Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2024), Abu Dhabi, UAE, Oct. 2024.

SDPL-SLAM: Introducing Lines in Dynamic Visual SLAM and Multi-Object Tracking

Argyris Manetas[1], Panagiotis Mermigkas[1] and Petros Maragos[1,2]

***Abstract*— The need for a robust visual SLAM system operating in real human environments has led to the gradual abandonment of the static world assumption and to the creation of many dynamic SLAM algorithms. Even though there have been many dynamic SLAM proposals, the vast majority of them relied on point features. However, research in static SLAM systems has demonstrated that the use of more complex geometric shapes such as lines can improve performance. Motivated by this we have created a new dynamic SLAM system that estimates the camera poses and the motion of rigid objects, by exploiting both static and dynamic points and lines. Line segments have been incorporated in a novel way in every aspect of our algorithm, by improving their correspondences through optical flow refinement, and by introducing line error terms in both camera and object motion, and in batch optimization. Our proposal has been tested extensively in indoor and outdoor datasets and has achieved significant improvement compared to other state-of-the-art dynamic SLAM systems. Our results demonstrated that line segments enhanced the robustness, thus contributing towards a fully operational SLAM system.**

Code is publicly available*.

I. Introduction

Simultaneous Localization and Mapping (SLAM) is a well-studied area of Robotics and Computer Vision, crucial for many applications, including Augmented Reality, driverless cars, and house robots. SLAM aims to find the most probable trajectory of a robot, building at the same time a map of the environment. Map existence prevents accumulation of pose drift caused by noisy sensor measurements, while it also provides meaningful information about the environment. Different sensors have been utilized to solve this problem, such as cameras [1], in which case it is termed as visual SLAM, IMUs [2] and LiDAR [3]. Advances in camera technology and easier access to RGB-D cameras have led to the development of many robust visual SLAM systems.

SLAM algorithms have diversified in many ways. For example, various structural elements, such as sparse points [4], [5], voxels [2], surfels [6] or other geometric entities like lines and planes, are used for map representation. Likewise, tracking in visual SLAM is performed either directly [7] or by detecting features, like ORB [8] or more complex geometric shapes such as lines [9], planes [10] or both [11].

This research has been co-financed by the European Union NextGenerationEU under the call RESEARCH – CREATE – INNOVATE 16971 Recovery and Resilience Facility (project: ENORASI, code: TAEDK- 06172).

*https://github.com/argyrissm/SDPL-SLAM

[1]Electrical and Computer Engineering, National Technical University of Athens, 15780 Zografou, Athens, Greece argyris.manetas@gmail.com, pmermigkas@central.ntua.gr, maragos@cs.ntua.gr

[2]Robotics Institute, Athena Research Center, 15125 Maroussi, Athens, Greece

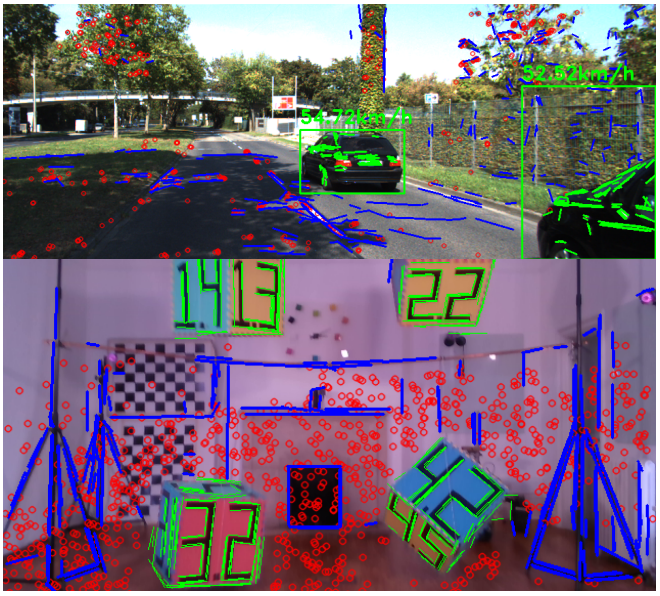

Fig. 1. **Output of our system:** Points and lines are tracked on both static and dynamic objects. Features presented: static points (Red), static lines (Blue), and dynamic lines (Green). Speed calculated from the estimated motion of cars is shown.

Traditionally in SLAM research, the world was assumed to be static and measurements on dynamic objects were handled with generic outlier rejection techniques, such as RANSAC [12], and robust loss functions, like the Huber loss function. This approach, however, is error-prone in highly dynamic environments, and due to the nonconvexity of the minimization problem used in SLAM, persisting outlier observations can prove detrimental to overall system accuracy. Therefore, it becomes apparent that the development of robust dynamic SLAM systems is vital for the operation of robots in real-life environments, which are dominated by humans, cars, and other moving objects.

Even though it has been proven that the use of more complex geometric shapes such as lines increases the robustness of SLAM [9], [13], especially in textureless and low-lit areas, little research has been done on their use in dynamic environments. Motivated by this and by the need for accurate SLAM systems in human-centered environments, we propose a SLAM system that tracks static and dynamic points and lines to estimate camera positions and motion of dynamic objects in the scene (see Fig. 1). Novelties are presented in every aspect of our implementation and include the usage of optical flow for richer line correspondences, the introduction of line reprojection error terms for camera tracking and

object motion estimation, with the concurrent optimization of optical flow as a two-fold contribution, and, lastly, the inclusion of lines in partial and global batch optimization. Combining the advantages of dynamic and line SLAMs, we developed a system that surpasses other state-of-the-art systems and verified its performance on both outdoor driving and dynamic indoor datasets.

II. Related Work

Traditionally, points were the de facto features used in the SLAM problem. PTAM [4] was such a system that divided tracking and mapping tasks into two threads to ensure real-time execution. ORB-SLAM2 [5] with the utilization of ORB features and the use of a sparse pose graph for Bundle Adjustment, achieved real-time performance, while also providing robustness with the capability to close loops and relocalize in cases of lost tracking.

However, points might provide insufficient correspondences in some low-light or low-texture areas and sparse point-based maps lack detailed information. On the contrary, more complex geometry shapes, such as lines, are commonly encountered and encapsulate more descriptive information about the environment. This observation led to the rise of many systems that utilized lines [9], [13], planes [10] or both [11]. To avoid suboptimal solutions when using these geometric entities in an optimization process, minimal representations are used, such as the orthonormal representation [14] for lines in [15].

Dynamic SLAM systems can be divided into two categories. Systems in the first category detect dynamic objects in frames and remove them from tracking and optimization procedures. DynaSLAM [16] leverages semantic masks provided by Mask R-CNN [17] and reprojection error checks to discard dynamic objects. In DS-SLAM [18] the distance from epipolar lines is used in conjunction with semantic segmentation to reject dynamic objects. StaticFusion [6] performs a joint estimation of camera pose and scene dynamicity, making use of a two-term energy error function. According to the estimated dynamicity a weight is attached to the observations, affecting their participation in the optimization problem.

On the contrary, systems of the second category detect dynamic features and track them without discarding parts of the frames, thus exploiting present information more efficiently and bridging the problem of SLAM and Moving Object Tracking. VDO-SLAM [19] utilizes semantic information to distinguish dynamic objects from the static environment, incorporates both in the SLAM framework, and calculates egomotion and dynamic rigid objects' independent movement without prior knowledge of their geometric models. DynaSLAM II [20] proposes a bundle adjustment problem that includes both static and dynamic features, in addition to providing and optimizing 3D bounding boxes of moving objects. AirDOS [21] addresses nonrigid dynamic objects, by including constraints in the motion of articulated objects.

In this paper, we propose a novel SLAM system of the second category, which combines the advantages of dynamic SLAM and the robustness of line SLAM systems, by tracking points and lines on both static environment and dynamic rigid objects, resulting in a highly accurate framework.

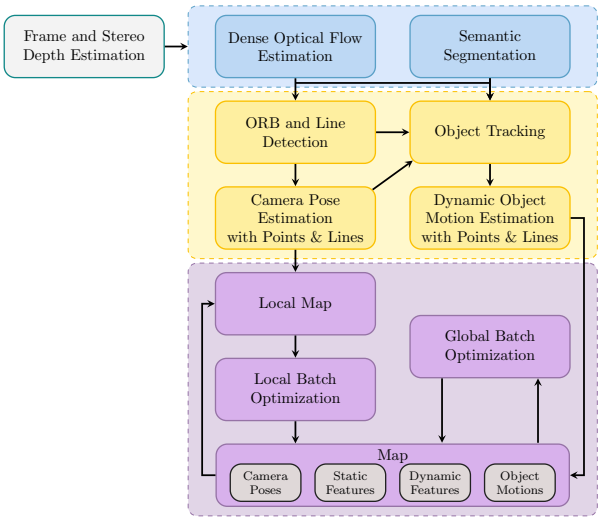

Fig. 2. **SDPL-SLAM (Static-Dynamic Point-Line SLAM) system overview:** Consists of three main components: pre-processing (Blue), tracking (Yellow), and batch optimization (Purple).

III. Methodology

The overview of our system can be seen in Fig. 2. The system receives RGB-D images as input, which are pre-processed to retrieve dense optical flow and semantic segmentation. In the tracking stage, the camera pose is calculated from the last frame using static point and line observations. Once camera pose is obtained, dynamic objects are tracked and their motion between two frames is retrieved. In parallel, a local and a global map are maintained. For every set number of time steps, a local batch optimization is performed on the local map to refine the local trajectory, whereas the global batch optimization is performed on the global map to jointly refine the whole trajectory and map.

A. Notation

Coordinate systems are denoted by C_k and placed as left superscripts for points and lines, excluding the global reference frame 0 which is omitted where possible.

Points: The (in)homogeneous 3D coordinates of the i^{th} point at frame k, with respect to coordinate system C_k, are denoted by ${}^{C_k}\mathbf{M}_k^i \in \mathbb{P}^3$ (and ${}^{C_k}\tilde{\mathbf{M}}_k^i \in \mathbb{R}^3$). Similarly, 2D coordinates with respect to coordinate frame I_k are represented as $\mathbf{m}_k^i \in \mathbb{P}^2$ (and $\tilde{\mathbf{m}}_k^i \in \mathbb{R}^2$). We consider that the last element of homogeneous coordinates is equal to 1.

Lines: A 3D line segment j at frame k can be represented by its endpoints $\{{}^{C_k}\mathbf{A}_k^j, {}^{C_k}\mathbf{B}_k^j\}$, while an infinite 2D line in coordinate frame I_k is denoted by $\mathbf{l}_k^j$. Plücker line coordinates can be constructed as:

$$ {}^{C_k}\mathcal{L}_k^j = \begin{bmatrix} {}^{C_k}\tilde{\mathbf{A}}_k^j \times {}^{C_k}\tilde{\mathbf{D}}_k^j \\ {}^{C_k}\tilde{\mathbf{D}}_k^j \end{bmatrix} = \begin{bmatrix} {}^{C_k}\tilde{\mathbf{N}}_k^j \\ {}^{C_k}\tilde{\mathbf{U}}_k^j \end{bmatrix} $$

where ${}^{C_k}\tilde{\mathbf{D}}_k^j$ is the directional unit vector of the line. It can be observed that this is not the general definition of

Plücker coordinates, since we also impose the two constraints $||^{C_k}\tilde{\mathbf{U}}^j_k|| = 1$ and $^{C_k}\tilde{\mathbf{N}}^j_k \cdot {}^{C_k}\tilde{\mathbf{U}}^j_k = 0$. These two constraints reduce the Plücker coordinates degrees of freedom to four, thus enabling a one-to-one transform to the orthonormal representation. The orthonormal representation of the line $(U, W) \in SO(3)\times SO(2)$ can be calculated from the Plücker coordinates as follows:

$$^{C_k}U^j_k(\boldsymbol{\theta}) = \begin{bmatrix} \frac{^{C_k}\tilde{\mathbf{N}}^j_k}{||^{C_k}\tilde{\mathbf{N}}^j_k||} & \frac{^{C_k}\tilde{\mathbf{U}}^j_k}{||^{C_k}\tilde{\mathbf{U}}^j_k||} & \frac{^{C_k}\tilde{\mathbf{N}}^j_k \times {}^{C_k}\tilde{\mathbf{U}}^j_k}{||^{C_k}\tilde{\mathbf{N}}^j_k \times {}^{C_k}\tilde{\mathbf{U}}^j_k||} \end{bmatrix}$$

$$^{C_k}W^j_k(\theta) = \begin{bmatrix} ||^{C_k}\tilde{\mathbf{N}}^j_k|| & -||^{C_k}\tilde{\mathbf{U}}^j_k|| \\ ||^{C_k}\tilde{\mathbf{U}}^j_k|| & ||^{C_k}\tilde{\mathbf{N}}^j_k|| \end{bmatrix}$$

Matrix U is updated with $\boldsymbol{\theta}$ and W with θ, as shown in [14].

Optical Flow: We define the vector that corresponds to the movement of a pixel $\tilde{\mathbf{m}}^i_{k-1}$ from I_{k-1} to I_k:

$$\boldsymbol{\phi}^i_k = \tilde{\mathbf{m}}^i_k - \tilde{\mathbf{m}}^i_{k-1}$$

Optical flows that correspond to a start or end point of a line j from I_{k-1} to I_k are $\boldsymbol{\phi}^{j,\mathbf{a}}_k$ and $\boldsymbol{\phi}^{j,\mathbf{b}}_k$, respectively.

Transformations: A transformation matrix from frame k' to k is denoted by $^{k'}X_k \in SE(3)$: $^{C_{k'}}\mathbf{M}^i_k = {}^{k'}X_k\, {}^{C_k}\mathbf{M}^i_k$. The transformation matrix $_{k-1}{}^0H_k \in SE(3)$ denotes a motion for points on dynamic rigid objects from frame $k-1$ to k with respect to the global reference frame 0, i.e. $^0\mathbf{M}^i_k = {}_{k-1}{}^0H_k\, {}^0\mathbf{M}^i_{k-1}$. A transformation $(R, \mathbf{t})$ can be applied to a line represented in Plücker coordinates with:

$$T_{line} = \begin{bmatrix} R & [\mathbf{t}]_\times R \\ 0_{3\times 3} & R \end{bmatrix} \tag{1}$$

B. Line Correspondences and Camera Pose Estimation

Lines are detected using the Line Segment Detector [22]. Lines that have a depth discontinuity, or whose endpoints belong to different semantic masks are culled.

Optical flow is employed to acquire line correspondences in consecutive frames in the same way point correspondences are found in [19]. This tackles a big problem often present in line-based SLAM systems that use line descriptors, as lines cannot be detected consistently between frames or are detected with different lengths. In the first case, the correspondent line is not found, while in the second one descriptors might not match due to different line appearance. Utilizing optical flow, we have achieved a higher number of line matches between frames ensuring long line tracklets.

An initial camera pose is estimated with a Perspective-n-Point algorithm in a RANSAC scheme, using only points that do not belong to objects. To refine this estimate, we propose a novel minimization problem, which concurrently optimizes the camera pose and optical flow, improving the initial point and line correspondences. Specifically, the following error term is proposed:

$$\mathbf{e}_{j,l} = \mathbf{e}_j(^0X_k, \boldsymbol{\phi}^{j,\mathbf{a}}_k, \boldsymbol{\phi}^{j,\mathbf{b}}_k) = \begin{bmatrix} \mathbf{l}^{j,obs}_k \cdot \boldsymbol{\pi}(^0X^{-1}_k \mathbf{A}^j_{k-1}) \\ \mathbf{l}^{j,obs}_k \cdot \boldsymbol{\pi}(^0X^{-1}_k \mathbf{B}^j_{k-1}) \end{bmatrix} \tag{2}$$

where $\mathbf{l}^{j,obs}_k$ is the observed infinite line given by:

$$\mathbf{l}^{j,obs}_k = \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \frac{\mathbf{a}^{j,obs}_k \times \mathbf{b}^{j,obs}_k}{||\mathbf{a}^{j,obs}_k \times \mathbf{b}^{j,obs}_k||}$$

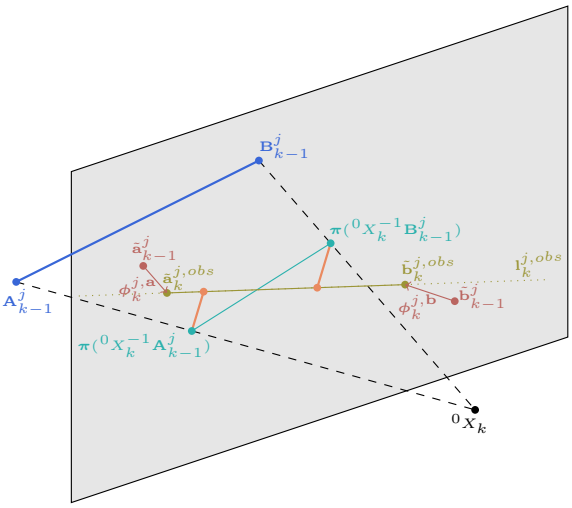

Fig. 3. **3D illustration of the line reprojection error term:** Line endpoints $\mathbf{A}^j_{k-1}$ and $\mathbf{B}^j_{k-1}$ project onto coordinate frame I_k at the endpoints $\boldsymbol{\pi}(^0X_k^{-1}\mathbf{A}^j_{k-1})$ and $\boldsymbol{\pi}(^0X_k^{-1}\mathbf{B}^j_{k-1})$ that define the reprojected line segment. Optical flows ($\boldsymbol{\phi}^{j,\mathbf{a}}_k$, $\boldsymbol{\phi}^{j,\mathbf{b}}_k$) and the endpoints of the line segment at frame $k-1$ ($\tilde{\mathbf{a}}^j_{k-1}$, $\tilde{\mathbf{b}}^j_{k-1}$) are added together to retrieve the observed endpoints of the corresponding line segment at frame k. Error term (2) corresponds to the orange lines and represents the distances of the reprojected line endpoints (Teal) from the corresponding observed infinite line (Olive).

$\boldsymbol{\pi}(\cdot)$ is the projective function returning a homogeneous vector, $\boldsymbol{\phi}^{j,\mathbf{a}}_k$ and $\boldsymbol{\phi}^{j,\mathbf{b}}_k$ are the optical flows corresponding to the start and end points of line j from coordinate frame I_{k-1} to I_k and $\tilde{\mathbf{a}}^{j,obs}_k = \tilde{\mathbf{a}}^j_{k-1} + \boldsymbol{\phi}^{j,\mathbf{a}}_k$, $\tilde{\mathbf{b}}^{j,obs}_k = \tilde{\mathbf{b}}^j_{k-1} + \boldsymbol{\phi}^{j,\mathbf{b}}_k$ the endpoints of the observed line in the current frame. This error term (2) consists of the stacked distances of the reprojected endpoints of line j at frame $k-1$ from the line defined by the observed corresponding endpoints at frame k (see Fig. 3). If the solution of the minimization problem results in an error term that exceeds a set threshold, the corresponding line is considered an outlier and is removed. This error term resembles that of [9], [13], however, in our proposal it is also dependent on the optical flow and new Jacobians had to be calculated (see Appendix).

The minimization problem, based on the reprojection errors of points and lines, is thus the following:

$$\begin{aligned} \{^0X^*_k, \boldsymbol{\Phi}^*_k\} = \underset{\{^0X_k, \boldsymbol{\Phi}_k\}}{\arg\min} \sum_i^{n_p} \{\boldsymbol{\rho}_h(\mathbf{e}^\top_{i,r}\Sigma^{-1}_{\boldsymbol{\phi}}\mathbf{e}_{i.r}) + \\ \boldsymbol{\rho}_h(\mathbf{e}^\top_{i,p}\Sigma^{-1}_p\mathbf{e}_{i,p})\} + \sum_j^{n_l} \{\boldsymbol{\rho}_h(\mathbf{e}^\top_{j,ra}\Sigma^{-1}_{\boldsymbol{\phi}}\mathbf{e}_{j,ra}) + \\ \boldsymbol{\rho}_h(\mathbf{e}^\top_{j,rb}\Sigma^{-1}_{\boldsymbol{\phi}}\mathbf{e}_{j,rb}) + \boldsymbol{\rho}_h(\mathbf{e}^\top_{j,l}\Sigma^{-1}_l\mathbf{e}_{j,l})\} \end{aligned} \tag{3}$$

where "*" denotes the optimal solution, n_p and n_l are the number of static point and line correspondences, $\mathbf{e}_{i,p}$ is the well-known reprojection error term for points [5], [19], [21], $\mathbf{e}_{i,r}$, $\mathbf{e}_{j,ra}$ and $\mathbf{e}_{j,rb}$ are regularization terms for the optical flows that correspond to the points [19], and the start and end points of lines, respectively, Σ_ϕ is the covariance matrix for the regularization error terms, and Σ_p and Σ_l are the covariance matrices associated with the reprojection error

terms of points and lines, respectively. The set $\mathbf{\Phi}_k$ contains all optical flow vectors from coordinate frame I_{k-1} to I_k that correspond to the points and line endpoints participating in the minimization problem. This problem is implemented using the g^{2}o library [23], and is solved with the iterative Levenberg-Marquardt algorithm.

C. Object Tracking and Motion Estimation

After determining the camera pose, optical flow is employed to correlate semantic masks of the same objects in consecutive frames. Subsequently, scene flow analysis is utilized to separate dynamic objects from static ones. Specifically, the estimated camera pose is used to align corresponding observations of consecutive frames, hence obtaining an approximation of point motions. Taking into consideration that scene flow should be negligible for static objects, those with a high number of points that do not meet this requirement are deemed as dynamic.

Once the dynamic objects are identified, their motion is estimated by slightly modifying the minimization problem of the previous subsection with the introduction of a similar error term to (2):

$$\mathbf{e}_{j,l} = \mathbf{e}_j({}_{k-1}{}^0G_k, \boldsymbol{\phi}_k^{j,\mathbf{a}}, \boldsymbol{\phi}_k^{j,\mathbf{b}}) = \begin{bmatrix} \mathbf{l}_k^{j,obs} \cdot \boldsymbol{\pi}({}_{k-1}{}^0G_k \mathbf{A}_{k-1}^j) \\ \mathbf{l}_k^{j,obs} \cdot \boldsymbol{\pi}({}_{k-1}{}^0G_k \mathbf{B}_{k-1}^j) \end{bmatrix} \tag{4}$$

where the variable to be estimated is ${}_{k-1}{}^0G_k = {}^0X_k^{-1}\ {}_{k-1}{}^0H_k$ and, thus, the minimization problem maintains the form of (3). It must be noted that even if a static object is initially incorrectly labeled as dynamic, during this stage it will be identified to have no relative motion and function as static.

D. Partial and Global Batch Optimization

A graph optimization formulation is proposed to jointly refine the trajectory of the camera, the motion of dynamic rigid objects, and the map which consists of points and lines on both static and dynamic objects. This graph encapsulates constraints on the variables to be estimated, in the form of error terms, which participate in a nonlinear least square problem, as in [23]. Specifically, two types of novel line constraints are proposed, (i) 3D line measurement constraints and (ii) constraints on the motion of lines that belong to dynamic rigid objects. The rest of the constraints, created by point and odometry observations, remain as in [19]. The novel constraints (i) and (ii) are presented as orange and magenta factors, respectively, in Fig. 4, which contains only **line** observations. Line representations have to be minimal in order to avoid numerical instability problems during their optimization and extra computational costs caused by extra degrees of freedom. Orthonormal representation [14] is chosen as a minimal representation for 3D lines.

The 3D line measurement error is defined as:

$$\mathbf{e}_{j,k}({}^0X_k, \mathscr{L}_k^j) = \begin{bmatrix} || {}^{C_k}\tilde{\mathbf{A}}_k^{j,obs} \times {}^{C_k}\tilde{\mathbf{U}}_k^j - {}^{C_k}\tilde{\mathbf{N}}_k^j || \\ || {}^{C_k}\tilde{\mathbf{B}}_k^{j,obs} \times {}^{C_k}\tilde{\mathbf{U}}_k^j - {}^{C_k}\tilde{\mathbf{N}}_k^j || \end{bmatrix} \tag{5}$$

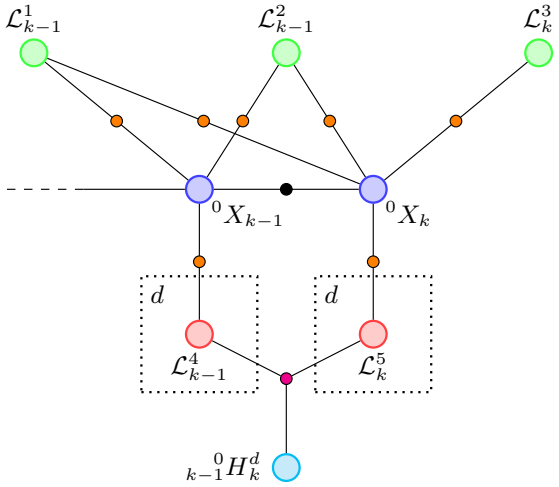

Fig. 4. **Factor graph representation for line landmarks:** Showcases only static and dynamic **line** features and the constraints imposed by them. Translucent Circles: 3D static lines (Green), poses (Blue), 3D dynamic lines (Red), object motion transform (Cyan). Opaque Circles: 3D line measurement constraints (Orange), constraints on the motion of lines that belong to dynamic objects d (Magenta), pose constraints (Black).

which represents the distances [25] of the observed 3D endpoints ${}^{C_k}\mathbf{A}_k^{j,obs}$, ${}^{C_k}\mathbf{B}_k^{j,obs}$ from the Plücker line j at frame k.

The following two notes are considered necessary. For static lines, subscript k of Plücker line elements ${}^{C_k}\tilde{\mathbf{U}}_k^j$ and ${}^{C_k}\tilde{\mathbf{N}}_k^j$ is chosen as the first frame line j was observed, whereas for dynamic lines it is actually the current frame k. Secondly, Plücker coordinates are used in the error calculation, however, the update parameters are calculated for the orthonormal representation of the lines, as can be seen in the calculation of the Jacobian with respect to the line parameters $\boldsymbol{\vartheta} = (\boldsymbol{\theta}, \theta)$ (see Appendix).

The constraint of motion of a line j that belongs to a dynamic rigid object d is divided into a distance and angular cost and is defined as:

$$\mathbf{e}_{j,d,k}(\mathscr{L}_k^j, {}_{k-1}{}^0H_k^d, \mathscr{L}_{k-1}^j) = \begin{bmatrix} \text{dist}(\mathscr{L}_k^j, \mathscr{L}_k^{j,H}) \\ 1 - \frac{\tilde{\mathbf{U}}_k^j \cdot \tilde{\mathbf{U}}_k^{j,H}}{||\tilde{\mathbf{U}}_k^j||\ ||\tilde{\mathbf{U}}_k^{j,H}||} \end{bmatrix} \tag{6}$$

where ${}_{k-1}{}^0H_k^d$ is the line motion transformation matrix for the object d. The superscript "^{H}" on a line j at frame k belonging to an object d is used to denote that it has undergone a motion transformation ${}_{k-1}{}^0H_k^d$:

$$\mathscr{L}_k^{j,H} = {}_{k-1}{}^0H_k^d \mathscr{L}_{k-1}^j = \begin{bmatrix} \tilde{\mathbf{N}}_k^{j,H} \\ \tilde{\mathbf{U}}_k^{j,H} \end{bmatrix}$$

Note that to simplify the notation for dynamic lines, we imply that they belong to an object d. The function dist is given by the formula for the distance of two Plücker lines:

$$\text{dist}(\mathscr{L}_k^j, \mathscr{L}_k^{j,H}) = \begin{cases} \frac{|\tilde{\mathbf{U}}_k^j \cdot \tilde{\mathbf{N}}_k^{j,H} + \tilde{\mathbf{N}}_k^j \cdot \tilde{\mathbf{U}}_k^{j,H}|}{||\tilde{\mathbf{U}}_k^j \times \tilde{\mathbf{U}}_k^{j,H}||} & \text{if } \tilde{\mathbf{U}}_k^j \times \tilde{\mathbf{U}}_k^{j,H} \neq 0 \\ \frac{||\tilde{\mathbf{U}}_k^j \times (\tilde{\mathbf{N}}_k^j - \tilde{\mathbf{N}}_k^{j,H}/s)||}{||\tilde{\mathbf{U}}_k^j||^2} & \text{if } \tilde{\mathbf{U}}_k^{j,H} = s\tilde{\mathbf{U}}_k^j \text{ for some } s \neq 0 \end{cases}$$

The Jacobians of (6) are discussed in the Appendix.

TABLE I

KITTI Raw Dataset results (E_t[m] and E_R[deg]). "–" = Result not reported in [20], "-" = Ground truth missing from [24]. *FO = Flow Optimization.

Sequence	Average Length of Static Line Tracklets		DynaSLAM II		VDO-SLAM				Ours (w. FO*)				Ours (wo. FO*)			
			Camera		Camera		Objects		Camera		Objects		Camera		Objects	
	w. FO*	wo. FO*	E_t	E_R	E_t	E_R	E_t	E_R	E_t	E_R	E_t	E_R	E_t	E_R	E_t	E_R
0926-0001	5.1	3.1	–	–	0.051	**0.056**	0.410	0.439	**0.050**	**0.056**	**0.353**	**0.423**	0.051	0.056	0.450	0.426
0926-0002	5.1	3.0	–	–	0.061	0.067	**0.178**	1.528	**0.055**	**0.066**	0.490	**0.674**	0.055	0.066	0.425	1.142
0926-0005	6.2	3.0	–	–	0.059	0.083	**0.378**	1.988	**0.051**	**0.071**	0.462	**1.799**	0.054	0.071	0.264	1.878
0926-0009	5.6	3.1	1.870	0.573	0.110	**0.065**	0.217	0.188	**0.095**	0.066	**0.211**	**0.165**	0.101	0.060	0.211	0.164
0926-0011	8.0	3.1	–	–	0.043	**0.057**	0.623	1.169	**0.034**	**0.057**	**0.265**	**0.325**	0.037	0.057	0.593	0.816
0926-0013	4.6	3.2	0.930	**0.000**	0.076	0.059	**0.139**	0.390	**0.074**	0.058	1.465	**0.355**	0.079	0.058	1.465	0.369
0926-0014	4.8	3.3	1.350	0.573	**0.108**	0.070	0.988	**2.853**	0.110	**0.069**	**0.811**	3.060	0.110	0.069	0.811	3.229
0926-0051	8.3	3.1	1.140	**0.000**	0.065	0.058	1.067	1.029	**0.061**	0.058	**0.644**	**0.415**	0.072	0.059	0.644	0.416
0926-0091	6.1	3.1	–	–	0.069	0.063	-	-	**0.066**	**0.062**	-	-	0.067	0.062	-	-
0926-0093	6.7	3.0	–	–	2.295	0.085	0.869	1.207	**2.284**	**0.084**	**0.669**	**0.391**	2.285	0.083	0.672	0.393
0926-0101	5.2	3.3	15.020	2.292	**0.570**	**0.072**	-	-	0.585	0.073	-	-	0.647	0.078	-	-
0926-0106	6.9	3.0	–	–	0.047	0.062	-	-	**0.039**	**0.058**	-	-	0.033	0.057	-	-
0929-0004	5.4	3.1	1.410	0.573	0.071	0.058	-	-	**0.065**	**0.057**	-	-	0.062	0.057	-	-

IV. Experimental Evaluation

To verify the performance of our proposed system we tested both outdoor and indoor scenarios. Specifically, the two following datasets were used: (i) KITTI Raw Dataset [24] and (ii) the Oxford Multimotion Dataset (OMD) [26]. In the results, we report and compare the accuracy of both the camera's egomotion and all dynamic rigid objects' poses. Adding line segments naturally increases computational complexity compared to VDO-SLAM, but our system still achieves close to real-time performance.

A. Preprocessing

For the semantic segmentation in KITTI Raw Dataset, an implementation of Mask R-CNN with pre-trained weights for MS COCO is used. For the OMD dataset, a simple color-based HSV segmentation method implemented by us is used, followed by morphological filtering for refinement.

The dense optical flow is retrieved by the PyTorch version of PWC-Net model without fine-tuning the weights [27], [28].

B. Error Metrics

To compare our results directly with VDO-SLAM, the error metric provided in their paper and implementation is used [19]. For each frame, the error is defined as $E = \hat{T}^{-1}T$, where $\hat{T}$ is the estimated motion transform for either the camera or an object and T is the corresponding ground truth motion. The translational error E_t is the L_2 norm of the translational part of E, while E_R is the rotation angle in an axis-angle representation of the rotational component of E.

C. KITTI Raw Dataset

The KITTI Raw Dataset consists of many sequences in real outdoor driving environments with given ground truth camera and object poses. To test our system in a variety of environments, a set of 13 sequences with different levels of dynamicity and geometric presence were chosen. The results of our proposed system are presented in Table I. We compare the effectiveness of our system, which enhances the point-only approach of VDO-SLAM by incorporating line segments, against VDO-SLAM itself and the reported results of DynaSLAM II [20], which are both considered state-of-the-art dynamic SLAM systems.

Regarding the camera's egomotion, our implementation outperforms the other two systems in almost all sequences in E_t, while it is on par or better in E_R. DynaSLAM II seems to achieve a lower rotational error in two sequences, however, it must be noted that its authors provided these results in radians, resulting in a loss of decimal accuracy when they are transformed into degrees.

To conduct a more comprehensive analysis, we have incorporated in our table of results a metric corresponding to the average number of frames static lines are tracked in. Our system demonstrates the most significant improvement in sequences 0926-(0009, 0011, 0093, 0005, 0106), which have, except 0926-0011, a strong presence of nearby buildings providing a lot of high-quality line segments for detection. This translates directly to higher values in the aforementioned metric, underscoring the importance of high-quality lines that provide consistent tracking. Interestingly, the significantly improved performance in sequence 0926-0011 is justified, despite the absence of nearby buildings, through the exhibition of one of the highest metric values (8.0). Conversely, our system underperforms slightly in sequences 0926-0014 and 0926-0101, which are characterized by open spaces and lack of buildings. Line features are mostly detected on the road and on tree leaves that are located far from the camera, thus causing a degradation in the results. This is reflected in the metric values of these sequences, with their average static line tracklet length (4.8 and 5.2) being significantly below the overall average (6), highlighting the correlation between low-quality line features and reduced accuracy.

Regarding the tracking accuracy of dynamic objects, the inclusion of lines enhances the results in the majority of sequences tested, which may be attributed to most dynamic objects being automobiles that provide a lot of line segments for detection in parts such as windows and license plates. The only sequences in which our implementation does not improve are 0926-(0002, 0005, 0013). A detailed qualitative analysis revealed, that in two of these (0926-0002, 0926-

TABLE II
OMD results (E_t[m] and E_R[deg]).

	VDO-SLAM		Ours	
	E_t	E_R	E_t	E_R
Full Sequence: Camera	0.038	0.578	**0.022**	**0.507**
Full Sequence: Box Mean	0.032	1.286	**0.029**	**1.231**
500 frames: Camera	0.017	0.466	**0.014**	**0.453**
500 frames: Top Right	0.033	1.369	**0.032**	**1.367**
500 frames: Bottom Right	0.030	1.166	**0.029**	**1.164**
500 frames: Top Left	0.036	1.494	**0.031**	**1.452**
500 frames: Bottom Left	**0.027**	**1.601**	**0.027**	1.605
500 frames: Box Mean	0.032	1.407	**0.030**	**1.397**

0005), the majority of the dynamic objects detected and tracked are moving bikes with humans, which do not follow the underlying rigidity assumption. Therefore, lines inside the bicycle wheels or lines at the feet of the cyclists contribute to the deterioration of the results in these two cases. However, it must be highlighted that even in these, the object E_R is improved greatly.

Finally, to assess the impact of optical flow optimization on system accuracy, we have conducted an ablation study (see last column of Table I) through modifications in (2) and (4), by excluding the optical flow dependence from the error terms. This resulted in less consistent line segment matches, with a clear decrease in the average length of static line tracklets and a performance deterioration in both camera E_t and object E_R metrics. The fact that sequences 0926-0002 and 0926-0005 perform worse in object pose accuracy when the flow is concurrently optimized, actually supports the findings of the previous paragraph, since more line correspondences in nonrigid objects are retained, therefore magnifying the problem.

D. Oxford Multimotion Dataset

The Oxford Multimotion Dataset consists of frame sequences captured in an indoor environment with moving toy cars or levitating cubes. This dataset is characterized by a strong geometric structure, as both the static environment and the moving cubes provide many quality line segments for detection; an ideal scenario to showcase the effect of lines. System performance is evaluated exclusively in the swinging box sequence, and specifically in the unconstrained camera movement case, a challenging realistic scenario. We tested our system both in the initial 500 frames for comparison with [19], as well as in the entirety of frames to evaluate our system's robustness in a long-running sequence.

As shown in Table II, our system outperforms VDO-SLAM both in egomotion and four moving boxes' pose accuracy, which is a natural outcome considering that the test is run indoors and dynamic objects in the sequence are cubes. Namely, in the full sequence (and in the first 500 frames), a $\sim$42% ($\sim$18%) and $\sim$12% ($\sim$2.8%) improvement is achieved in camera E_t and E_R, respectively, compared to VDO-SLAM. Furthermore, the inclusion of lines enhanced the accuracy of the moving boxes' pose estimation in the full sequence and had marginal improvements in the first 500 frames, reducing their average E_t by $\sim$9.4% ($\sim$6.3%) and E_R by $\sim$4.3% ($\sim$0.7%).

E. Result Summary

We demonstrated that the inclusion of lines resulted in an enhancement to overall performance, on both egomotion and dynamic object tracking, in outdoor driving (Table I) and indoor (Table II) scenarios. Additionally, we introduced the average length of static line tracklets, which quantifies the quality and robustness of line segments. A thorough analysis verified a high correlation between this metric and the improvement in accuracy of our implementation compared to the other state-of-the-art systems. The utilization of optical flow for line matching provides better and more line correspondences, resulting in long-lasting and consistent line tracklets, a benefit that was proved to be further amplified by the concurrent optimization of optical flow in the tracking stage.

V. Conclusions

In this work, we have presented a novel SLAM system, which exploits line features detected on both static and dynamic objects, in order to estimate camera trajectory and object motions. We have demonstrated our contributions with novel optimization formulations, which employed line observations to refine camera tracking, dynamic object trajectories, and static and dynamic feature positions in the map. Our experimental evaluation showed that leveraging the line structure of the environment resulted in an overall increase in the accuracy and robustness of the SLAM algorithm compared to other state-of-the-art point-based dynamic systems. In future work, we seek to address the existence of humans who significantly contribute to nonrigidity within dynamic environments, by extending our implementation to handle the independent movements of their linear skeleton parts.

Appendix

Jacobian of the Line Reprojection Error Term

The Jacobian of Eq. (2) with respect to the optical flow of the start point (similarly for the end point) is:

$$\frac{\partial \mathbf{e}_{j,l}}{\partial \boldsymbol{\phi}_k^{j,\mathbf{a}}} = \boldsymbol{\pi}({}^0X_k^{-1}\mathbf{A}_{k-1}^j)^\top \frac{\partial \mathbf{l}_k^{j,obs}}{\partial \boldsymbol{\phi}_k^{j,\mathbf{a}}}$$

where $\frac{\partial \mathbf{l}_k^{j,obs}}{\partial \boldsymbol{\phi}_k^{j,\mathbf{a}}}$ is calculated analytically with a symbolic language, and with respect to the pose parameters $\boldsymbol{\Xi}_k$ is:

$$\frac{\partial \mathbf{e}_{j,l}}{\partial \boldsymbol{\Xi}_k} = \begin{bmatrix} \begin{bmatrix}\lambda_0\\ \lambda_1\end{bmatrix}^\top \frac{\partial \boldsymbol{\pi}({}^0X_k, \mathbf{A}_{k-1}^j)}{\partial \boldsymbol{\Xi}_k} \\ \begin{bmatrix}\lambda_0\\ \lambda_1\end{bmatrix}^\top \frac{\partial \boldsymbol{\pi}({}^0X_k, \mathbf{B}_{k-1}^j)}{\partial \boldsymbol{\Xi}_k} \end{bmatrix}$$

The Jacobians $\frac{\partial \boldsymbol{\pi}({}^0X_k, \mathbf{A}_{k-1}^j)}{\partial \boldsymbol{\Xi}_k}$ and $\frac{\partial \boldsymbol{\pi}({}^0X_k, \mathbf{B}_{k-1}^j)}{\partial \boldsymbol{\Xi}_k}$ can be found in [15], [29], in which the reader may gain a deeper insight into the mathematical background.

Jacobian of 3D Line Measurement Errors

The Jacobian of Eq. (5) with respect to orthonormal line parameters $\boldsymbol{\vartheta}_k^j$ of $\mathscr{L}_k^j$ can be broken down via chain rule:

$$\frac{\partial \mathbf{e}_{j,k}({}^0X_k, \mathscr{L}_k^j)}{\partial \boldsymbol{\vartheta}_k^j} = \frac{\partial \mathbf{e}_{j,k}({}^0X_k, \mathscr{L}_k^j)}{\partial\, {}^{C_k}\mathscr{L}_k^j} \cdot \frac{\partial\, {}^{C_k}\mathscr{L}_k^j}{\partial \mathscr{L}_k^j} \cdot \frac{\partial \mathscr{L}_k^j}{\partial \boldsymbol{\vartheta}_k^j}$$

The first factor can be calculated analytically, the second factor is equal to a transform (see Eq. (1)) that converts the Plücker line from the local coordinate system of frame k to the global reference frame, and the third factor is known [15].

Jacobians of Motion of Lines Errors

The Jacobian of Eq. (6) with respect to $\boldsymbol{\vartheta}_k^j$ is computed as:

$$\frac{\partial \mathbf{e}_{j,d,k}(\mathscr{L}_k^j, {}_{k-1}{}^0H_k^d, \mathscr{L}_{k-1}^j)}{\partial \boldsymbol{\vartheta}_k^j} = \frac{\partial \mathbf{e}_{j,d,k}(\mathscr{L}_k^j, {}_{k-1}{}^0H_k^d, \mathscr{L}_{k-1}^j)}{\partial \mathscr{L}_k^j} \cdot \frac{\partial \mathscr{L}_k^j}{\partial \boldsymbol{\vartheta}_k^j}$$

The calculation of the first factor, apart from the fact that dist is a piecewise function, is quite straightforward and is conducted by differentiating with respect to Plücker elements.

The Jacobian with respect to parameters $\boldsymbol{\vartheta}_{k-1}^j$ can also be broken down into simpler factors:

$$\frac{\partial \mathbf{e}_{j,d,k}(\mathscr{L}_k^j, {}_{k-1}{}^0H_k^d, \mathscr{L}_{k-1}^j)}{\partial \boldsymbol{\vartheta}_{k-1}^j} = \frac{\partial \mathbf{e}_{j,d,k}(\mathscr{L}_k^j, {}_{k-1}{}^0H_k^d, \mathscr{L}_{k-1}^j)}{\partial \mathscr{L}_k^{j,H}} \cdot \frac{\partial \mathscr{L}_k^{j,H}}{\partial \mathscr{L}_{k-1}^j} \cdot \frac{\partial \mathscr{L}_{k-1}^j}{\partial \boldsymbol{\vartheta}_{k-1}^j}$$

Finally, the Jacobian with respect to pose parameters $\boldsymbol{\Xi}_k^d$ of ${}_{k-1}{}^0H_k^d$ can be broken down into the following factors:

$$\frac{\partial \mathbf{e}_{j,d,k}(\mathscr{L}_k^j, {}_{k-1}{}^0H_k^d, \mathscr{L}_{k-1}^j)}{\partial \boldsymbol{\Xi}_k^d} = \frac{\partial \mathbf{e}_{j,d,k}(\mathscr{L}_k^j, {}_{k-1}{}^0H_k^d, \mathscr{L}_{k-1}^j)}{\partial \mathscr{L}_k^{j,H}} \cdot \frac{\partial \mathscr{L}_k^{j,H}}{\partial \boldsymbol{\Xi}_k^d}$$

REFERENCES

[1] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[2] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping," in *Proc. of IEEE Intl. Conf. on Robotics and Automation*, Paris, France, May-Aug. 2020.

[3] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1029–1036, Feb. 2023.

[4] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *Proc. of IEEE/ACM Intl. Symposium on Mixed and Augmented Reality*, Nara, Japan, Nov. 2007.

[5] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[6] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments," in *Proc. of IEEE Intl. Conf. on Robotics and Automation*, Brisbane, Australia, May 2018.

[7] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *Proc. of European Conf. on Computer Vision*, Zurich, Switzerland, Sep. 2014.

[8] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *Proc. of Intl. Conf. on Computer Vision*, Barcelona, Spain, Nov. 2011.

[9] R. Gomez-Ojeda, F.-A. Moreno, D. Zuniga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM: A Stereo SLAM System through the Combination of Points and Line Segments," *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 734–746, Jun. 2019.

[10] M. Kaess, "Simultaneous Localization and Mapping with Infinite Planes," in *Proc. of IEEE Intl. Conf. on Robotics and Automation*, Seattle (WA), USA, May 2015.

[11] W. Zhen, H. Yu, Y. Hu, and S. Scherer, "Unified Representation of Geometric Primitives for Graph-SLAM Optimization Using Decomposed Quadrics," in *Proc. of Intl. Conf. on Robotics and Automation*, Philadelphia (PA), USA, May 2022.

[12] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[13] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "PL-SLAM: Real-Time Monocular Visual SLAM with Points and Lines," in *Proc. of IEEE Intl. Conf. on Robotics and Automation*, Singapore, May-Jun. 2017.

[14] A. Bartoli and P. Sturm, "Structure-From-Motion Using Lines: Representation, Triangulation, and Bundle Adjustment," *Computer Vision and Image Understanding*, vol. 100, no. 3, pp. 416–441, Dec. 2005.

[15] X. Zuo, X. Xie, Y. Liu, and G. Huang, "Robust Visual SLAM with Point and Line Features," in *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Vancouver, Canada, Sep. 2017.

[16] B. Bescos, J. M. Facil, J. Civera, and J. Neira, "DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, Oct. 2018.

[17] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proc. of Intl. Conf. on Computer Vision*, Venice, Italy, Oct. 2017.

[18] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments," in *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Madrid, Spain, 2018.

[19] J. Zhang, M. Henein, R. Mahony, and V. Ila, "VDO-SLAM: A Visual Dynamic Object-aware SLAM System," *arXiv:2005.11052*, May 2020.

[20] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "DynaSLAM II: Tightly-Coupled Multi-Object Tracking and SLAM," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5191–5198, Jul. 2021.

[21] Y. Qiu, C. Wang, W. Wang, M. Henein, and S. Scherer, "AirDOS: Dynamic SLAM benefits from Articulated Objects," in *Proc. of IEEE Intl. Conf. on Robotics and Automation*, Philadelphia (PA), USA, May 2022.

[22] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A Fast Line Segment Detector with a False Detection Control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722–732, Apr. 2008.

[23] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g^{2}o: A General Framework for Graph Optimization," in *Proc. of IEEE Intl. Conf. on Robotics and Automation*, Shanghai, China, May 2011.

[24] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[25] T. Brox, B. Rosenhahn, J. Gall, and D. Cremers, "Combined Region- and Motion-based 3D Tracking of Rigid and Articulated Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 402–415, Mar. 2009.

[26] K. M. Judd and J. D. Gammell, "The Oxford Multimotion Dataset: Multiple SE(3) Motions With Ground Truth," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 800–807, Apr. 2019.

[27] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume," in *Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, Salt Lake City (UT), USA, Jun. 2018.

[28] S. Niklaus, "A Reimplementation of PWC-Net Using PyTorch," https://github.com/sniklaus/pytorch-pwc, Jul. 2018.

[29] J. L. Blanco-Claraco, "A tutorial on $\mathbf{SE}(3)$ transformation parameterizations and on-manifold optimization," *arXiv:2103.15980*, Sep. 2010.