Preprint of the paper which appears in:
European Conference on Mobile Robots (ECMR) 2021

A Flexible Framework for Virtual Omnidirectional Vision to Improve Operator Situation Awareness

Martin Oehler and Oskar von Stryk

***Abstract*— During teleoperation of a mobile robot, providing good operator situation awareness is a major concern as a single mistake can lead to mission failure. Camera streams are widely used for teleoperation but offer limited field-of-view. In this paper, we present a flexible framework for virtual projections to increase situation awareness based on a novel method to fuse multiple cameras mounted anywhere on the robot. Moreover, we propose a complementary approach to improve scene understanding by fusing camera images and geometric 3D Lidar data to obtain a colorized point cloud. The implementation on a compact omnidirectional camera reduces system complexity considerably and solves multiple use-cases on a much smaller footprint compared to traditional approaches such as actuated pan-tilt units. Finally, we demonstrate the generality of the approach by application to the multi-camera system of the Boston Dynamics Spot. The software implementation is available as open-source ROS packages on the project page `https://tu-darmstadt-ros-pkg.github.io/omnidirectional_vision`.**

I. INTRODUCTION

Teleoperated robots are often deployed for tasks in hazardous environments like search and rescue. They enable the operator to safely explore and interact with the environment by remote operation. However, operator errors are a common reason for the failure of a robot mission. A frequent cause for these errors is a lack of situation awareness [1].

Commonly, remotely operated robots are equipped with a camera to stream video data to the operator. Compared to the human field-of-view, standard perspective cameras are very limited and lead to "tunnel vision". The peripheral vision loss hinders the operator's ability to build a mental model of the robot's surroundings.

In many cases, this problem is addressed by one or a combination of the following approaches: 1) Mounting a wide-angle fish-eye camera instead of a standard camera [2] increases field-of-view (FOV) in one direction. The severe distortion of these images makes the judging of spatial relations very challenging for the operator. Furthermore, blind spots remain to the sides and behind the vehicle. 2) Additional cameras can be mounted to cover these blind spots at the cost of increased system complexity. This increases the mental load of the operator because attention has to be switched between views [2]. In practice, operators mainly focus on the primary camera and often miss changes in auxiliary views [3]. 3) A wide-spread alternative is a mechanical pan-tilt sensor head that points a mounted camera in the desired viewing direction. These systems are complex with higher space requirements compared to individual cameras, limiting applicability on small robot platforms. Further drawbacks are introduced by the actuation. Mechanical limits can lead to blind spots and slow-moving actuators and network latency can result in a low control responsiveness.

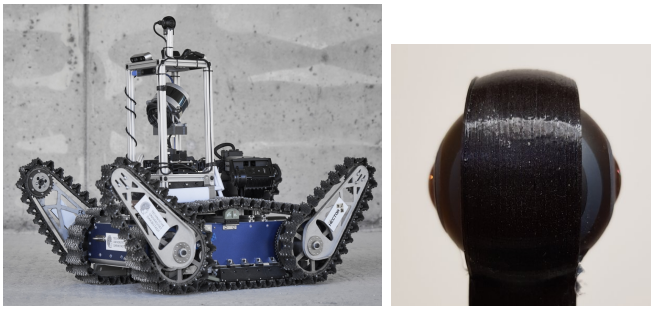
All authors are with the Simulation, Systems Optimization and Robotics Group, Technical University of Darmstadt, Germany {oehler, stryk}@sim.tu-darmstadt.de

Research presented in this paper has been supported in parts by the German Federal Ministry of Education and Research (BMBF) within the subproject "Autonomous Assistance Functions for Ground Robots" of the collaborative A-DRZ project (grant no. 13N14861), by the German Federal Ministry of Economics & Technology's "EXIST Forschungstransfer" (grant no. 03EFLHE061) and by Nexplore within the AICO Collaboration Lab. This work has been co-funded by the LOEWE initiative (Hesse, Germany) within the emergenCITY center.
978-1-6654-1213-1/21/$31.00 ©2021 IEEE

Fig. 1: Two fish-eye images are streamed from the omnidirectional camera (top-left), which are converted to perspective (bottom-left) and Mercator projection (right).

A further problem of teleoperation using a video stream is the lack of depth information. As a result, estimating the distance to obstacles is challenging, making navigation in narrow spaces demanding and error-prone [2]. There are different sensor modalities available that provide depth information, with Lidar and depth cameras being the most popular choices. Lidar sensing provides high-precision range information with a high field-of-view, but a fundamental drawback is the lack of color information. Based on the geometric range data alone, it is challenging for humans to understand the environment. For instance, doors are hard to distinguish from walls. Most depth cameras provide color information, but their range, accuracy and FOV are limited compared to Lidar.

In this paper, we propose two complementary approaches improving operator situation awareness using omnidirectional vision. We present a flexible framework to compute virtual projections from multiple cameras mounted anywhere on the robot. This allows generating novel views, giving the operator an unrestricted 360° view of the environment (see Fig. 1). By recomputing the projection online, arbitrary



(a) *Asterix* (b) Insta360 Air

Fig. 2: The *Insta360 Air* mounted on top of the center-frame of the highly mobile tracked platform *Asterix*.

frames on the robot or in the environment can be tracked. In this way, a virtual pan-tilt sensor head can be realized. This is complemented by a second approach that improves scene understanding by fusing Lidar data with color information from multiple cameras simultaneously to obtain a colorized point cloud.

The combination of both approaches in a comprehensive user interface allows for an improved situation-aware teleoperation. By generating virtual views, the operator can navigate with rectified pinhole projections, which are easier to comprehend than distorted fish-eye images. Spatial awareness is increased by additionally displaying a colorized point cloud in a 3D scene. The implementation of both approaches is made available as open-source ROS packages[1].

We demonstrate advantages compared to traditional approaches on two different multi-vision systems. First, we implement our approach on the *Insta360 Air* (Fig. 2b) mounted on the tracked robot *Asterix* [4] (Fig. 2a). While we used a complex multi-camera system in the past, switching to a single compact omnidirectional camera reduced system complexity considerably and solved different use-cases on a much smaller footprint. We highlight the generality of our projection framework by implementing it on the multi-camera system of the *Boston Dynamics Spot*. With five standard cameras mounted on the robot body, its camera configuration is significantly different to the *Insta360 Air*. *Spot* in its standard configuration does not feature a directly forward-facing camera. Therefore, we fuse the two angled front cameras to generate a view suitable for teleoperation. As a quadrupedal system, Spot can move in any direction. In this case, it is especially important to be aware of obstacles to the sides. To improve awareness of surrounding obstacles, we generate a virtual bird's eye view that shows the environment from above.

II. RELATED WORK

Various approaches to improve operator situation awareness using omnidirectional vision have been proposed.

The authors of [3] compare the effect of different camera placements on the teleoperation performance. They find that visualizing the robot and its surroundings in one image significantly reduces collisions. This view can be achieved by an overhead camera, but mounting a pole for attachment is impractical. In [5], the authors compute a virtual bird's-eye view from multiple fish-eye cameras by transforming the images to the ground plane with a homography. This leads to a distortion of obstacles, because their height does not match the ground. This problem is addressed in [6] by incorporating data from depth sensors. Obstacles are superimposed on the bird's-eye view to correctly visualize their location.

An overview of human performance issues related to user interface design is given in [2]. A low FOV is identified to cause erroneous speed and distance judgments as well as peripheral vision loss. In [7], a fish-eye camera is used to increase operator FOV. The authors present an approach to generate a virtual focus and context view that shows a perspective view in the center and transitions to a distorted view in border regions. This allows for an undistorted view in the center while still providing peripheral information.

The authors of [1] compare different approaches for human-robot interaction in a case study of the DARPA Robotics Challenge Finals. They conclude that situation awareness is improved when data is visualized in a virtual environment and the operator is able to control the viewpoint. In [8], this is achieved by projecting the image of four fish-eye cameras onto a hemispherical mesh. The approach leads to distortions because the mesh does not match the environment. Moreover, scale depends on the mesh size, which can be irritating. This is improved in [9] by taking the environment into account. An environment mesh is approximated with laser range finders and textured with images from multiple fish-eye cameras. While this provides good situation awareness in certain environments, the assumption that only flat walls are present leads to a distortion of obstacles. The authors of [10] take a similar approach and project the images of two fish-eye cameras to a sphere. They render perspective views by placing a virtual camera inside the sphere and display this image on a head-mounted display with orientation tracking for an intuitive visualization. As part of our evaluation, we compare this approach to ours. Additionally, they propose a second approach to project Lidar points onto the sphere to fuse color information with depth. Further research on colorizing point clouds has been performed in [11]. The authors propose an approach to color Lidar point data with an attached camera. They fuse color information from multiple observations and take visibility and temporal offsets into account.

III. METHOD

In the following section, we describe our approach to create projections from multiple source cameras. Afterwards, we explain how point cloud data is colorized from the same cameras.

A. Camera and Lidar Calibration

The described approaches require an accurate calibration of intrinsic and extrinsic parameters of all cameras and the

[1] https://tu-darmstadt-ros-pkg.github.io/omnidirectional_vision

extrinsic calibration of the Lidar.

The general model of projection for a single camera C_i is given by:

$$\pi_{c_i}\left({}^{C_i}T_R\mathbf{p}\right) = \mathbf{u} \tag{1}$$

The extrinsic camera calibration describes the relative 6D pose of each camera to a common reference frame R, e.g. the first camera or a common camera head. It is given by the SE(3) transformation ${}^{R}T_{C_i}$. The inverse ${}^{C_i}T_R$ transforms point $\mathbf{p} = (x, y, z)$ from reference frame to camera frame. The intrinsic camera calibration yields the projection function $\pi_{c_i} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ which projects a 3D point onto a 2D image coordinate $\mathbf{u} = (u, v)$. Note that we use a general projection model and no assumption about a specific lens type is made. This allows our approach to generalize to different projection models, e.g. the standard pinhole camera model and wide-angle lens models. Extrinsic and intrinsic calibration are performed jointly for all cameras with the calibration framework *kalibr*[2] [12].

In a second step, the extrinsic calibration between camera and Lidar is estimated. It describes the 6D pose of the Lidar frame L relative to the camera reference frame R given by the transform ${}^{R}T_L$. A target-less calibration is performed using the approach described in [13].

B. Image Projection with Multiple Cameras

A projection is defined by a function $P : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ which maps coordinates from the output projection image to the input camera image. In case of multiple cameras, one projection function is defined per camera:

$$P_{c_i}(\mathbf{u}_p) = \mathbf{u}_{c_i} \tag{2}$$

with $\mathbf{u}_p = (u_p, v_p)$ being the projection image coordinates and $\mathbf{u}_{c_i} = (u_{c_i}, v_{c_i})$ the image coordinates in camera C_i respectively. The projection function is used to look up camera image coordinates for each pixel of the output image.

Geometrically, projections can be represented by 3D surfaces. The shape of the surface depends on the projection type and parameters, which should be chosen based on the desired application. The 6D pose of the projection surface frame P relative to the camera reference frame R is specified by the user with the transform ${}^{R}T_P$.

Based on the projection surface, the projection function is computed. First, the surface is discretized by grid-based sampling with the desired target resolution $(W_p \times H_p)$. Each 3D sample point $\mathbf{x}_p = (x_p, y_p, z_p)$ corresponds to a pixel in the output image (u_p, v_p). This mapping between 2D projection image and 3D projection surface is given by the surface projection function $S : \mathbb{R}^2 \rightarrow \mathbb{R}^3$:

$$S(u_p, v_p) = \mathbf{x}_p = \begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} \tag{3}$$

The projection surface function for three different projections is detailed in the following.

[2] https://github.com/ethz-asl/kalibr

Perspective projection: The perspective projection surface function describes a plane. The parameters focal length f and vertical FOV ϕ_h of the pinhole model are specified by the user. Based on this, the pixel size m_p is computed:

$$m_p = \frac{2f\tan\left(\phi_h/2\right)}{W_p} \tag{4}$$

which leads to:

$$S\left(u_p, v_p\right) = \begin{pmatrix} (u_p - W_p/2)\cdot m_p \\ (v_p - H_p/2)\cdot m_p \\ f \end{pmatrix} \tag{5}$$

Mercator projection: The Mercator projection is represented by a cylinder and gives a panoramic view of the environment. The desired vertical FOV ϕ_v and cylinder radius c_r are specified by the user. Based on this, angular pixel size α_p and cylinder height c_h are computed:

$$\alpha_p = \frac{2\pi}{W_p} \tag{6}$$

$$c_h = 2c_r\tan\left(\phi_v\right) \tag{7}$$

The surface projection function is then given by:

$$S\left(u_p, v_p\right) = \begin{pmatrix} c_r\cos(-u_p\alpha_p) \\ c_r\sin(-v_p\alpha_p) \\ c_h(0.5 - \frac{v_p}{H_p}) \end{pmatrix} \tag{8}$$

Spherical projection: A spherical projection can be geometrically represented by a sphere. The FOV ϕ and sphere radius s_r are given by the user. First, the projection image coordinates (u_p, v_p) are transformed to normalized polar coordinates $r \in [0; 1]$ and $\gamma \in (-\pi, \pi]$. The origin of the polar coordinate system is located in the center of the image:

$$p_x = \frac{u_p}{W_p} - 0.5 \tag{9}$$

$$p_y = 0.5 - \frac{v_p}{H_p} \tag{10}$$

$$r = \sqrt{p_x^2 + p_y^2} \tag{11}$$

$$\gamma = \arctan 2(p_y, p_x) \tag{12}$$

To cast a ray into 3D, we also compute the second angle θ which corresponds to the latitude while γ corresponds to the longitude of the sphere:

$$\theta = r\frac{\phi}{2} \tag{13}$$

Based on latitude θ and longitude γ, the sphere is computed:

$$S\left(\theta, \gamma\right) = \begin{pmatrix} s_r\sin\theta\cos\gamma \\ -s_r\sin\theta\sin\gamma \\ s_r\cos\theta \end{pmatrix} \tag{14}$$

Fish-eye lenses produce spherical images. An ideal fish-eye lens has a linear relationship between distance from image center r and latitude θ as can be seen in (13). Real fish-eye

lenses are rarely linear, therefore, a projection can be used to "rectify" the image.

Using the surface projection function S, a surface sampling point $\mathbf{x}_p$ is computed for each pixel in the output image.

In the next step, each point $\mathbf{x}_p$ is transformed into the camera frame of each camera using the user-specified surface pose ${}^{R}T_P$ and the extrinsic camera calibration ${}^{C_i}T_R$:

$$\mathbf{x}_{c_i} = {}^{C_i}T_R \, {}^{R}T_P \mathbf{x}_p \tag{15}$$

Finally, we project the point into all camera images with the intrinsic calibration function $\pi(\cdot)$ of each camera:

$$\pi_i\left(\mathbf{x}_{c_i}\right) = \mathbf{u}_{c_i} \tag{16}$$

Using this process, we compute the mapping P_{c_i} for each pixel of the output image $\mathbf{u}_p$ and for each camera C_i. In the case where we have multiple valid mappings for a pixel, i.e. when the projection surface is seen by multiple cameras, we prefer the mapping, where $\mathbf{u}_{c_i}$ is closest to the respective image center. Alternatively, one could consider all observations by blending their color.

The projection mapping function has to be computed only once. Each time a new image is received by the cameras, the output image is populated using the computed mappings for each camera. The color is bi-linearly interpolated from the camera images.

C. Cloud Coloring

The Lidar cloud is colored online by assigning colors to each point as soon as a new scan cloud is received. For this, each point of the scan is projected into the latest camera images. This follows the same idea as the previously described projection method. The 3D projection surface is replaced by the 3D points retrieved from the Lidar scans.

Lidar scan points $\mathbf{x}_L$ are given relative to the Lidar sensor frame L. They are transformed to each camera frame using the extrinsic Lidar-camera calibration ${}^{R}T_L$ and the extrinsic camera calibration ${}^{C_i}T_R$ of each camera:

$$\mathbf{x}_{c_i} = {}^{C_i}T_R \, {}^{R}T_L \mathbf{x}_L \tag{17}$$

Using the intrinsic calibration, the points are projected into all camera images using (16). If a point is visible from multiple cameras, the observation closest to the respective image center is preferred. The color is interpolated bi-linearly. This process is repeated for each point of the scan and for each camera.

IV. RESULTS

In this section, we highlight advantages of our proposed methods compared to traditional approaches in multiple use-cases. We use two different multi-camera systems for demonstration: the compact omnidirectional camera *Insta360 Air* mounted on our tracked platform *Asterix* and the quadrupedal robot *Boston Dynamics Spot*.

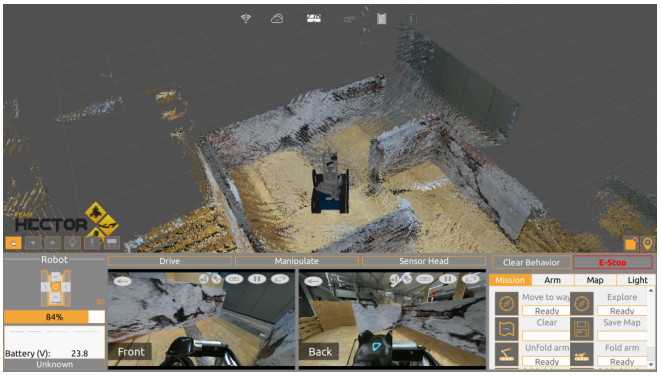


Fig. 3: Operator User Interface. The main view visualizes the colorized point cloud in a virtual representation of the environment. Below the main view, projected forward and backward cameras are shown. Their viewing direction can be adjusted online with a joystick.

A. Insta360 Air

The *Insta360 Air* (Fig. 2b) is a compact omnidirectional camera composed of two 210° fish-eye lenses on a spherical body with a diameter of 4 cm. The camera is mounted on our highly mobile tracked robot *Asterix* [4] (Fig. 2a) and is positioned on top of the central frame for maximum visibility. By replacing our previous multi-camera system with an omnidirectional camera, we reduced system complexity and space requirements significantly while solving the same use-cases. Additionally, calibration effort is reduced to a single camera with two lenses.

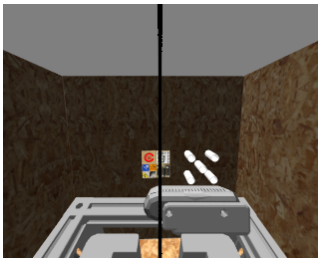
In the following, we present the integration of our approach into a single comprehensive user interface (see Fig. 3). It is composed of a virtual 3D scene in the main window and virtual views underneath.

Virtual views: We use two pinhole projections for virtual forward and backward driving cameras, giving the operator the capability to quickly switch the driving direction while also providing additional situation awareness behind the robot as advised in [3]. The projections feature a high FOV of 130° for improved visibility to the sides. Using projection, we can achieve a high FOV without distortions as would be observable on a comparable fish-eye lens. To ensure accuracy of the projection and reduce stitching seams, the projection surface can be adapted to the standard operational viewing distance by the user.

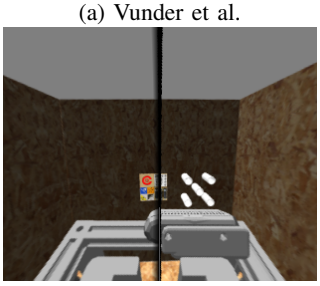
Additionally, the operator can rotate and zoom the projections to inspect the robot's surroundings without restrictions. This functionality is realized by recomputing the projection based on joystick input. This effectively implements a virtual sensor head without the mechanical complexity and space requirements of an actuated pan-tilt unit. Additionally, we can track arbitrary frames on the robot or the environment, for instance a manipulator end-effector.

The projections are generated directly on the robot to reduce bandwidth requirements. Only images of regions of interest are transmitted instead of the high-resolution spherical source images.

(a) Vunder et al.



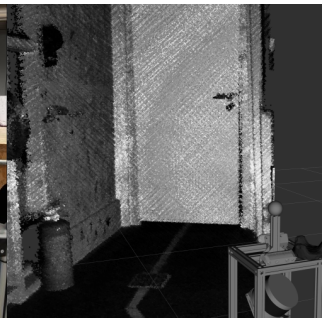
(b) Ours



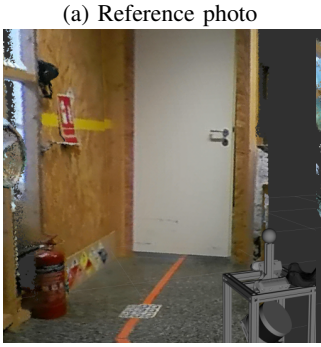
(c) Vunder et al., corrected by our approach

Fig. 4: Comparison between Vunder et al. [10] and our approach on a simulated omnidirectional camera with two lenses showing an offset of 3° around the optical axis. In (a), the wall shows a misalignment, which is not present in our approach (b). By correcting the fish-eye images in a pre-processing step using our approach, the wall is aligned (c).

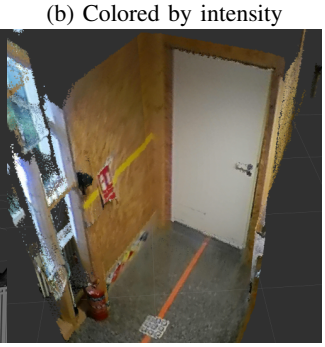
(a) Reference photo



(b) Colored by intensity



(c) Proposed approach



(d) Different perspective

Fig. 5: The colorized point cloud provides improved semantic understanding by making objects recognizable by color.

Creating virtual projections allows us to use the camera for more than one purpose at a time. Simultaneous to its function as an operator view, we use the camera for 360° object detection. A panoramic view of the environment based on the Mercator projection (see Fig. 1) is generated which is used to detect hazmat signs in the environment.

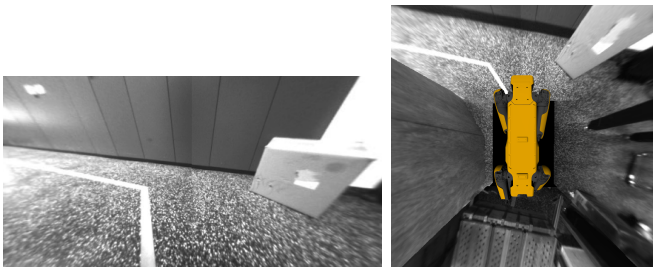
Next, we compare the pinhole projection in our approach to the one described by Vunder et al. [10] (Fig. 4). The approaches are fundamentally different as our approach creates pinhole projections directly as opposed to an intermediate mapping to a sphere. This intermediate step relies on the assumption that two ideal spherical images are used and the camera centers coincide. This assumption is rarely met on real cameras and leads to artifacts, as can be seen in the misalignment of the wall in Fig. 4a. Our approach considers extrinsic and intrinsic calibration of each camera and, therefore, does not show these artifacts (Fig. 4b). We can correct the projection using our approach by first computing an ideal spherical projection from the original fish-eye images (Fig. 4c).

3D view: The robot is equipped with a rotating tilted *Velodyne VLP-16* Lidar that generates a point cloud of the environment. The cloud is colorized online with the omnidirectional camera mounted above. Cloud points that are obstructed by robot parts are filtered to prevent parts of the robot itself being projected onto the environment. A 3D scene is displayed to the operator above the virtual camera views (Fig. 3). Following the conclusions in [1], spatial awareness is increased by visualizing the colorized cloud in this virtual representation of the environment that allows the operator to freely change the viewpoint. Based on the geometric data of the Lidar alone, it is very challenging for a human to develop a semantic understanding of the robot's environment. Consider the comparison made in Fig. 5. A reference photo of the scene is given in Fig. 5a. Fig. 5b shows the Lidar data, viewed from the same angle and colored by return intensity. While this view already allows us to identify obstacles and some objects, the white door, fire extinguisher and signs on the left are easier to identify in the colored cloud (Fig. 5c). Therefore, this approach combines spatial awareness through precise Lidar clouds with semantic information provided by colored camera data. Additionally to the benefits for teleoperation, the color information can also be leveraged for improved automatic 3D object detection.

B. Boston Dynamics Spot

The *Boston Dynamics Spot* is a quadrupedal robot equipped with a camera system composed of five grayscale pinhole cameras for omnidirectional vision. Compared to the previous omnidirectional camera application example, this camera configuration is significantly different and highlights the generality of the proposed approach. The five source camera images can be seen in Fig. 6a. The robot does not possess a forward-facing camera. Therefore, we use a perspective projection to fuse the two angled front cameras to generate a view suitable for teleoperation (Fig. 6b). With a horizontal FOV of 130°, the resulting projection provides better peripheral vision than a single camera. As a

(a) Spot camera images



(b) Virtual front view (c) Virtual bird's-eye

Fig. 6: The five source camera images of Spot can be seen in (a). They are used to compute a forward view (b) and a bird's eye view with overlayed robot model (c) for additional situation awareness.

legged platform, *Spot* can move in any direction. Switching between cameras to be aware of all movement directions simultaneously is demanding and error-prone [2]. We address this issue by generating a virtual bird's-eye view (Fig. 6c) overlayed with the rendered robot model, showing the robot and its environment in a single image as recommended by [3]. This view is achieved with a perspective projection by positioning the virtual camera above the robot and the projection surface on ground level. Compared to [5], the projection is generated directly without any intermediate steps. While this visualization increases situation awareness in close proximity to the robot, the operator has to be aware of the fact that objects are distorted, because they are higher than the ground.

C. Performance Evaluation

The performance has been evaluated on an *Intel i7-4710HQ@2.50GHz* mobile CPU and the *Insta360 Air* camera. We tested different projections and target resolutions. The required computation time for the two processing steps is shown in Table I. The projection mapping has to be generated only when the projection changes, e.g. on startup or when the view is rotated. The computation time scales linearly with the target resolution. This step is rather costly compared to the map operation which applies the mapping to the camera images to generate a new projection frame. Even with high target resolutions, a real-time application is possible. The cloud colorizing performance has been tested with the *VLP-16* Lidar. In our setup it produces about 18000 points per scan with a rate of 10 Hz. Processing of a single scan took 57.72 ms on average, allowing for real-time usage.

TABLE I: Image Projection Performance

Projection	Resolution	Projection mapping (ms)	Map operation (ms)
Perspective	512x256	23.78	12.74
Perspective	1024x512	86.27	12.5
Perspective	2048x1024	334.62	23.36
Mercator	1024x512	141.67	12.33
Spherical	512x512	92.26	10.9

V. CONCLUSION

We presented a novel flexible framework for providing virtual projections to remote robot operators based on multiple cameras anywhere on the robot as well as a complementary approach to colorize Lidar scans from image data. The software implementation of both approaches is available as open-source ROS packages. The combination of both approaches in a comprehensive user interface allows for significantly improved situation awareness during teleoperation.

As future work, we are planning to evaluate our approach on omnidirectional cameras with increased resolution. Furthermore, we expect GPU acceleration to significantly speed up computation times as the structure of data processing is well suited for parallelization.

REFERENCES

[1] A. Norton, W. Ober, L. Baraniecki, E. McCann, J. Scholtz, D. Shane, A. Skinner, R. Watson, and H. Yanco, "Analysis of human–robot interaction at the darpa robotics challenge finals," *IJRR*, vol. 36, no. 5-7, pp. 483–513, 2017.

[2] J. Y. Chen, E. C. Haas, and M. J. Barnes, "Human performance issues and user interface design for teleoperated robots," *IEEE Trans Syst Man Cybern C Appl Rev*, vol. 37, no. 6, pp. 1231–1245, 2007.

[3] B. Keyes, R. Casey, H. A. Yanco, B. A. Maxwell, and Y. Georgiev, "Camera placement and multi-camera fusion for remote robot operation," in *IEEE SSRR*, 2006, pp. 22–24.

[4] M. Schnaubelt, T. Ullrich, M. Torchalla, J. Diegelmann, M. Hoffmann, and O. von Stryk, "Entwicklung eines autonomiefokussierten hochmobilen Bodenrobotersystems für den Katastrophenschutz," in *VDI-MECHATRONIK*, 2021, pp. 20–25.

[5] T. Sato, A. Moro, A. Sugahara, T. Tasaki, A. Yamashita, and H. Asama, "Spatio-temporal bird's-eye view images using multiple fish-eye cameras," in *IEEE/SICE SII*, 2013, pp. 753–758.

[6] Y. Awashima, R. Komatsu, H. Fujii, Y. Tamura, A. Yamashita, and H. Asama, "Visualization of obstacles on bird's-eye view using depth sensor for remote controlled robot," in *IWAIT*, 2017.

[7] J. Vaughan, S. Kratz, and D. Kimber, "Look where you're going: Visual interfaces for robot teleoperation," in *IEEE RO-MAN*, 2016, pp. 273–280.

[8] S. Iwataki, H. Fujii, A. Moro, A. Yamashita, H. Asama, and H. Yoshinada, "Visualization of the surrounding environment and operational part in a 3dcg model for the teleoperation of construction machines," in *IEEE/SICE SII*, 2015, pp. 81–87.

[9] R. Komatsu, H. Fujii, Y. Tamura, A. Yamashita, and H. Asama, "Free viewpoint image generation system using fisheye cameras and a laser rangefinder for indoor robot teleoperation," *ROBOMECH*, vol. 7, no. 1, pp. 1–10, 2020.

[10] V. Vunder, R. Valner, C. McMahon, K. Kruusamäe, and M. Pryor, "Improved situational awareness in ros using panospheric vision and virtual reality," in *IEEE/IES HSI)*, 2018, pp. 471–477.

[11] P. Vechersky, M. Cox, P. Borges, and T. Lowe, "Colourising point clouds using independent cameras," *IEEE RA-L*, vol. 3, no. 4, pp. 3575–3582, 2018.

[12] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *IEEE/RSJ IROS*, 2013, pp. 1280–1286.

[13] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information," in *AAAI*, vol. 26, no. 1, 2012.