FlexSight - A Flexible and Accurate System for Object Detection and Localization for Industrial Robots

Daniele Evangelista
Dep. of Information Engineering (DEI)
University of Padova, Italy
evangelista@dei.unipd.it

Marco Imperoli
Dep. of Computer, Control, and Management Engineering (DIAG)
Sapienza University of Rome, Italy
imperoli@diag.uniroma1.it

Emanuele Menegatti
Dep. of Information Engineering (DEI)
University of Padova, Italy
emg@dei.unipd.it

Alberto Pretto
FlexSight S.r.l.
Padova, Italy
alberto.pretto@flexsight.eu

***Abstract*—We present a novel smart camera - the FlexSight C1 - designed to enable an industrial robot to detect and localize several types of objects and parts in an accurate and reliable way. The C1 integrates all the sensors and a powerful mini computer with a complete Operating System running robust 3D reconstruction and object localization algorithms on-board, so it can be directly connected to the robot that is guided directly by the device during the production cycle without any external computers in the loop.**
In this paper, we describe the FlexSight C1 hardware configuration along with the algorithms designed to face the model based localization problem of textureless objects, namely: (1) an improved version of the PatchMatch Stereo matching algorithm for depth estimation; (2) an object detection pipeline based on deep transfer learning with synthetic data. All the presented algorithms have been tested on publicly available datasets, showing effective results and improved runtime performance.

***Index Terms*—Structured light cameras, Object Detection, Stereo Matching, Deep Learning, Texture-less Objects**

I. Introduction

One of the key challenges in many industrial robotics applications is the capability to automatically identify and locate various types of objects, in such a way that a robot can inspect, grasp or manipulate them accurately and reliably. Depending on the application, objects can be regularly disposed in trays, but often they are randomly placed inside bins, or over tables, pallets or conveyor belts. A reliable perception systems is thus required to identify and precisely locate the objects in 3D, and to guide robots during the manipulation tasks.

Industrial machine vision systems currently work with 3D sensors, sometimes coupled with a color or gray-level cameras. Such sensors can be classified between *passive stereo* cameras (e.g., the KUKA_3D Perception sensor [1]), structured *Structured Light* (SL) cameras (e.g., the Photoneo PhoXi 3D Scanner [2]), and *Time-of-Flight* (ToF) cameras (e.g., the Basler ToF Camera [3]).

Differently from the systems presented above, we present here the FlexSight C1 embedded device (Fig. 1, see Sec. III) that integrates 2D/3D acquisition, data processing and data

This work was supported by the European Commission under 601116-ECHORD++ (FlexSight experiment).

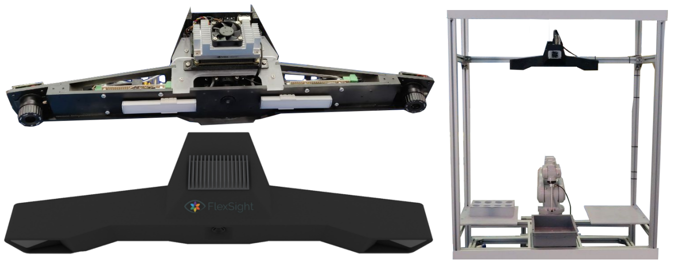
Fig. 1: The FlexSight C1 Sensor: (top left) The prototype without the external case; (bottom left) A render of the finalized sensor; (right) The FlexSight C1 installed on a real robotic cell for random bin-picking applications.

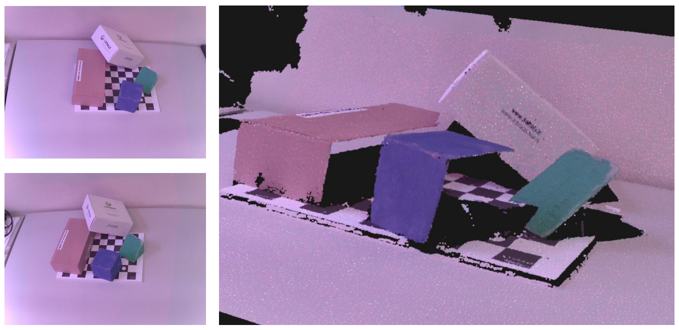
Fig. 2: An example of active stereo 3D reconstruction: (left) The input left and right RGB images; (right) The resulting colored point cloud. The slight red brightness originates from the projected red pseudo-random pattern, that enables to obtain an accurate dense 3D reconstruction also for untextured surfaces.

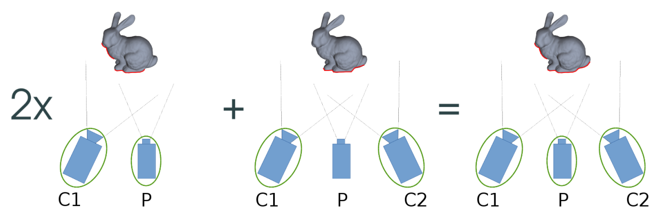

Fig. 3: The FlexSight C1 sensor embeds multiple types of vision technologies by using two color cameras (C_1 and C_2) and a visible pseudo-random pattern (P): two active stereo systems ($C_1 + P$ and $C_2 + P$) and one passive stereo system ($C_1 + C_2$). The green ovals highlight which of the sensors are calibrated between them.

interpretation capabilities in the same sensor. In particular, two high resolution color cameras provide the system with passive stereo capabilities, and a pseudo-random pattern projector mounted at the center of the cameras baseline enable both active stereo and structured light capabilities (Fig. 3). The proposed device also integrates a CPU and a powerful Graphical Processing Unit (GPU) specifically designed to run complex and high demanding processing algorithms (e.g. Deep Learning inference) and a complete Linux based Operating System.

A high resolution depth map is internally generated by leveraging the C1 active stereo capabilities and a novel pyramidal stereo matcher (see Sec. IV-A) built upon the Patch Match Stereo algorithm [4]. From the extracted depth map it is possible to generate accurate colored 3D point clouds of the scene even in presence of textureless objects or untextured surfaces Fig. 2.

To enable real-time model based object detection and localization capabilities, we propose to employ deep neural network based object detectors trained by using synthetically generated data (see Sec. IV-B). This process drastically decreases the time needed for collecting data, and does not require any human intervention for annotating the data.

Quantitative evaluations of the proposed solutions on publicly available datasets are reported in Sec. V, where we employ the Kitti Stereo and the Middlebury Stereo datasets to evaluate the depth estimation, and the T-Less dataset to evaluate the object detection for texture-less objects. All the proposed algorithms have been implemented on the real embedded system.

II. Related Works

A. Stereo Matching

Depth estimation from stereo is one of the most active topics in computer vision of the last 30 years. Given two rectified images, the problem is to find for each pixel in the reference image the corresponding point in the second image. Rectification reduces the correspondences' search along the same scanline. As described in [5], the main steps of stereo algorithms are: matching cost computation, cost aggregation, disparity optimization followed by a disparity refinement step. Methods can be categorized in local [4], [6]–[8], global [9]–[12] or semiglobal [13], [14], depending on the techniques used to solve each step of the pipeline.

Recent works exploit the framework of PatchMatch Stereo [4], [10]. These methods exploit alternatively a random depth generation procedure and the propagation of depth, resulting in a total runtime cost of $O(W \log L)$, where W is the window size used to compute the matching cost between patches and L the number of searched disparities. The method proposed in [8], instead, strongly relies on superpixels, removing the linear dependency on on both the window size and label space. However, the superpixels's estimation requires a high computational time.

The active stereo problem has been recently addressed by exploiting efficient learning-based solutions [15] [16] [17] [18]. Recent deep learning based methods, among the others [19] [20], provide very accurate results. However, these techniques usually don't generalize well to different contexts and require a fine-tuning of the CNN. Others [21] [22] try to predict depth from a single image, but in practice are limited to very specific scenes.

B. Texture-less Object Detection

Object detection in images has been addressed traditionally in two ways: methods based on sliding window as Deformable Part Model [23]; classification of regions produced with region proposal algorithms, e.g. Selective Search [24]. Methods based on region proposals have become recently prominent thanks to the growing interest in Convolutional Neural Networks (CNNs). R-CNN [25] has been the first deep neural network trained for extracting features from region proposals using convolutional networks. This approach has been further improved in Faster R-CNN [26] where the selective search region proposal algorithm has been replaced with a Region Proposal Network (RPN, introduced in [27] and [28]) and the complete deep network is trained end-to-end for extracting the proposals and performing classification on the object's bounding box retrieved performing regression.

In the last years, object detectors like Single Shot Detector [29] and YOLO [30] improved the quality and speed of the detection by simultaneously producing a score for each object category in each predicted box and then classifying them. In this way, the deep network is easier to train, faster, and ready to be integrated into other tasks.

All the aforementioned methods require huge amount of annotated data for training, most of the time taken by manually annotating thousand of images of the real objects. To overcome this high effort task, different solution based on synthetic data has been proposed. [31] trained large and complex deep convolutional networks with pure synthetic images performing transfer learning from a pre-trained network. Although this approach has been proved to be very promising in terms of minimal human intervention while generating the data, it suffers from the so called *domain gap* problem: real and synthetic images domains are too separated. To recover this gap and increase the performance of the detection, different solutions has been recently proposed, e.g. [32] learns 3D features from synthetic depth images and then tries to map the feature vectors to the RGB domain optimizing a specific loss function. [33] uses CAD models to produce synthetic depth data with domain-relevant background and randomized augmentation to train an end-to-end, multi-task network to detect and estimate poses of texture-less objects in real-world depth images.

III. Hardware

The proposed system integrates all the sensors and processing units inside a compact and robust case (Fig. 1). As processing unit, we employ a NVIDIA Jetson TX1 module, running Ubuntu 16.04 Linux as Operating System, that includes in a small form factor both a fast ARM8 CPU with 4 GB of RAM and a powerful NVIDIA GPU with 256 CUDA cores and 4 GB of dedicated RAM. Following the configuration depicted in Fig. 3, as imaging sensors we use two 12 megapixels color cameras connected to the processing unit through the MIPI CSI-2 interface. As pattern projector, we selected a visible pseudo random projector that produces a fixed pattern by means of a special lens placed in front of a red dot laser (wavelenght 660nm). The projector has a horizontal and vertical field of view of 35°, while the pattern is composed of 23,880 red dots.

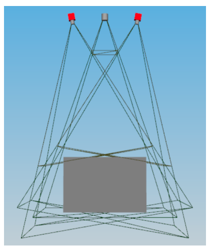

Fig. 4: Simulation of the sensors placement: the two red devices represent the cameras, the central gray device represents the pattern projector, while the gray box in the bottom represents the working volume. The green lines delimit the sensors frustums.

We studied the sensors placement by simulating the projector and cameras frustums and checking their intersections (Fig. 4). The current sensor placement, with a baseline between cameras of 500 mm, has been optimized for a $600 \times 400 \times 400$ mm working volume, at a distance of 1250 mm from the volume centroid. The theoretical resolution at farthest distance is 0.3 mm.

The FlexSight C1 exposes an Ethernet interface for data exchange, an USB interface, a HDMI video connection, and some general purpose I/Os for interfacing with low level devices, such as grippers, proximity sensors, etc. It also integrates a led illumination system to self illuminate the working area (white bars in Fig. 1, top left).

This design enables the possibility to mount the system directly on top of a robotic cell and being connected bidirectionally with the robot system without the need of any external unit (e.g., Fig. 1, right).

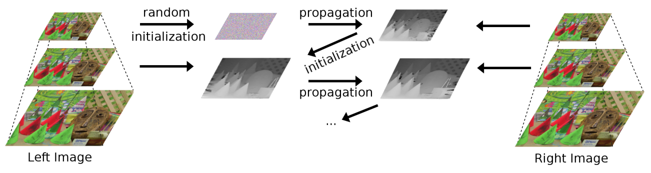

Fig. 5: Overview of the proposed pyramidal stereo matching. Given two rectified images, we construct the pyramids and process from top to bottom. On each level of the pyramids, the matching algorithm proposed in [4] is performed, and the results of each level are used as initialization of the next lower level. The disparities computed at the bottom level are finally refined by applying occlusion filling via left/right consistency checking.

IV. Perception

A. Pyramidal Stereo Matching

In local stereo matching, a support window is centered on a pixel of the reference frame. In order to find the correspondence, this support window is displaced in the second image along the related epipolar line to find the point of lowest dissimilarity. Here is used the implicit assumption that the pixels within the support region have a constant disparity. This does not apply to slanted surfaces, which are then reconstructed as compositions of frontal-parallel surfaces. The PatchMatch Stereo algorithm presented in [4] overcomes this problem by estimating a 3D plane at each pixel onto which the support window is projected (i.e., each disparity is parametrized as a 3D plane) . In particular, after a random initialization of the 3D planes guesses, three main steps are performed in [4]: spatial propagation, random search and view propagation. In the spatial propagation step, the planes guesses with low cost are propagated to the neighboring planes. Then, each estimated plane is randomly perturbated within a certain range in order to find a better estimation that has lower cost. In the view propagation step, each plane is reparametrized in the other view and propagated if the cost is lowered. These three steps are iterated until convergence. At the end, the algorithm provides both left and right disparities that are refined by applying occlusion filling via left/right consistency checking.

As shown in [34], this technique provides very accurate disparities but it is also very slow, i.e., it is not suitable for real-time computing.

Inspired by [35], we propose to embed the PatchMatch Stereo algorithm in a pyramidal framework (see Fig. 5) in order to reduce the matching time, while sensibly increasing the accuracy of the estimated disparities.

Two pyramids are built from the input rectified stereo image pair and processed from top (lower image resolution) to bottom (full image resolution). On each pyramid level, the PatchMatch Stereo algorithm provides left and right disparities estimations that are used as initialization for the next levels. The good initial guesses coming from the upper pyramids levels enable a considerable speed up of the random search step of the matching algorithm in the lower levels.

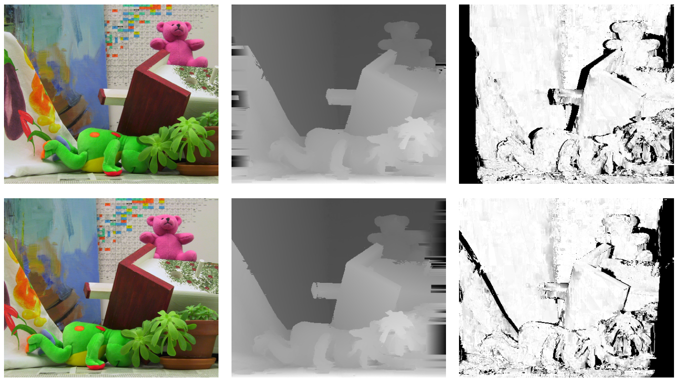
Fig. 6: An example of estimated confidence maps: (left) Input left and right RGB images; (middle) Left and right disparity maps used for the confidence computation; (right) Left and right resulting confidence maps.

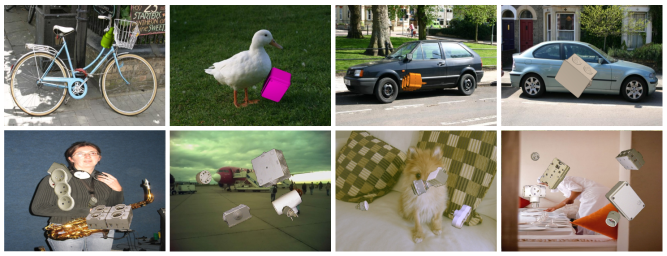
Fig. 7: Sample images from the *full-synthetic* (top row) and *semi-synthetic* (bottom row) datasets.

The random search step is the most time consuming part of the matching algorithm. However, decreasing the iterations of the random search may lead to not accurate disparity estimations. Let assume having the confidence of each estimated pixel's disparity, the idea is to relax the random search for high confidence disparities, while focus on the refinement of disparity estimation for those pixels with low confidence. In this work, we compute the confidence maps from the left/right disparities estimated in the previous (higher) pyramid level. Formally, for each pixel p in the reference image with estimated disparity d_p, we compute its matching point p' in the other view with disparity $d_{p'}$. The confidence $C(p)$ is given by:

$$C(p) = 1/(1 + |d_p - d_{p'}|) \tag{1}$$

An example of estimated confidence maps is shown in Fig. 6. Then, we use $C(p)$ in the random search step of the PatchMatch Stereo matching algorithm in order to reduce the number of iterations accordingly to the estimated disparity confidence.

Finally, the disparities provided at the bottom of the pyramids are post-processed for occlusion handling as done in [4].

B. Deep Learning Texture-less Object Detection

We address the object detection task using state of the art methods based on CNNs. These methods require huge amount of annotated data for training. We overcome the problem of manual annotating thousand of images implementing a full-automatic procedure for synthetic data generation. In

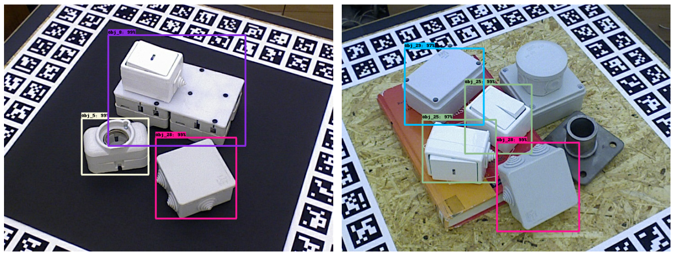

Fig. 8: Object detection examples with the *semi-synthetic* training.

particular, synthetic data have been generated using 2 different approaches (see Fig. 7):

a) *full-synthetic*: exploiting the information of the 3D CAD model of the object, train data are created by projecting the model on the image plane using random natural images as background;
b) *semi-synthetic*: train data are generated using images of real objects. Those images are segmented and the extracted object patches are then overlapped onto random natural images as background.

Both approaches make possible to generate infinite amount of data by controlling multiple parameters such as:

- For case a): Number and type of the objects for each sample, object pose in camera frame, level of occlusion, color of the render, illumination type (diffuse or spotlight) and position, level of gaussian noise to be applied to the rendered object;
- For case b): Number and type of the objects for each sample, patch position in the xy image 2D space, level of occlusion, level of gaussian noise to be applied to the patch, rotation and scale of the patch.

In both cases, it is also possible to apply some post-processing to the generated images, e.g., horizontal-vertical flip and slight rotation along the z-axis of the image plane.

Using the two sets of data, we are able to train the *Single Shot Detector (SSD)* network using the implementation available within the Google Detection API[1]. In particular we chose the SSD implementation that uses the *MobileNet* [36] network as feature extractor because it has been demonstrated to be the fastest network while maintaining comparable performance in terms of detection accuracy. The network has been trained performing transfer learning by freezing during train all the feature extractor layers set with the pre-trained on ImageNet weights, and only the detector part of the network has been left free to be trained with our datasets. This method has been demonstrated to be very effective when training large and complex deep convolutional networks with synthetic images [31].

C. 6D Object Localization

The neural network provides as output a set of 2D bounding boxes enclosing the searched objects within the image (e.g.,

[1]https://github.com/tensorflow/models/tree/master/research/object_detection

Algorithm	bad 0.5	bad 1.0	bad 2.0	bad 4.0	Runtime
PSMNet [19]	89.4%	76.5%	57.1%	35.9%	0.7 s (GPU)
MC-CNN [20]	67.9%	40.2%	26.7%	13.9%	101 s (GPU)
ELAS [38]	67.3%	38.6%	25.9%	13.5%	**0.3 s**
PMS [4]	47.2%	27.5%	15.8%	6.2%	22.3 s
PPMS (Ours)	**46.3%**	**25.8%**	**12.9%**	**5.5%**	8.7 s

TABLE I: Average performance on Middlebury training dataset [39] (best results are highlighted in bold).

Algorithm	bad 2.0	bad 3.0	Runtime
PSMNet [19]	**2.4%**	**1.5%**	0.4 s (GPU)
MC-CNN [20]	3.9%	2.4%	67 s (GPU)
ELAS [38]	10.8%	8.2%	**0.2 s**
PMS [4]	8.1%	5.3%	13.1 s
PPMS (Ours)	7.4%	4.5%	5.6 s

TABLE II: Average performance on Kitti Stereo 2012 dataset [40].

Fig. 8) and the related object class. This information is used by the position refinement module to estimate the exact 6D position (rotation and translation) of the object w.r.t. the camera reference frame. To this purpose, we use the model based localization algorithm proposed in [37]. This algorithm is executed only inside the detected bounding boxes, searching for a single object class for each box and using the depth values in that part of the image as initial guess for the object scale. To further refine the estimated object position, we align each object model with the input point cloud (e.g., Fig. 2, right) by using the Iterative Closest Point (ICP) algorithm.

V. Experiments

A. Stereo Matching

The proposed stereo matching algorithm has been tested and evaluated on two popular benchmark datasets: Middlebury Stereo 2014 [39] and Kitti Stereo 2012 [40]. We evaluated our approach (Pyramidal PatchMatch Stereo, PPMS) against the original PatchMatch Stereo (PMS) method [4], the popular and very efficient ELAS stereo matcher [38], and two state-of-the-art deep learning based methods: PSMNet [19] and MC-CNN [20]. For ELAS, PSMNet and MC-CNN, we used the original open-source implementations provided by the authors, while for PMS we used an effective third-party open-source implementation[2]. Although we have efficiently implemented the PPMS algorithm for our sensor, we tested PPMS, PMS and ELAS on an Intel Core-i7 5700HQ 2.70GHz CPU since the the last two are not provided with implementations optimized for ARM CPUs. PSMNet and MC-CNN have been tested on a NVIDIA GTX 1070 GPU using the pretrained models on the Kitti Stereo 2012 training dataset. The results in Tab. I refer to down-scaled ($0.5Mpx$) version of the Middlebury training images. The evaluation on the Kitti dataset (see Tab. II), instead, has been performed using the original resolution (1242x375px) of the images. As reported in Tab. II, PSMNet and MC-CNN show superior performance when using fine-tuned models on the specific benchmark. However, the degraded results in Tab. I show the difficulty of these techniques to generalize to completely different scenarios. The proposed method, instead, is able to generalize (the same set of parameters has been used in both evaluations), providing comparable results, in terms of bad pixel rate[3], in both benchmarks and outperforming the other algorithms. More specifically, the proposed method, compared to [4], is able to decrease the computational time up to 60%, while the accuracy of the disparities is improved up to 20%, showing the effectiveness of the pyramidal framework.

[2]https://github.com/ZhaozhengPlus/PatchMatchStereo

Training Data	Obj_5	Obj_8	Obj_9	Obj_10	Average
full-synthetic	0.3732	0.288	0.3179	0.2725	0.3129
semi-synthetic	0.5283	0.468	0.4956	0.477	**0.49225**

TABLE III: Detection performance for 4 classes and average results for all the 30 classes. Results are given in terms of mAP@0.5 (mean Average Precision with 0.5 Intersection Over Union threshold).

B. Deep Learning Texture-less Object Detection

For the object detection experiments, datasets have been generated using the objects from the *T-Less Dataset* [41] and the images used as background are from a selection of the *Microsoft Research Cambridge Object Recognition Image Database*[4]. For both the approaches, a set of 10000 samples have been generated. Some examples of the generated synthetic images are given in Fig.7.

In Table III we report quantitative results showing the mean average precision among all the 30 classes of the *T-Less test primesense* dataset. The table also shows detection results for some specific object classes. Training the model with the *full-synthetic* data results in poor performance, this means that the network is not capable of generalize well the task. An improvement of performance has been obtained with the *semi-synthetic* one. This demonstrates that the latter approach makes the transfer learning task easier: real object patches contains more visual information and this helps the network in learning more robust features. The SSD model with the *MobileNet* [36] feature extractor runs on the FlexSight C1 in inference mode at 7 fps.

VI. Conclusion

In this work an embedded, all-in-one system for machine vision in industrial settings has been proposed. Moreover, two contributions have been shown, namely an efficient pyramidal implementation of the PatchMatch stereo matching algorithm and an object detection pipeline based on deep convolutional networks trained with synthetic data. The two approaches are meant for running on the proposed embedded device and showed promising results in terms of accuracy and computation time trade-off.

Future works include a highly optimized GPU implementation of the proposed stereo matcher and an enhanced synthetic training data generator that employ state-of-the-art ray tracing techniques to improve the photorealism of the generated data.

[3]The "bad N" metric, used in Tab. I and II, refers to the percentage of pixels whose disparity error is grater than N.

[4]https://www.microsoft.com/en-us/download/details.aspx?id=52644

References

[1] *KUKA_3D Perception sensor by KUKA*, 2019 (accessed May, 2019). [Online]. Available: https://www.kuka.com/en-us/products/robotics-systems/robot-periphery/kuka-3d-sensoren

[2] *PhoXi 3D Scanner by Photoneo*, 2019 (accessed May, 2019). [Online]. Available: https://www.photoneo.com/phoxi-3d-scanner

[3] *Basler ToF Camera by Basler*, 2019 (accessed May, 2019). [Online]. Available: https://www.baslerweb.com/en/products/cameras/3d-cameras/time-of-flight-camera

[4] C. R. Michael Bleyer and C. Rother, "Patchmatch stereo - stereo matching with slanted support windows," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2011, pp. 14.1–14.11.

[5] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1, pp. 7–42, Apr 2002.

[6] K.-J. Yoon and I.-S. Kweon, "Locally adaptive support-weight approach for visual correspondence search," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, June 2005, pp. 924–931 vol. 2.

[7] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," in *IEEE Computer Vision and Pattern Recognition*, 2011.

[8] J. Lu, H. Yang, D. Min, and M. N. Do, "Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[9] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *International Journal of Computer Vision*, vol. 70, no. 1, pp. 41–54, Oct 2006.

[10] F. Besse, C. Rother, A. Fitzgibbon, and J. Kautz, "Pmbp: Patchmatch belief propagation for correspondence field estimation," *International Journal of Computer Vision*, vol. 110, 10 2013.

[11] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 109–117.

[12] Y. Li, D. Min, M. Brown, M. Do, and J. Lu, "Spm-bp: Sped-up patch-match belief propagation for continuous mrfs," in *2015 International Conference on Computer Vision, ICCV 2015*. United States: Institute of Electrical and Electronics Engineers Inc., 2 2015, pp. 4006–4014.

[13] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, Feb 2008.

[14] M. Bleyer and M. Gelautz, "Simple but effective tree structures for dynamic programming-based stereo matching," in *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2008, pp. 415–422.

[15] Y. Zhang, S. Khamis, C. Rhemann, J. P. C. Valentin, A. Kowdle, V. Tankovich, M. Schoenberg, S. Izadi, T. A. Funkhouser, and S. R. Fanello, "Activestereonet: End-to-end self-supervised learning for active stereo systems," *CoRR*, vol. abs/1807.06009, 2018.

[16] S. R. Fanello, J. Valentin, C. Rhemann, A. Kowdle, V. Tankovich, P. Davidson, and S. Izadi, "UltraStereo: Efficient learning-based matching for active stereo systems," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] S. R. Fanello, J. Valentin, A. Kowdle, C. Rhemann, V. Tankovich, C. Ciliberto, P. Davidson, and S. Izadi, "Low compute and fully parallel computer vision with hashmatch," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 3894–3903.

[18] S. R. Fanello, C. Rhemann, V. Tankovich, A. Kowdle, S. O. Escolano, D. Kim, and S. Izadi, "Hyperdepth: Learning depth from structured light without matching," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 5441–5450.

[19] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5410–5418.

[20] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *Journal of Machine Learning Research*, vol. 17, pp. 1–32, 2016.

[21] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 824–840, May 2009.

[22] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, 2010.

[24] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[25] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 580–587.

[26] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 91–99.

[27] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2155–2162, 2014.

[28] C. Szegedy, S. E. Reed, D. Erhan, and D. Anguelov, "Scalable, high-quality object detection," *CoRR*, vol. abs/1412.1441, 2014.

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

[30] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[31] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," *CoRR*, vol. abs/1710.10710, 2017.

[32] M. Rad, M. Oberweger, and V. Lepetit, "Domain transfer for 3d pose estimation from color images without manual annotations," *CoRR*, vol. abs/1810.03707, 2018.

[33] S. Thalhammer, T. Patten, and M. Vincze, "Towards object detection and pose estimation in clutter using only synthetic depth data for training," in *Proceedings of ARW and OAGM Workshop 2019*, 2019.

[34] B. Tippetts, D. J. Lee, K. Lillywhite, and J. Archibald, "Review of stereo vision algorithms and their suitability for resource-limited systems," *Journal of Real-Time Image Processing*, vol. 11, no. 1, pp. 5–25, Jan 2016.

[35] Y. Hu, R. Song, and Y. Li, "Efficient coarse-to-fine patch match for large displacement optical flow," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 5704–5712.

[36] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[37] M. Imperoli and A. Pretto, "D^{2}CO: Fast and robust registration of 3D textureless objects using the Directional Chamfer Distance," in *Proc. of 10th International Conference on Computer Vision Systems (ICVS 2015)*, 2015, pp. 316–328.

[38] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Asian Conference on Computer Vision*, Queenstown, New Zealand, November 2010.

[39] D. Scharstein, H. Hirschmller, Y. Kitajima, G. Krathwohl, N. Nesic, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth." in *GCPR*, ser. Lecture Notes in Computer Science, vol. 8753, 2014, pp. 31–42.

[40] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[41] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.