive performance on standard stereo benchmarks when compared against state-of-the-art methods.

Acknowledgement: We thank Rick Szeliski for his help in developing the online rectification model.

References

[1] Jonathan Barron, Andrew Adams, YiChang Shih, and Carlos Hernandez. Fast bilateral-space stereo for synthetic defocus. In *CVPR*, 2015. 2

[2] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, pages 5410–5418, 2018. 2

[3] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Hongdong Li, Tom Drummond, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. *NeurIPS*, 33, 2020. 1, 2, 7

[4] Thao Dang, Christian Hoffmann, and Christoph Stiller. Continuous stereo self-calibration by camera parameter tracking. *IEEE TIP*, 18(7):1536–1550, 2009. 2

[5] Mihail Georgiev, Atanas Gotchev, and Miska Hannuksela. A fast and accurate re-calibration technique for misaligned stereo cameras. In *ICIP*, 2013. 2, 4

[6] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021. 7

[7] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *CVPR*, pages 3828–3838, 2019. 3

[8] Peter Hansen, Hatem Alismail, Peter Rander, and Brett Browning. Online continuous stereo extrinsic parameter estimation. In *CVPR*, pages 1059–1066, 2012. 2

[9] Richard Hartley. Theory and practice of projective rectification. *IJCV*, 35(2):115–127, 1999. 2

[10] Heiko Hirschmüller and Stefan Gehrig. Stereo matching in the presence of sub-pixel calibration errors. In *CVPR*, pages 437–444, 2009. 2, 4

[11] Per Karlsson and Lucy Yu. The technology behind Cinematic photos. In *Google AI Blog*. https://ai.googleblog.com/2021/02/the-technology-behind-cinematic-photos.html, 2021. 2

[12] Daya Khudia, Jianyu Huang, Protonu Basu, Summer Deng, Haixin Liu, Jongsoo Park, and Mikhail Smelyanskiy. Fbgemm: Enabling high-performance low-precision deep learning inference. *arXiv preprint arXiv:2101.05615*, 2021. 7

[13] Johannes Kopf, Kevin Matzen, Suhib Alsisan, Ocean Quigley, Francis Ge, Yangming Chong, Josh Patterson, Jan-Michael Frahm, Shu Wu, Matthew Yu, Peizhao Zhang, Zijian He, Peter Vajda, Ayush Saraf, and Michael F. Cohen. One shot 3D photography. *ACM TOG*, 39(4):76–1, 2020. 2, 3, 6, 7, 8

[14] Andreas Kuhn, Lukas Roth, Jan-Michael Frahm, and Helmut Mayer. Improvement of extrinsic parameters from a single stereo pair. In *WACV*, pages 1011–1019, 2018. 2

[15] Avinash Kumar, Manjula Gururaj, Kalpana Seshadrinathan, and Ramkumar Narayanswamy. Multi-capture dynamic calibration of multi-camera systems. In *CVPR Workshops*, pages 1843–1851, 2018. 2

[16] Jiankun Li, Peisen Wang, Pengfei Xiong, Tao Cai, Ziwei Yan, Lei Yang, Jiangyu Liu, Haoqiang Fan, and Shuaicheng Liu. Practical stereo matching via cascaded recurrent network with adaptive correlation. In *CVPR*, pages 16263–16272, 2022. 2

[17] Yonggen Ling and Shaojie Shen. High-precision online markerless stereo extrinsic calibration. In *IROS*, 2016. 2

[18] Lahav Lipson, Zachary Teed, and Jia Deng. RAFT-Stereo: Multilevel recurrent field transforms for stereo matching. In *3DV*, pages 218–227, 2021. 1, 2, 7

[19] John Mallon and Paul Whelan. Projective rectification from the fundamental matrix. *IVC*, 23(7):643–650, 2005. 2

[20] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, pages 4040–4048, 2016. 8

[21] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, pages 3061–3070, 2015. 2

[22] Matteo Poggi, Fabio Tosi, Konstantinos Batsos, Philippos Mordohai, and Stefano Mattoccia. On the synergies between machine learning and binocular stereo for depth estimation from images: A survey. *IEEE TPAMI*, 4432(9):5314–5334, 2022. 2

[23] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *ICCV*, 2021. 3

[24] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE TPAMI*, 44(3):1623–1637, 2022. 3, 5, 6, 7

[25] Eike Rehder, Christian Kinzig, Philipp Bender, and Martin Lauer. Online stereo camera calibration from scratch. In *IEEE Intelligent Vehicles Symposium*, pages 1694–1699, 2017. 2

[26] Robust Vision Challenge (RVC 2022). http://www.robust-vision.net/, 2022. 3

[27] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018. 6

[28] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nesic, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *GCPR*, volume 8753, pages 31–42, 2014. 2, 4, 8

[29] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1-3):7–42, 2002. 4

[30] Thomas Schöps, Johannes Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *CVPR*, 2017. 2

[31] Faranak Shamsafar, Samuel Woerz, Rafia Rahim, and Andreas Zell. MobileStereoNet: Towards lightweight deep networks for stereo matching. In *WACV*, pages 2417–2426, 2022. 2, 7

[32] Suraj Subramanian, Mark Saroufim, and Jerry Zhang. Practical quantization in PyTorch. In *https://pytorch.org/blog/quantization-in-practice/*, 2022. 7

[33] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2nd edition, 2020. 4, 5

[34] Vladimir Tankovich, Christian Häne, Yinda Zhang, Adarsh Kowdle, Sean Fanello, and Sofien Bouaziz. HITNet: Hierarchical iterative tile refinement network for real-time stereo matching. In *CVPR*, pages 14362–14372, 2021. 2

[35] Yanghai Tsin, Sing Bing Kang, and Richard Szeliski. Stereo matching with reflections and translucency. In *CVPR*, pages 702–709, 2003. 6

[36] Jure Žbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *JMLR*, 17(65):1–32, 2016. 2

[37] Jialiang Wang, Varun Jampani, Deqing Sun, Charles Loop, Stan Birchfield, and Jan Kautz. Improving deep stereo network generalization with geometric priors. *arXiv preprint arXiv:2008.11098*, 2020. 3

[38] Jialiang Wang and Todd Zickler. Local detection of stereo occlusion boundaries. In *CVPR*, pages 3818–3827, 2019. 2

[39] Jamie Watson, Oisin Mac Aodha, Daniyar Turmukhambetov, Gabriel Brostow, and Michael Firman. Learning stereo from single images. In *ECCV*, pages 722–740, 2020. 6

[40] Cho-Ying Wu, Jialiang Wang, Michael Hall, Ulrich Neumann, and Shuochen Su. Toward practical monocular indoor depth estimation. In *CVPR*, pages 3814–3824, 2022. 3

[41] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. In *CVPR*, pages 5515–5524, 2019. 2

[42] Zhichao Yin, Trevor Darrell, and Fisher Yu. Hierarchical discrete distribution decomposition for match density estimation. In *CVPR*, pages 6044–6053, 2019. 2

[43] Hang Yu, Ke Wang, Lun Zhou, and Zhen Wang. MStereo-Net: A lightweight stereo matching network using MobileNet. In *AICIT*, 2022. 2

[44] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV*, pages 151–158, 1994. 2

[45] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip Torr. GA-Net: Guided aggregation net for end-to-end stereo matching. In *CVPR*, pages 185–194, 2019. 1, 2, 7

[46] Yinda Zhang, Neal Wadhwa, Sergio Orts-Escolano, Christian Häne, Sean Fanello, and Rahul Garg. Du^{2}Net: Learning depth estimation from dual-cameras and dual-pixels. In *ECCV*, pages 582–598, 2020. 2

[47] Frederik Zilly, Marcus Müller, Peter Eisert, and Peter Kauff. Joint estimation of epipolar geometry and rectification parameters using point correspondences for stereoscopic TV sequences. In *3DPVT*, 2010. 2, 4

9. Appendix

9.1. Derivation of projection model

Below is an expanded version of Section 4.1 with the full derivation of the projection model.

A 3D point $P = (X, Y, Z)$ projects to pixel coordinates (u_0, v_0), (u_1, v_1) in the two images. Assuming that radial distortion has been corrected for and that intrinsics (focal lengths f_i and principal points c_{x_i}, c_{y_i}) are known, we convert to normalized image coordinates $x_i = (u_i - c_{x_i})/f_i$, $y_i = (v_i - c_{y_i})/f_i$.

Under the above assumptions the cameras are located at $\mathbf{t}_0 = [0\ 0\ 0]^T$ and $\mathbf{t}_1 = [1\ 0\ 0]^T$, and their rotations are $\mathbf{R}_0 = \mathbf{R}(\boldsymbol{\omega}_0)$ and $\mathbf{R}_1 = \mathbf{R}(\boldsymbol{\omega}_1)$. We use a linear approximation for the rotations since we expect the rotational corrections to be small:

$$\mathbf{R}(\boldsymbol{\omega}) \approx \mathbf{I} + [\boldsymbol{\omega}]_\times = \begin{bmatrix} 1 & -\omega_z & \omega_y \\ \omega_z & 1 & -\omega_x \\ -\omega_y & \omega_x & 1 \end{bmatrix}. \quad (7)$$

In normalized image coordinates, point P projects into the left camera at

$$\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \sim \begin{bmatrix} \mathbf{I} + [\boldsymbol{\omega}_0]_\times \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \quad (8)$$

Parametrizing by inverse depth (*disparity*) $d = 1/Z$ we can "unproject" the point

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \sim \begin{bmatrix} X/Z \\ Y/Z \\ 1 \\ 1/Z \end{bmatrix} \sim \left[\begin{array}{c|c} \mathbf{I} - [\boldsymbol{\omega}_0]_\times & \mathbf{0} \\ \hline \mathbf{0} & 1 \end{array}\right] \begin{bmatrix} x_0 \\ y_0 \\ 1 \\ d \end{bmatrix}. \quad (9)$$

Projecting it into the right camera, we get

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \sim \left[\mathbf{I}+[\boldsymbol{\omega}_1]_\times \,\middle|\, \mathbf{0}\right] \left[\begin{array}{c|c} \mathbf{I} & \begin{matrix}1\\0\\0\end{matrix} \\ \hline \mathbf{0} & 1 \end{array}\right] \left[\begin{array}{c|c} \mathbf{I}-[\boldsymbol{\omega}_0]_\times & \mathbf{0} \\ \hline \mathbf{0} & 1 \end{array}\right] \begin{bmatrix} x_0 \\ y_0 \\ 1 \\ d \end{bmatrix} \quad (10)$$

$$\approx \begin{bmatrix} \mathbf{I}+[\boldsymbol{\omega}_1-\boldsymbol{\omega}_0]_\times \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} + \begin{bmatrix} \mathbf{I}+[\boldsymbol{\omega}_1]_\times \end{bmatrix} \begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix} \quad (11)$$

$$= \begin{bmatrix} \mathbf{I}+[\Delta\boldsymbol{\omega}]_\times \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} + d \begin{bmatrix} 1 \\ -\omega_{z_1} \\ \omega_{y_1} \end{bmatrix}, \quad (12)$$

where $\Delta\boldsymbol{\omega} = \boldsymbol{\omega}_1 - \boldsymbol{\omega}_0$ is the relative orientation of the two cameras. We can see that for $d = 0$ (i.e., a point at infinity), we can only recover the relative orientation $\Delta\boldsymbol{\omega}$. For closer points ($d < 0$) we also get a constraint for absolute roll and absolute pan, but not absolute pitch, as discussed earlier.

If we use $\Delta x = x_1 - x_0$ as our estimate for d and also introduce a scale correction $(1 + \Delta f)$, we have

$$\begin{bmatrix} x_1 \\ y_1 \\ 1{+}\Delta f \end{bmatrix} \sim \begin{bmatrix} 1 & \Delta\omega_z & -\Delta\omega_y \\ -\Delta\omega_z & 1 & \Delta\omega_x \\ \Delta\omega_y & -\Delta\omega_x & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} + \Delta x \begin{bmatrix} 1 \\ -\omega_{z_1} \\ \omega_{y_1} \end{bmatrix} \quad (13)$$

$$= \begin{bmatrix} x_0{+}\Delta\omega_z y_0{-}\Delta\omega_y{+}\Delta x \\ y_0{-}\Delta\omega_z x_0{+}\Delta\omega_x{-}\omega_{z_1}\Delta x \\ 1{+}\Delta\omega_y x_0{-}\Delta\omega_x y_0{+}\omega_{y_1}\Delta x \end{bmatrix}. \quad (14)$$

Cross-multiplying and dropping higher-order terms we get two equations in the 6 unknowns, where $\Delta y = y_1 - y_0$:

$$\begin{bmatrix} -y_0 x_1 & 1{+}x_0 x_1 & -y_0 & \Delta x x_1 & 0 & -x_1 \\ 1{+}y_0 y_1 & -x_0 y_1 & -x_0 & -\Delta x y_1 & -\Delta x & y_0 \end{bmatrix} \begin{bmatrix} \Delta\omega_x \\ \Delta\omega_y \\ \Delta\omega_z \\ \omega_{y_1} \\ \omega_{z_1} \\ \Delta f \end{bmatrix} = \begin{bmatrix} 0 \\ \Delta y \end{bmatrix}. \quad (15)$$

Of these, only the second equation gives us a constraint relating y_0 and y_1. We can collect these equations, one for each matched feature point, and solve the over-constrained system using robust least squares.