

# VDO-SLAM

## 论文情况

- 标题: VDO-SLAM: A Visual Dynamic Object-aware SLAM System
- 作者: Jun Zhang, Mina Henein, Robert Mahony and Viorela Ila
- 网站: [arXiv:2005.11052 \[cs.RO\]](https://arxiv.org/abs/2005.11052)
- 源码: [https://github.com/halajun/VDO\\_SLAM](https://github.com/halajun/VDO_SLAM)
  - 教程-1: <https://blog.csdn.net/slzlincent/article/details/108380252>
  - 教程-2: [https://blog.csdn.net/weixin\\_43946315/article/details/120461310](https://blog.csdn.net/weixin_43946315/article/details/120461310)

## 1 Introduction

多数现有 SLAM 的特点:

- 假设环境静止。
- 对待动态物体:
  - 将传感器检测到的动态物体作为 outliers 从预测过程中移除, 构建只包含静态特征的地图。
  - 使用传统多目标追踪单独追踪动态物体, 精确度依赖于相机位姿估计, 复杂环境中容易建图失败。

本文研究:

- 将 SLAM 中的 mapping 重新定义为 spatiotemporal representation of the world, 这与传统 SLAM 对静态地图的侧重不同。
- 本文的方法注重场景中动态物体的准确运动估计, 这些与机器人在动态环境的路径

规划和导航高度相关。

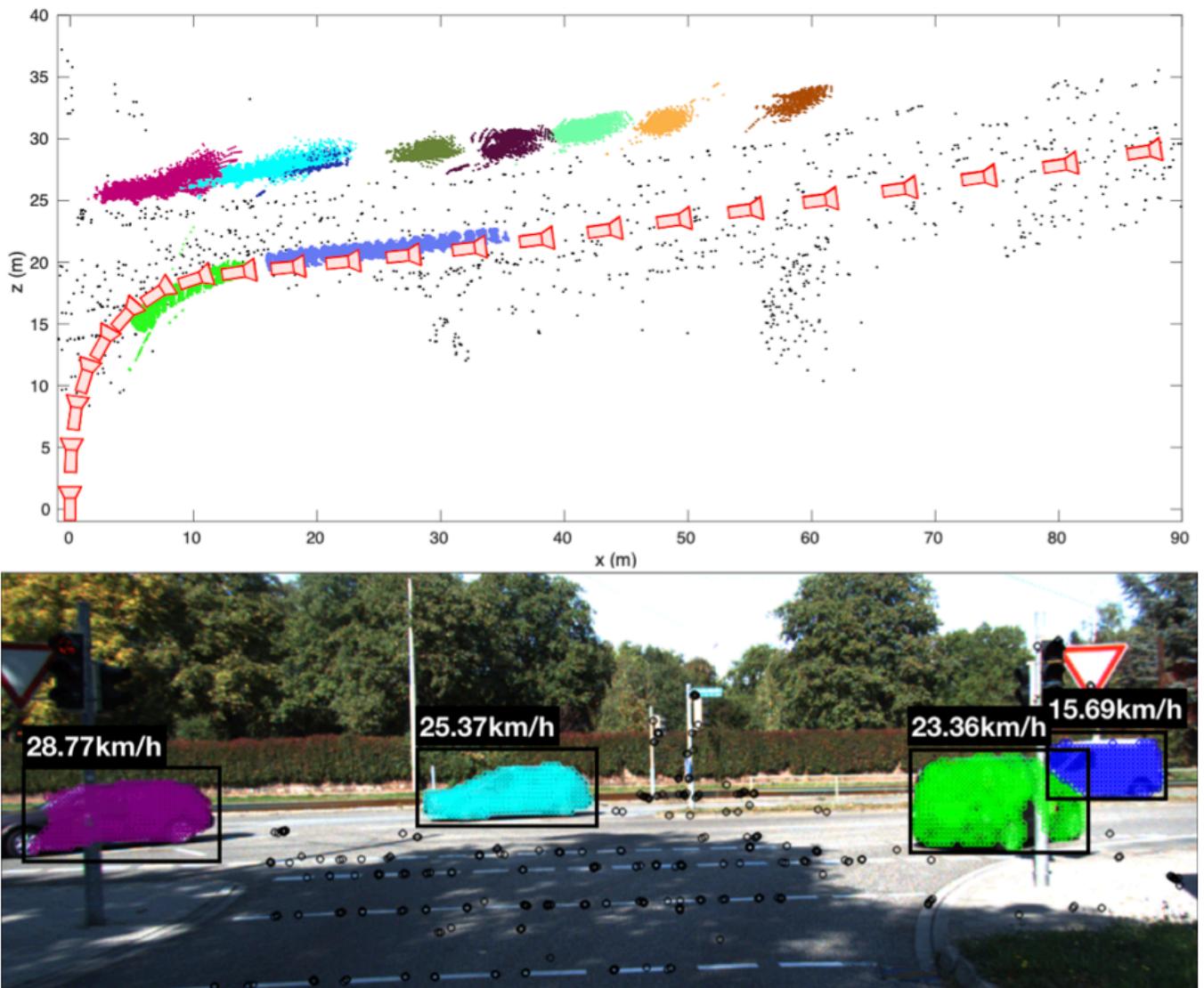
现有的运动估计（motion estimation）：

- optical flow estimation: optical flow 通过测算图像平面上与亮度相关的速度来记录场景运动。
- scene flow estimation: scene flow 描述不同时刻观测到的场景的 3D 运动场。
- 缺陷：仅计算单个像素或 3D 点的线性平移，没有使用刚体上的集体行为（collective behaviour）点，不能得到刚体的 SE(3) 位姿。
- 本文：利用单个物体上的点的集体行为获得物体的运动，并完成机器人的 SLAM。

深度学习在特征提取和追踪方面表现优越，而为了地图的丰富性就需要添加一些线特征和面特征。因此，为了融入这些信息，需要：

- 要么场景中每个物体的 3D 模型为先验；
- 要么前端可以显示给出物体的位姿信息。

这些增加了问题的复杂性。因此，需要利用 DL 的优越，不依赖于额外位姿估计和模型先验的，在特征级别上的算法。



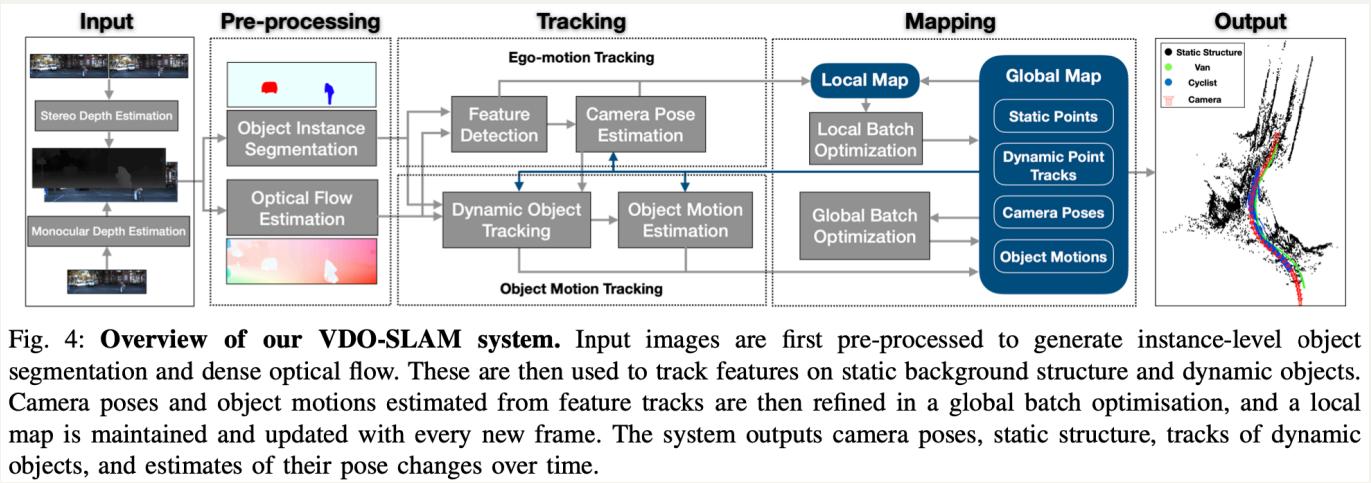
**Fig. 1: Results of our VDO-SLAM system.** (Top) A full map including camera trajectory in red, static background points in black and points on moving objects colour coded by their instance. (Bottom) Detected 3D points on the static background and the objects' body, and the estimated object speed. Black circles represents static points, and each object is shown with a different colour.

本文结合前作 [1] 和 [2], 提出 VDO-SLAM —— 基于特征的 stereo/RGB-D 动态 SLAM, 使用基于图像的语义信息来定位机器人、构建静态和动态结构、跟踪场景中运动的刚体。主要贡献:

- 在一个统一的框架中对机器人位姿、静态和动态 3D 点以及物体运动进行场景建模;
- 对动态物体超越 SOTA 对精确 SE(3) 位姿估计, 提取场景中动态物体的速度;

- 利用语义信息跟踪运动物体。

## 2 Methodology



- 首先，介绍变量及其含义；
- 然后，介绍了如何在系统的 tracking 部分估计相机的位姿和物体运动；
- 最后，提出了一种因子图优化方法，并将其应用于系统的 mapping 中，对相机位姿和物体运动进行优化，构建包含静态和动态结构的全局一致性映射。

### 2.1 Background and Notation

#### 2.1.1 Coordinate Frames

设  ${}^0X_k, {}^0L_k \in SE(3)$  为时刻  $k$  时机器人（相机）和物体在世界坐标系 0 下的位姿。Fig.2 表示了这些位姿关系：

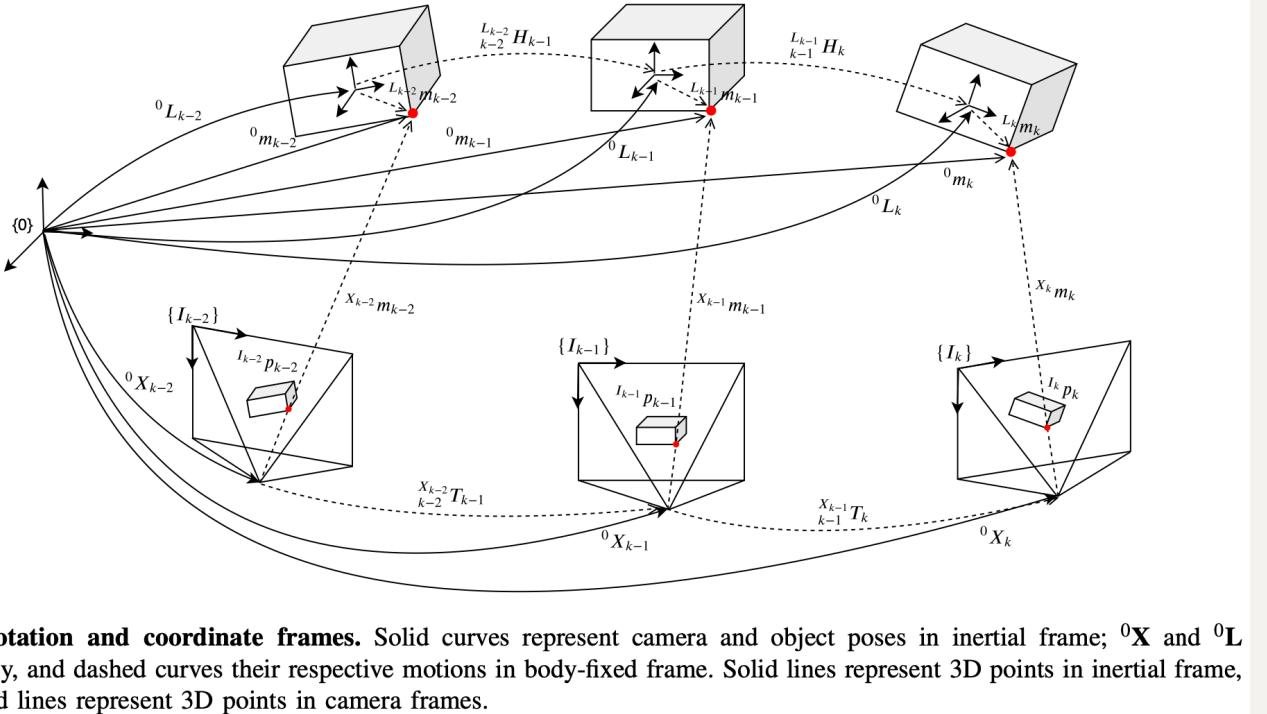


Fig. 2: **Notation and coordinate frames.** Solid curves represent camera and object poses in inertial frame;  ${}^0\mathbf{X}$  and  ${}^0\mathbf{L}$  respectively, and dashed curves their respective motions in body-fixed frame. Solid lines represent 3D points in inertial frame, and dashed lines represent 3D points in camera frames.

## 2.1.2 Points

设  ${}^0m_k^i = [m_x^i, m_y^i, m_z^i, 1] \in \mathbb{R}^3$  为时刻  $k$  时第  $i$  个 3D 点的齐次坐标。将一个点在机器人的坐标系下的坐标写为  ${}^{X_k}m_k^i = {}^0X_k^{-1}{}^0m_k^i$ 。

定义  $I_k$  为  $k$  时刻的像素坐标系，取左上角为原点，设  ${}^{I_k}p_k^i = [u^i, v^i, 1] \in \mathbb{R}^2$  为 3D 点  ${}^{X_k}m_k^i$  投影到  $I_k$  上的像素坐标，其通过投影函数  $\pi(\cdot)$  或相机内参矩阵  $K$  转换得到：

$${}^{I_k}p_k^i = \pi({}^{X_k}m_k^i) = K{}^{X_k}m_k^i \quad (1)$$

相机和/或物体的运动都会产生光流  ${}^{I_k}\phi^i \in \mathbb{R}^2$ ，表示像素  ${}^{I_k}p_k^i$  从图像  $I_{k-1}$  到  $I_k$  的运动：

$${}^{I_k}\phi^i = {}^{I_k}\tilde{p}_k^i - {}^{I_{k-1}}p_{k-1}^i \quad (2)$$

其中  ${}^{I_k}\tilde{p}_k^i$  表示  ${}^{I_{k-1}}p_{k-1}^i$  在  $I_k$  中的对应像素。本文使用光流来寻找连续帧之间的关系。

## 2.1.3 Object and 3D Point Motions

$k-1$  到  $k$  之间的物体运动表示为 (Fig.2 中的虚线则表示了这些运动) :

$${}_{k-1}^{L_{k-1}}H_k = {}^0L_{k-1}^{-1}{}^0L_k \quad (3)$$

将物体坐标系下的点表示为 (Fig.2 中红点间虚线上的向量, 应该说是落在物体上的点)  
 ${}^L m_k^i = {}^0L_k^{-1}{}^0m_k^i$ , 替换 (3) 中的对应部分, 得到

$${}^0m_k^i = {}^0L_k {}^L m_k^i = {}^0L_{k-1} {}_{k-1}^{L_{k-1}}H_k {}^L m_k^i \quad (4)$$

对于刚体而言,  ${}^L m_k^i$  是常量, 即  ${}^L m_k^i = {}^L m^i$ , 从而有  
 ${}^L m^i = {}^0L_k^{-1}{}^0m_k^i = {}^0L_{k+n}^{-1}{}^0m_{k+n}^i, n \in \mathbb{Z}$ 。对于刚体且  $n = -1$ , (4) 变为:

$${}^0m^i = {}^0L_{k-1} {}_{k-1}^{L_{k-1}}H_k {}^0L_{k-1}^{-1}{}^0m_{k-1}^i \quad (5)$$

(5) 将运动刚体上的同一个点在连续时间上通过同质变换

${}_{k-1}^0H_k = {}^0L_{k-1} {}_{k-1}^{L_{k-1}}H_k {}^0L_{k-1}^{-1} \in SE(3)$  联系起来。这个等式表达了位姿变换的坐标系变化。因此运动中的点在世界坐标系下可以表述为:

$${}^0m_k^i = {}_{k-1}^0H_k {}^0m_{k-1}^i \quad (6)$$

(6) 为运动估计方法的核心, 其以无模型的方式表达了刚体上的点的位姿变化, 而不需要将刚体的 3D 位姿作为随机变量包含在计算中。

## 2.2 Camera Pose and Object Motion Estimation

### 2.2.1 Camera Pose Estimation

给定  $k-1$  时刻在世界坐标系下观测到的一个静态 3D 点集  $\{{}^0m_{k-1}^i\}$ , 其对应在  $I_k$  图像上的 2D 点集为  $\{{}^{I_k}\tilde{p}_k^i\}$ , 相机位姿  ${}^0X_k$  通过最小化重投影误差得到:

$$e_i({}^0X_k) = {}^{I_k}\tilde{p}_k^i - \pi({}^0X_k^{-1}{}^0m_k^i) \quad (7)$$

使用李代数  $x_k \in se(3)$  对  $SE(3)$  位姿参数化:

$${}^0X_k = \exp({}^0x_k) \quad (8)$$

定义  ${}^0x_k^\vee \in \mathbb{R}^6$ ,  $\vee$  将  $se(3)$  映射到  $\mathbb{R}^6$ 。此时上述 (7) (8) 变成了如下的最小二乘:

$${}^0x_k^{*\vee} = \underset{{}^0x_k^\vee}{\operatorname{argmin}} \sum_i^{n_b} \rho_h(e_i^\top({}^0x_k) \Sigma_p^{-1} e_i({}^0x_k)) \quad (9)$$

$\rho_h$  为 Huber 函数,  $\Sigma_p$  为和重投影误差相关的协方差。预测的相机位姿通过 L-M 算法求解并表示为  ${}^0X_k^* = \exp({}^0x_k^*)$ 。

## 2.2.2 Object Motion Estimation

类似于相机位姿估计, 用一个基于重投影误差的损失函数来求解  ${}_{k-1}^0H_k$ 。使用式子 (6), 一个 3D 点和图像  $I_k$  上对应的 2D 点之间的误差为:

$$\begin{aligned} e_i({}_{k-1}^0H_k) &= {}^{I_k}\tilde{p}_k^i - \pi({}^0X_k^{-1}{}_{k-1}^0H_k {}^0m_{k-1}^i) \\ &= {}^{I_k}\tilde{p}_k^i - \pi({}_{k-1}^0G_k {}^0m_{k-1}^i) \end{aligned} \quad (10)$$

其中  ${}_{k-1}^0G_k \in SE(3)$ 。参数化  ${}_{k-1}^0G_k = \exp({}_{k-1}^0g_k)$ , 其中  ${}_{k-1}^0g_k \in se(3)$ , 最优解通过给定  $k-1$  和  $k$  时刻动态物体上的  $n_d$  个 3D-2D 动态点, 然后优化以下函数得到:

$${}_{k-1}^0g_k^{*\vee} = \underset{{}_{k-1}^0g_k^\vee}{\operatorname{argmin}} \sum_i^{n_d} \rho_h(e_i^\top({}_{k-1}^0g_k^\vee) \Sigma_p^{-1} e_i({}_{k-1}^0g_k^\vee)) \quad (11)$$

然后物体运动通过  ${}_{k-1}^0H_k = {}^0X_k {}_{k-1}^0G_k$  得到。

## 2.2.3 Joint Estimation with Optical Flow

为了确保对点的跟踪的鲁棒性, 联合运动估计来估计光流。对于相机位姿估计, 考虑式子 (2), (7) 中的误差重新构建为:

$$e_i({}^0X_k, {}^{I_{k-1}}\phi) = {}^{I_k}p_{k-1}^i + {}^{I_k}\phi^i - \pi({}^0X_k^{-1}{}^0m_{k-1}^i) \quad (12)$$

使用李代数参数化  $SE(3)$  元素, 最优解通过以下式子获取:

$$\begin{aligned} \{{}^0x_k^{*\vee}, {}^{I_k}\Phi_k^*\} = \underset{\{{}^0x_k^{*\vee}, {}^{I_k}\Phi_k\}}{\operatorname{argmin}} \sum_i^{n_b} & \left\{ \rho_h(e_i^\top({}^{I_k}\phi^i)\Sigma_\phi^{-1}e_i({}^{I_k}\phi^i)) \right. \\ & \left. + \rho_h(e_i^\top({}^0x_k, {}^{I_k}\phi^i)\Sigma_p^{-1}e_i({}^0x_k, {}^{I_k}\phi^i)) \right\} \end{aligned} \quad (13)$$

其中  $\rho_h(e_i^\top({}^{I_k}\phi^i)\Sigma_p^{-1}e_i({}^{I_k}\phi^i))$  为规范化项:

$$e_i({}^{I_k}\hat{\phi}^i) = {}^{I_k}\hat{\phi}^i - {}^{I_k}\phi^i \quad (14)$$

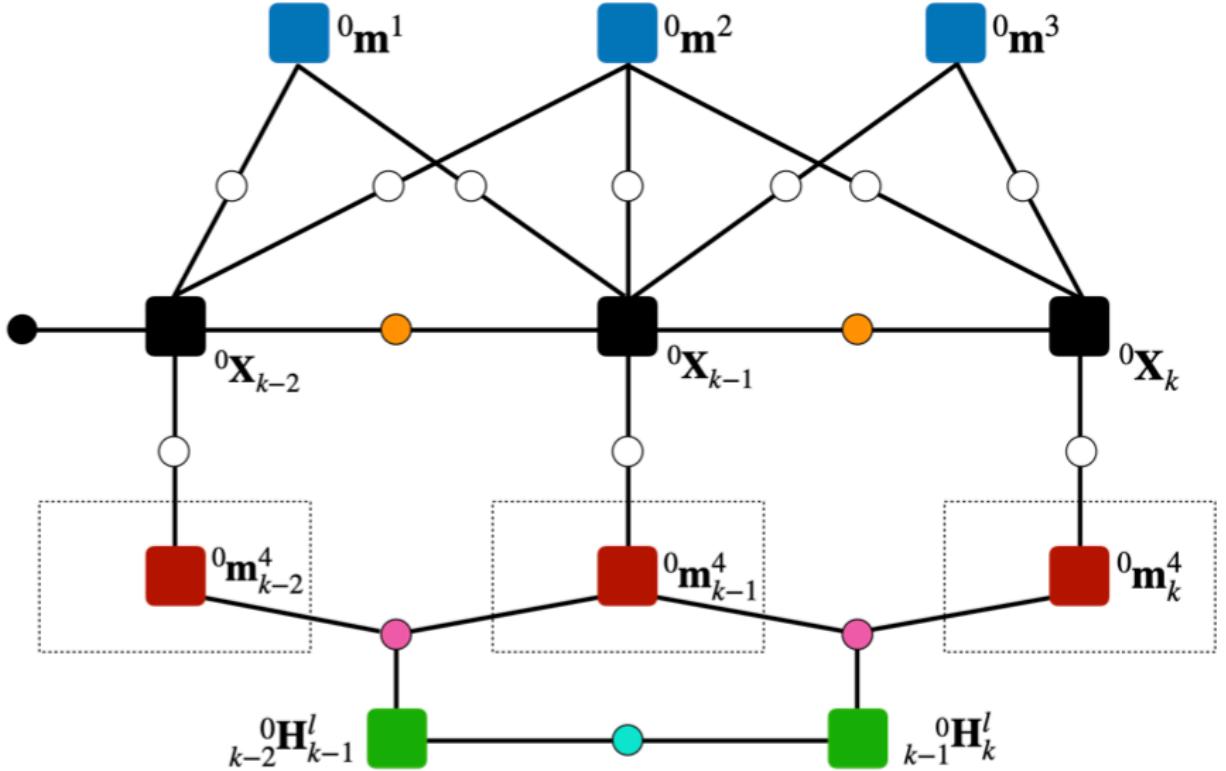
其中  ${}^{I_k}\hat{\Phi}^i = \{{}^{I_k}\hat{\phi}^i\}$  为通过经典算法或者机器学习获得的初始光流,  $\Sigma_\phi$  为协方差矩阵。

同理, (11) 中物体运动估计的损失函数重新定义为:

$$\begin{aligned} \{{}_{k-1}^0g_k^{*\vee}, {}^{I_k}\Phi_k^*\} = \underset{\{{}_{k-1}^0g_k^{*\vee}, {}^{I_k}\Phi_k\}}{\operatorname{argmin}} \sum_i^{n_d} & \left\{ \rho_h(e_i^\top({}^{I_k}\phi^i)\Sigma_\phi^{-1}e_i({}^{I_k}\phi^i)) \right. \\ & \left. + \rho_h(e_i^\top({}_{k-1}^0g_k, {}^{I_k}\phi^i)\Sigma_p^{-1}e_i({}_{k-1}^0g_k, {}^{I_k}\phi^i)) \right\} \end{aligned} \quad (15)$$

## 2.3 Graph Optimization

将动态 SLAM 问题转化为图优化问题, 因子图的表示如 Fig.3。



**Fig. 3: Factor graph representation of an object-aware SLAM with a moving object.** Black squares stand for the camera poses at different time steps, blue for static points, red for the same dynamic point on an object (dashed box) at different time steps and green for the object pose change between time steps. For ease of visualisation, only one dynamic point is drawn here. A prior factor is shown as a black circle, odometry factors are shown as orange, point measurement factors as white and point motion factors as magenta. A smooth motion factor is shown as cyan circle.

因子图的联合优化问题融合了 4 种测量/观测：

- 3D 点测量；
- 视觉惯导测量；
- 动态物体上点的观测；
- 物体连续平滑运动的观测。

3D 点测量误差：

$$e_{i,k}({}^0X_k, {}^0m_k^i) = {}^0X_k^{-1} {}^0m_k^i - z_k^i \quad (16)$$

$z = \{z_k^i\}$  表示在所有时间步下的所有 3D 点测量，其大小为  $n_z$  且  $z_k^i \in \mathbb{R}^3$ 。3D 点测量因子为 Fig.3 中白色圆圈。

视觉惯导误差定义为：

$$e_k({}^0X_{k-1}, {}^0X_k) = ({}^0X_{k-1}^{-1} {}^0X_k)^{-1} {}^{X_{k-1}}_{k-1} T_k \quad (17)$$

其中  $T = \{{}^{X_{k-1}}_{k-1} T_k\}$  表示惯导测量集合，且有大小为  $n_o$  且  ${}^{X_{k-1}}_{k-1} T_k \in SE(3)$ 。惯导因子为 Fig.3 中的橙色圆圈。

动态物体上点的运动误差定义为：

$$e_{i,l,k}({}^0m_k^i, {}^0_{k-1}H_k^l, {}^0m_{k-1}^i) = {}^0m_k^i - {}^0_{k-1}H_k^l {}^0m_{k-1}^i \quad (18)$$

在检测到的刚体  $l$  上的所有点的运动使用 (6) 中的位姿转换  ${}^0_{k-1}H_k^l$  进行表述，在因子图中对应 Fig.3 的粉红色圆圈。这是一个三元因子，称其为刚体上点的运动模型。

引入平滑运动因子，最小化物体连续运动的变化：

$$e_{l,k}({}^0_{k-2}H_{k-1}^l, {}^0_{k-1}H_k^l) = {}^0_{k-2}H_{k-1}^l {}^{-1} {}^0_{k-1}H_k^l \quad (19)$$

物体平滑运动因子  $e_{l,k}({}^0_{k-2}H_{k-1}^l, {}^0_{k-1}H_k^l)$  用于最小化物体在连续时间步下的运动变化，其对应 Fig.3 中的青色圆圈。

取  $\theta_M = \{{}^0m_k^i\}$  为所有的 3D 点构成的点集，取  $\theta_X = \{{}^0x_k^\vee\}$  为所有的相机位姿的集合。

将  ${}^0_{k-1}H_k^l$  参数化为李代数  ${}^0_{k-1}h_k^l \in se(3)$ ：

$${}^0_{k-1}H_k^l = \exp({}^0_{k-1}h_k^l) \quad (20)$$

定义  $\theta_H = \{{}^0_{k-1}h_k^l \in \mathbb{R}^6\}$  为所有物体运动的集合。

给定  $\theta = \theta_M \cup \theta_X \cup \theta_H$  为因子图的所有节点，结合李代数，最小二乘的损失定义为：

$$\begin{aligned}
\theta^* = \operatorname{argmin}_{\theta} \Big\{ & \sum_{i,k}^{n_z} \rho_h(e_{i,k}^\top ({}^0x_k, {}^0m_k^i) \Sigma_z^{-1} e_{i,k} ({}^0x_k, {}^0m_k^i)) \\
& + \sum_k^{n_o} \rho_h(\log(e_k({}^0x_{k-1}, {}^0x_k))^\top \Sigma_o^{-1} \log(e_k({}^0x_{k-1}, {}^0x_k))) \\
& + \sum_{i,l,k}^{n_g} \rho_h(e_{i,l,k}^\top ({}^0m_k^i, {}_{k-1}^0h_k^l, {}^0m_{k-1}^i) \Sigma_g^{-1} e_{i,l,k} ({}^0m_k^i, {}_{k-1}^0h_k^l, {}^0m_{k-1}^i)) \\
& + \sum_{l,k}^{n_s} \rho_h(\log(e_{l,k}({}_{k-2}^0h_{k-1}^l, {}_{k-1}^0h_k^l))^\top \Sigma_s^{-1} \log(e_{l,k}({}_{k-2}^0h_{k-1}^l, {}_{k-1}^0h_k^l))) \Big\}
\end{aligned} \tag{21}$$

使用 L-M 算法对上述优化问题进行求解。

### 3 System

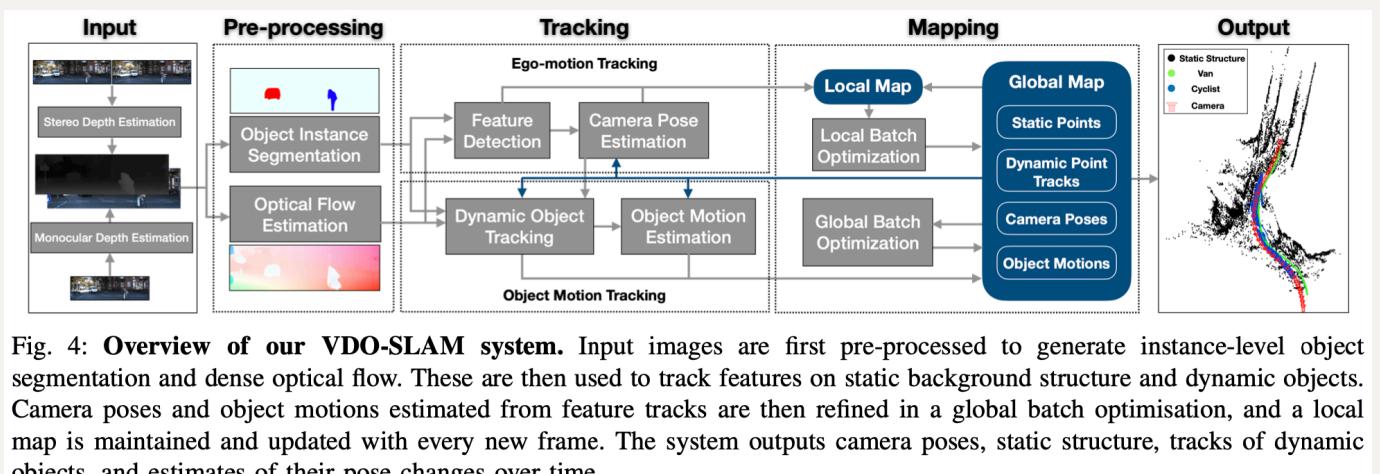


Fig. 4: Overview of our VDO-SLAM system. Input images are first pre-processed to generate instance-level object segmentation and dense optical flow. These are then used to track features on static background structure and dynamic objects. Camera poses and object motions estimated from feature tracks are then refined in a global batch optimisation, and a local map is maintained and updated with every new frame. The system outputs camera poses, static structure, tracks of dynamic objects, and estimates of their pose changes over time.

系统结构如 Fig.4，包含 3 个主要部分：image pre-processing, tracking, mapping。

系统以立体或 RGB-D 图像作为输入。对于立体图像，则在第一步，使用立体深度估计 [3] 从立体图像中提取深度信息，生成深度图作为 RGB-D 图像。

虽然该系统最初被设计为 RGB-D 系统，但为了充分利用基于图像的语义信息，使用单幅图像深度估计来获取单目相机的深度信息。因为只使用 RGB 图像作为系统的输入，但是估计问题是使用 RGB-D 数据，其中深度是使用单幅图像深度估计获得的。

## 3.1 Pre-processing

此模块存在两个关键点需要处理：

- 鲁棒地分离静态背景和动态物体；
- 确保对动态物体的长期追踪。

### 3.1.1 Object Instance Segmentation

实例级语义分割（instance-level semantic segmentation）用于分割和区分潜在的动态物体。实例分割帮助区分语义前端为不同的实例掩码，这些掩码便于跟踪每一个单独的物体。同时，分割掩码提供了物体的“精确”边界框，确保了对物体上点的鲁棒追踪。

### 3.1.2 Optical Flow Estimation

密集光流用于最大化移动物体上被追踪点的数量。大多数物体只出现在一小部分图像，使用稀疏特征匹配不能确保鲁棒性和长期特征跟踪。

本文的方法利用密集光流，通过从语义掩码内的所有点采样，大大增加目标点的数量。密集光流还用于通过传播分配给对象掩码上每个点的唯一对象标识符来一致跟踪多个对象。此外，它还允许在语义分割失败时恢复对象掩码。

## 3.2 Tracking

### 3.2.1 Feature Detection

检测一个稀疏的角特征（corner feature）点集，并光流跟踪这些点。对于每一帧，只有符合估计的相机运动的 inlier 特征点会被保存到地图，这些点被用于在下一帧中跟踪对应点（correspondences）。如果 inlier 的数量少于某一数值（默认为 1200）则会检测并加入新的特征点。这些稀疏特征在静态背景（图像上出去检测到的动态物体的区域）上提取。

### 3.2.2 Camera Pose Estimation

对于所有检测到的 3D-2D 静态点对，使用 (13) 计算相机位姿。为保证鲁棒性，使用一个运动模型生成方法进行初始化：

- 此方法使用两个模型，基于重投影误差对比它们的 inlier 数量。其中一个模型通过传播相机的先前运动得到，另一个则通过 P3P[4] 结合 RANSAC 得到。产生最多 inlier 的模型被用于初始化。

### 3.2.3 Dynamic Object Tracking

包含两个步骤：

- 分割物体并分类为静态和动态；
- 在连续帧对之间关联动态物体。

#### (1) 实例级物体分割使得能够从背景中分割出物体

尽管算法能够识别所有分割出的物体的运动，但是动态物体的区分 (idnetify) 能降低计算成本。这通过使用场景流估计完成。

具体地，在得到相机位姿  ${}^0X_k$  后，场景流向量  $f_k^i$  描述了 3D 点  ${}^0m_k^i$  在帧  $k - 1$  和  $k$  之间的运动，可以通过如下式子计算：

$$f_k^i = {}^0m_{k-1}^i - {}^0m_k^i = {}^0m_{k-1}^i - {}^0X_k X_k m_k^i \quad (22)$$

不同于光流，场景流（理想情况只由场景运动产生）可以直接决定某些结构是否正在移动。理想下所有静态点的场景流为 0，然而这些会受到深度和匹配噪声或误差的影响。

- 出于鲁棒性，在每个物体的采样点上计算场景流，如果场景流的模大于阈值（设为 0.12），则视为动态点。如果一个物体上的采样点中，动态点数量大于阈值（设为 30%），则视为动态物体。

## (2) 实例级物体分割仅提供单图像的物体标签

物体需要在不同帧之间被追踪，提出使用光流在帧间关联点标签。

- 点标签和其被采样的物体的唯一识别符相同。维护一个有限标签集  $\mathcal{L} \subset \mathbb{N}, l \in \mathcal{L}$ ，对于第一个被检测到的运动物体，从  $l = 1$  开始打标签。 $\mathcal{L}$  的大小随着被检测到的运动物体的增加而增大。静态物体和背景的标签为  $l = 0$ 。

理想下，帧  $k$  中检测到的每个物体，其所有的点标签应与  $k - 1$  帧上对应的点标签相同，然而实际中这会收到噪声等因素影响。

- 为克服上述问题，将所有点的标签设为前一帧中它们的对应点里出现最多的标签。
- 如果一个动态物体，其在前一帧中出现的最多的标签为 0，说明物体开始运动或重新出现在场景中。这种情况下，物体会被分配一个新的追踪标签。

### 3.2.4 Object Motion Estimation

物体通常出现在场景中的小部分，使得难以获取足够稀疏特征来追踪和估计运动。因此，在物体掩码内每三个点采样一次，仅有 inlier 被保留于建图和追踪。当被追踪点物体点少于阈值时，则采样并加入新的点。

## 3.3 Mapping

### 3.3.1 Local Batch Optimization

维护和更新一个局部地图。局部批优化的目的是确保精确的相机位姿估计提供给全局批优化。

局部地图的构建使用了固定大小的滑动窗口，其包含前  $n_w (= 20)$  帧的信息。只在窗口大小内局部优化相机位姿和静态结构，因为优化动态结果没有任何收益（除非有强约束，如物体的恒定运动）。然而，需要时，系统能在局部地图中合并静态和动态结构。

当局部地图被构建，则同样执行因子图优化来优化每一个局部地图中的变量，让后更新到全局地图。

### 3.3.2 Global Batch Optimization

tracking 部分和局部批优化的输出包含相机位姿、物体运动以及 inlier 的结构，这些保存在全局地图。全局地图由之前的所有帧构建并且在每一个新帧上持续更新。

每个输入帧处理结束后基于全局地图构建因子图。为高效探讨时间约束，只有被追踪超过 3 个实例的点会被加入因子图。因子图设计为如同 2.3 中的优化问题，优化结果作为整个系统的输出。

### 3.3.3 From Motion to Tracking

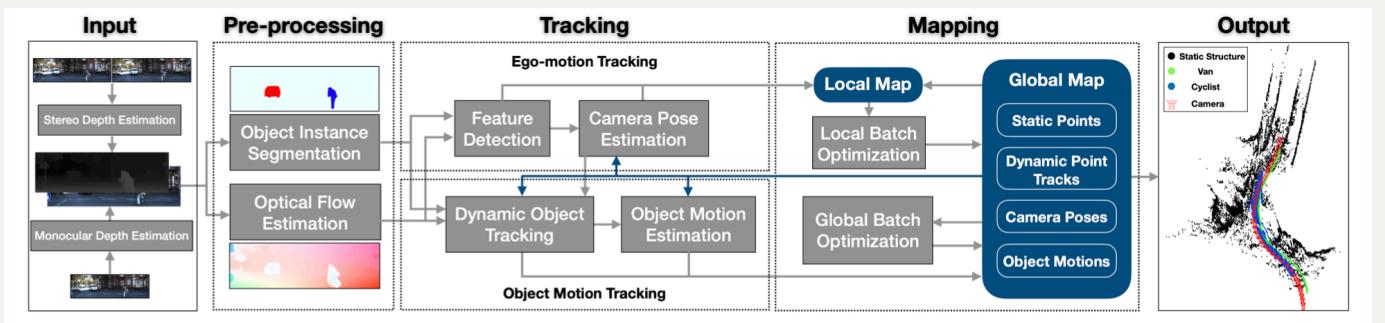


Fig. 4: Overview of our VDO-SLAM system. Input images are first pre-processed to generate instance-level object segmentation and dense optical flow. These are then used to track features on static background structure and dynamic objects. Camera poses and object motions estimated from feature tracks are then refined in a global batch optimisation, and a local map is maintained and updated with every new frame. The system outputs camera poses, static structure, tracks of dynamic objects, and estimates of their pose changes over time.

维护地图以给 tracking 模块的当前状态估计提供历史信息，如 Fig.4 中蓝色箭头。

# 4 Experiments

---

- 评估内容：相机运动、物体运动和速度、物体追踪。
- 数据集：Oxford Multimotion Dataset for indoor, KITTI Tracking dataset for outdoor。
- SOTA：MVO, ClusterVO, DynaSLAM II, CubeSLAM。

## 4.1 Deep Model Setup

- 物体分割：Mask R-CNN
- 光流：PWC-Net [5]
- 单目的深度估计：MonoDepth2 [6]（如果使用单目相机图像作为输入）
- 特征检测：FAST [7] [8]

## 4.2 Error Metrics

给定 GT 运动位姿（可以是相机或物体）变换矩阵  $T \in SE(3)$  和估计运动  $\hat{T} \in SE(3)$ ，误差定义为  $E = \hat{T}^{-1}T$ 。平移误差  $E_t$  为  $E$  中平移部分的  $L_2$  范式，旋转误差  $E_r$  为  $E$  中旋转部分的角轴旋转角度。

在 body-fixed 坐标系中的物体位姿变换由  ${}^0_{k-1}H_k$  得到：

$${}^{L_{k-1}}_{k-1}H_k = {}^0L_{k-1}^{-1}{}^0_{k-1}H_k {}^0L_{k-1} \quad (23)$$

在惯性参考系中物体上点的线速度表示为：

$$\begin{aligned} v &\approx {}^0m_k^i - {}^0m_{k-1}^i \\ &= ({}^0_{k-1}H_k - I_4) {}^0m_{k-1}^i \\ &= {}^0_{k-1}t_k - (I_3 - {}^0_{k-1}R_k) {}^0m_{k-1}^i \end{aligned} \quad (24)$$

为得到更可靠的速度测量，平均同一时刻物体上的所有点。对于  $k-1$  时物体上的所有  $n$  个点，定义  $c_{k-1} = \frac{1}{n} \sum_i {}^0m_{k-1}^i$ ，则

$$\begin{aligned}
v &\approx \frac{1}{n} \sum_{i=1}^n \left( {}_0 \underset{k-1}{t_k} - \left( I_3 - {}_0 \underset{k-1}{R_k} \right)^0 m_{k-1}^i \right) \\
&= {}_0 \underset{k-1}{t_k} - \left( I_3 - {}_0 \underset{k-1}{R_k} \right) c_{k-1}
\end{aligned} \tag{25}$$

则速度估计值  $\hat{v}$  和 GT 值  $v$  之间的误差为  $E_s = |\hat{v}| - |v|$ 。

### 4.3 Oxford Multimotion Dataset

TABLE I: Comparison versus MVO [51] and ClusterVO [52] for camera pose and object motion estimation accuracy on the sequence of swinging\_4\_unconstrained sequence in Oxford Multi-motion dataset. Bold numbers indicate the better results.

	VDO-SLAM		MVO		ClusterVO	
	$E_r$ (deg)	$E_t$ (m)	$E_r$ (deg)	$E_t$ (m)	$E_r$ (deg)	$E_t$ (m)
Camera	0.7709	0.0112	1.1948	0.0314	<b>0.7665</b>	<b>0.0066</b>
Top-left Swinging Box	<b>1.1889</b>	<b>0.0207</b>	1.4553	0.0288	3.2537	0.0673
Top-right Swinging and rotating Box	<b>0.7631</b>	0.0132	0.8992	<b>0.0130</b>	3.5308	0.0256
Bottom-left Swinging Box	<b>0.9153</b>	<b>0.0149</b>	1.4949	0.0261	4.9146	0.0763
Bottom-right Rotating Box	0.8469	0.0192	<b>0.7815</b>	<b>0.0115</b>	4.0675	0.0144

轨迹跟踪输出如 Fig.5，摆动箱子上的动态特征轨迹在视觉上与箱子的实际运动相对应。

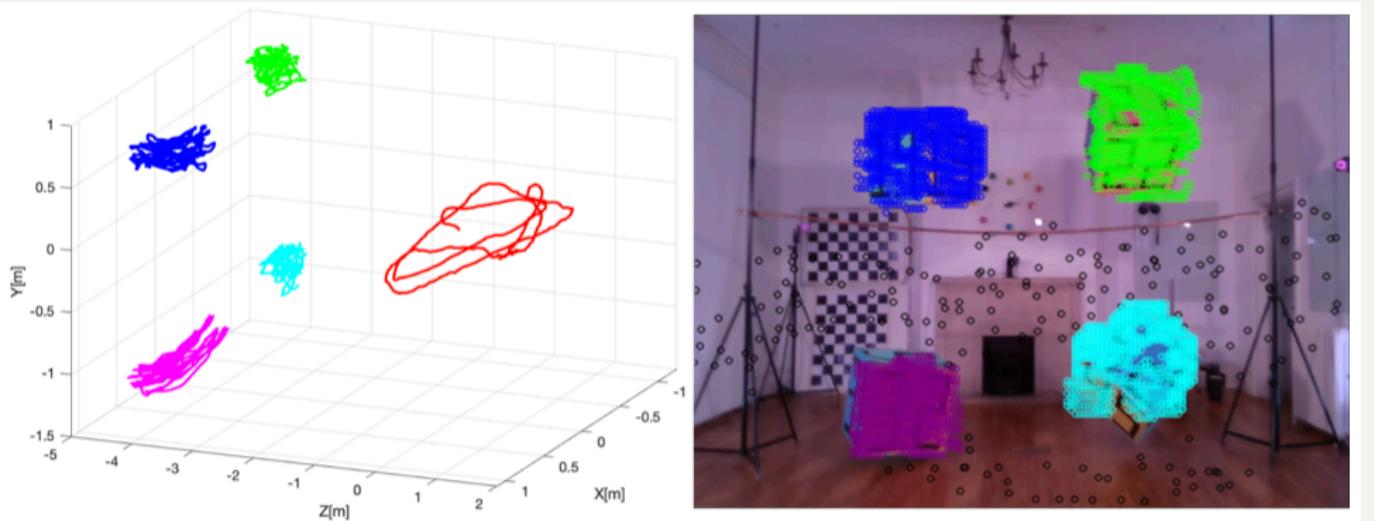


Fig. 5: Qualitative results of our method on Oxford Multimotion Dataset. (Left) The 3D trajectories of camera (red) and centres of the four boxes. (Right) Detected points on static background and object body. Black color corresponds to static points and features on each object are shown in a different color.

## 4.4 KITTI Tracking Dataset

### 4.4.1 Camera Pose and Object Motion

TABLE II: Comparison versus DynaSLAM II [49] and CubeSLAM [24] for camera pose and object motion estimation accuracy on nine sequences with moving objects drawn from the KITTI dataset. Bold numbers indicate the better result.

Seq	DynaSLAM II				VDO-SLAM (RGB-D)				VDO-SLAM (Monocular)				CubeSLAM			
	Camera		Object		Camera		Object		Camera		Object		Camera		Object	
	$E_r$ (deg)	$E_t$ (m)	$E_r$ (deg)	$E_t$ (m)	$E_r$ (deg)	$E_t$ (m)	$E_r$ (deg)	$E_t$ (m)	$E_r$ (deg)	$E_t$ (m)	$E_r$ (deg)	$E_t$ (m)	$E_r$ (deg)	$E_t$ (m)	$E_r$ (deg)	$E_t$ (m)
00	<b>0.06</b>	<b>0.04</b>	0.0741	0.0674	<b>1.0520</b>	<b>0.1077</b>	0.1830	0.1847	2.0021	0.3827	-	-	-	-	-	-
01	0.04	<b>0.05</b>	<b>0.0382</b>	0.1220	<b>0.9051</b>	<b>0.1573</b>	0.1772	0.4982	1.1833	0.3589	-	-	-	-	-	-
02	0.02	<b>0.04</b>	<b>0.0182</b>	0.0445	<b>1.2359</b>	<b>0.2801</b>	0.0496	0.0963	1.6833	0.4121	-	-	-	-	-	-
03	0.04	<b>0.06</b>	<b>0.0311</b>	0.0816	<b>0.2919</b>	<b>0.0965</b>	0.1065	0.1505	0.4570	0.2032	0.0498	0.0929	3.6085	4.5947	-	-
04	0.06	<b>0.07</b>	<b>0.0482</b>	0.1114	<b>0.8288</b>	<b>0.1937</b>	0.1741	0.4951	3.1156	0.5310	0.0708	0.1159	5.5803	32.5379	-	-
05	0.03	<b>0.06</b>	<b>0.0219</b>	0.0932	<b>0.3705</b>	<b>0.1140</b>	0.0506	0.1368	0.6464	0.2669	0.0342	0.0696	3.2610	6.4851	-	-
06	<b>0.04</b>	0.02	0.0488	<b>0.0186</b>	<b>1.0803</b>	<b>0.1158</b>	0.0671	0.0451	2.0977	0.2394	-	-	-	-	-	-
18	<b>0.02</b>	<b>0.05</b>	0.0211	0.0749	<b>0.2453</b>	<b>0.0825</b>	0.1236	0.3551	0.5559	0.2774	0.0433	0.0510	3.1876	3.7948	-	-
20	0.04	<b>0.07</b>	<b>0.0271</b>	0.1662	<b>0.3663</b>	<b>0.0824</b>	0.3029	1.3821	1.1081	0.3693	0.1348	0.1888	3.4206	5.6986	-	-

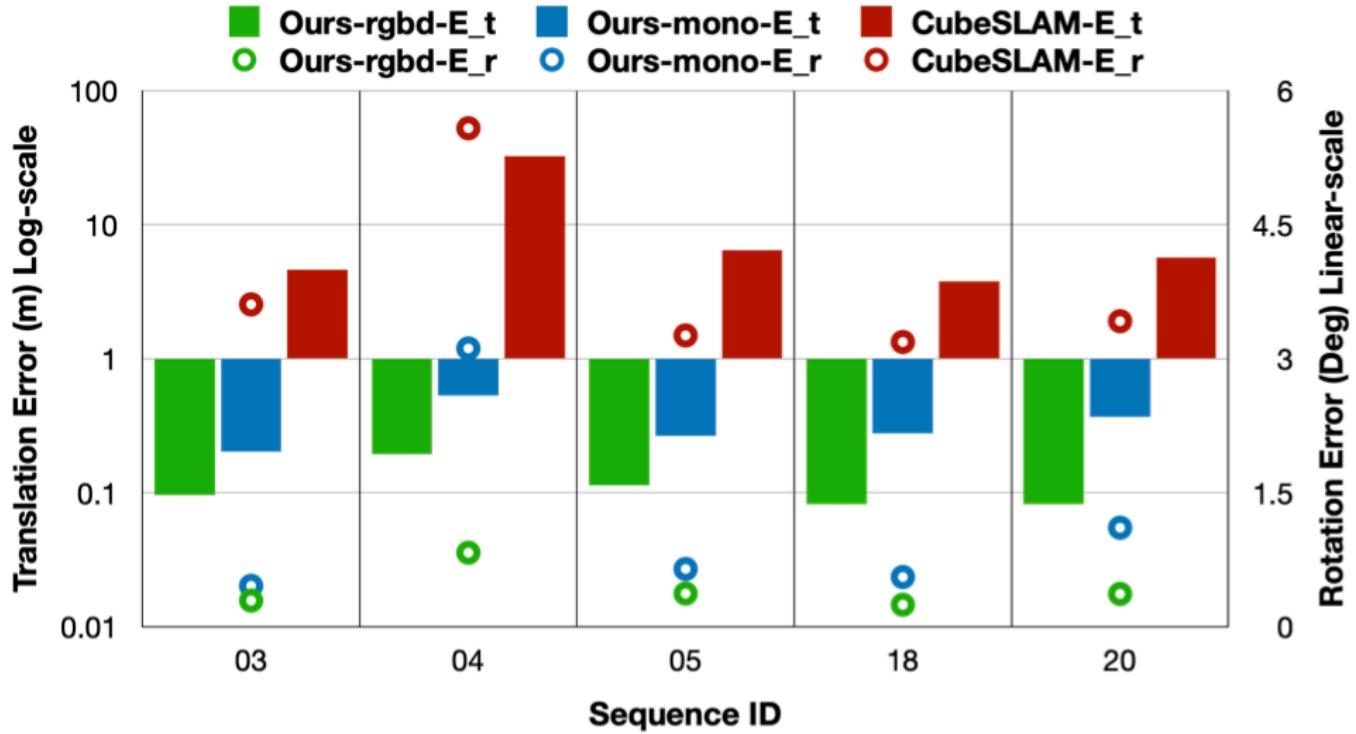
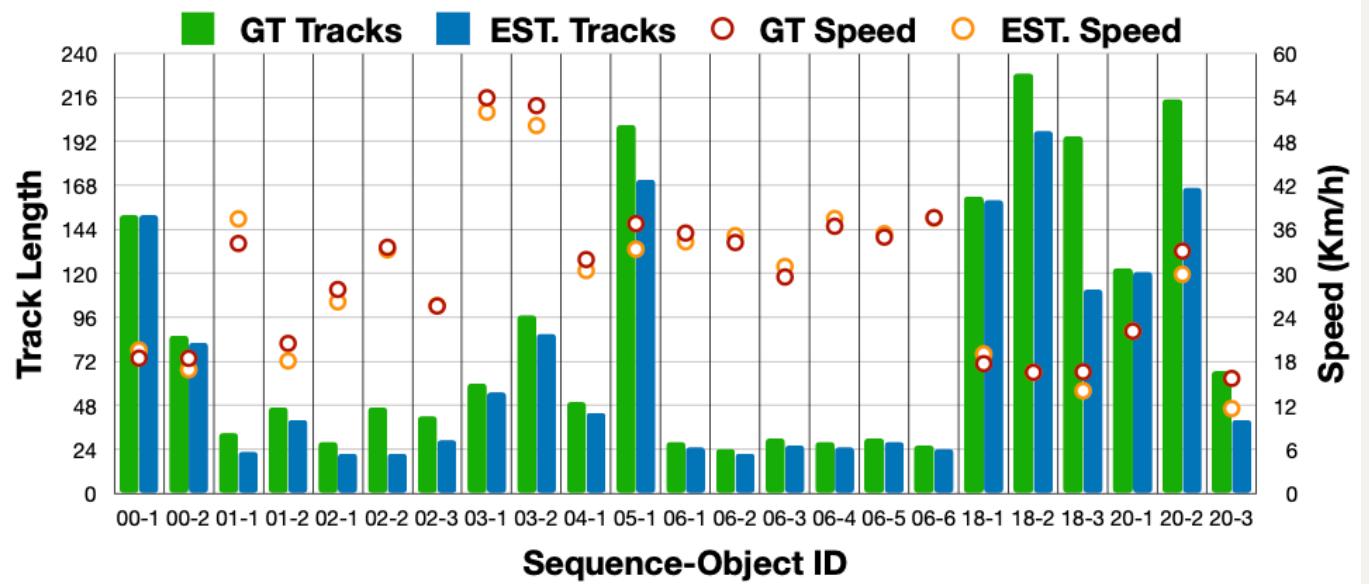


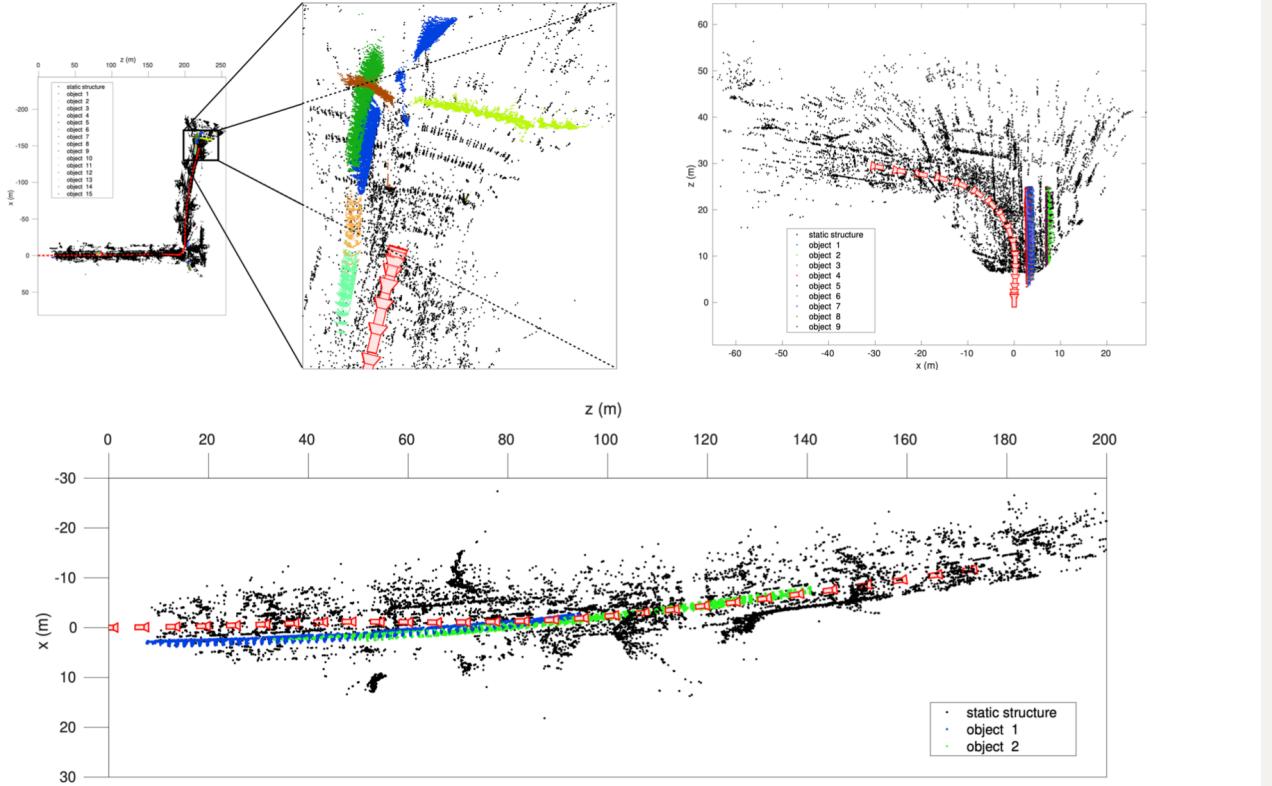
Fig. 6: **Accuracy of object motion estimation of our method compared to CubeSLAM ([24]).** The color bars refer to translation error that is corresponding to the left Y-axis in log-scale. The circles refer to rotation error, which corresponds to the right Y-axis in linear-scale.

#### 4.4.2 Object Tracking and Velocity



**Fig. 7: Tracking performance and speed estimation.** Results of object tracking length and object speed for some selected objects (tracked for over 20 frames), due to limited space. The color bars represent the length of object tracks, which is corresponding to the left Y-axis. The circles represent object speeds, which is corresponding to the right Y-axis. GT refers to ground truth, and EST. refers to estimated values.

#### 4.4.3 Qualitative Results



**Fig. 8: Illustration of system output; a dynamic map with camera poses, static background structure, and tracks of dynamic objects.** Sample results of VDO-SLAM on KITTI sequences. Black represents static background, and each detected object is shown in a different colour. Top left figure represents Seq.01 and a zoom-in on the intersection at the end of the sequence, top right figure represents Seq.06 and bottom figure represents Seq.03.

## 4.5 Discussion

### 4.5.1 Robust Tracking of Points

对比式子(9)(10)与(13)(15)产生的特征点数量:

TABLE III: The number of points tracked for more than five frames on the nine sequences of the KITTI dataset. Bold numbers indicate the better results. Underlined bold numbers indicate an order of magnitude increase in number.

Seq	Background		Object	
	Motion Only	Joint	Motion Only	Joint
00	1798	<b>12812</b>	1704	<b>7162</b>
01	237	<b>5075</b>	907	<b>4583</b>
02	7642	<b>10683</b>	52	<b>1442</b>
03	778	<b>12317</b>	343	<b>3354</b>
04	9913	<b>25861</b>	339	<b>2802</b>
05	713	<b>11627</b>	2363	<b>2977</b>
06	7898	<b>11048</b>	482	<b>5934</b>
18	4271	<b>22503</b>	5614	<b>14989</b>
20	9838	<b>49261</b>	9282	<b>13434</b>

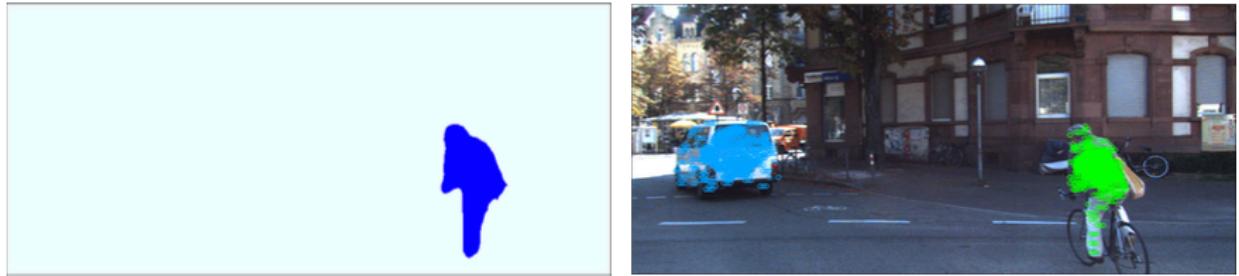
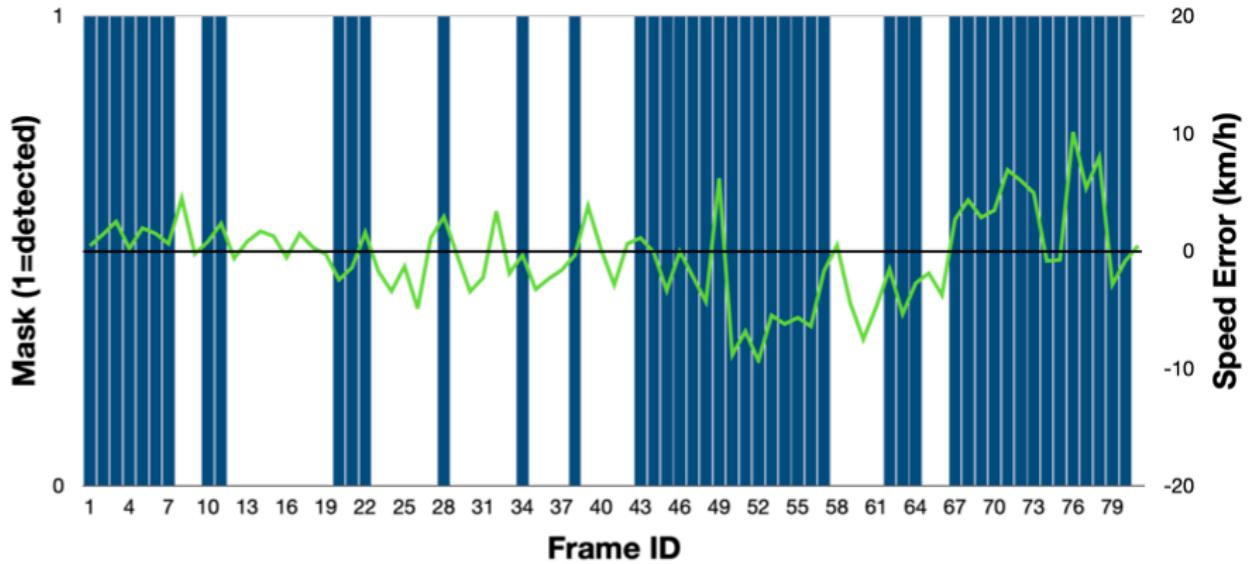
对应的实验误差：

TABLE IV: Average camera pose and object motion errors over the nine sequences of the KITTI dataset. Bold numbers indicate the better results.

	Motion Only		Joint	
	$E_r$ (deg)	$E_t$ (m)	$E_r$ (deg)	$E_t$ (m)
Camera	0.0412	0.0987	<b>0.0365</b>	<b>0.0866</b>
Object	1.0179	0.1853	<b>0.7085</b>	<b>0.1367</b>

#### 4.5.2 Robustness against Non-direct Occlusion

Fig.9 中为跟踪白色厢型车 80 帧，掩码分割在有 33 失败，系统仍然能对其进行跟踪。后半部分速度误差较大是因为出现部分遮挡以及物体远离相机。

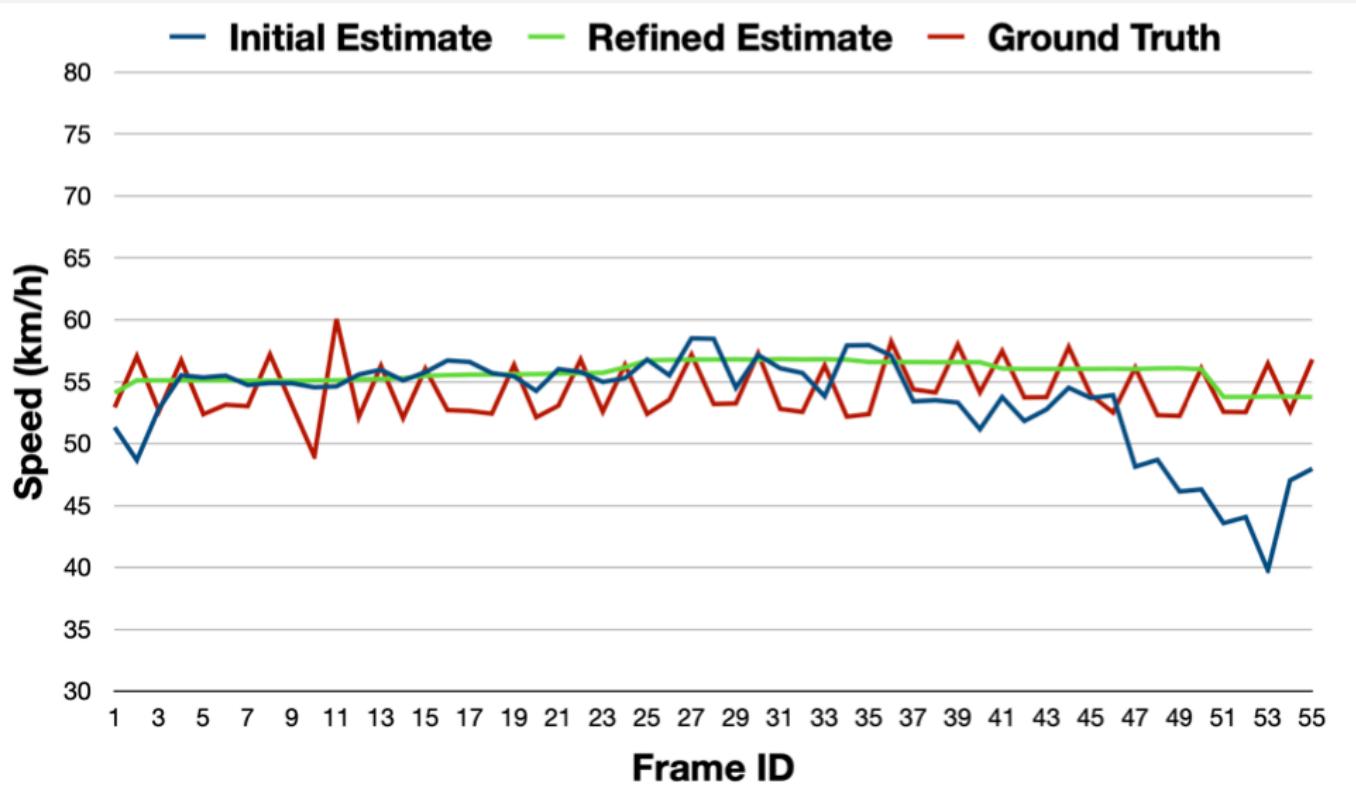


**Fig. 9: Robustness in tracking performance and speed estimation in case of semantic segmentation failure.**

An example of tracking performance and speed estimation for a white van (ground-truth average speed 20km/h) in Seq.00. (Top) Blue bars represent a successful object segmentation, and green curves refer to the object speed error. (Bottom-left) An illustration of semantic segmentation failure on the van. (Bottom-right) Result of propagating the previously tracked features on the van by our system.

### 4.5.3 Global Refinement on Object Motion

对比强化前后的结果，使用 2.3 中的优化方法：



**Fig. 10: Global refinement effect on object speed estimation.** The initial (blue) and refined (green) estimated speeds of a wagon in Seq.03, travelling along a straight road, compared to the ground truth speed (red). Note the ground truth speed is slightly fluctuating. We believe it is due to the ground truth object poses being approximated from lidar scans.



**Fig. 11: Improvement on object motion after graph optimization.** The numbers in the heatmap show the ratio of decrease in error on the nine sequences of the KITTI dataset.

#### 4.5.4 Computational Analysis

使用修改版本的 g2o[9] 作为后端：

**TABLE V:** Runtime of different system components for both datasets. The time cost of every component is averaged over all frames and sequences, except for the object motion estimation and object motion estimation that are averaged over the number of objects.

Dataset	Tasks	Runtime (mSec)
KITTI	Feature Detection	16.2550
	Camera Pose Estimation	52.6542
	Dynamic Object Tracking (avg/object)	8.2980
	Object Motion Estimation (avg/object)	22.9081
	Map and Mask Updating	22.1830
	Local Batch Optimisation	18.2828
OMD	Feature Detection	7.5220
	Camera Pose Estimation	32.0909
	Dynamic Object Tracking (avg/object)	7.0134
	Object Motion Estimation (avg/object)	19.5280
	Map and Mask Updating	30.3153
	Local Batch Optimisation	15.3414

## 参考

- [1] M. Henein, J. Zhang, R. Mahony, and V. Ila, “Dynamic SLAM: The Need for Speed,” in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2123–2129.

- [2] J. Zhang, M. Henein, R. Mahony, and V. Ila, “Robust Ego and Object 6-DoF Motion Estimation and Tracking,” in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5017–5023.
- [3] K. Yamaguchi, D. McAllester, and R. Urtasun, “Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation,” in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 756–771.
- [4] T. Ke and S. I. Roumeliotis, “An Efficient Algebraic Solution to the Perspective-three-point Problem,” in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.
- [5] D. Sun, X. Yang, M. -Y. Liu, and J. Kautz, “PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume,” in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.
- [6] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, “Digging into Self-supervised Monocular Depth Estimation,” in *International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 3828–3838.
- [7] E. Rosten and T. Drummond, “Machine Learning for High-speed Corner Detection,” in *European Conference on Computer Vision (ECCV)*. Springer, 2006, pp. 430–443.
- [8] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An Efficient Alternative to SIFT or SURF,” in *International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2564–2571.
- [9] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “g2o: A General Framework for Graph Optimization,” in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3607–3613.