From Scene Flow to Visual Odometry through Local and Global Regularisation in Markov Random Fields

Raluca Scona, Hidenobu Matsuki and Andrew Davison

***Abstract*—We revisit pairwise Markov Random Field (MRF) formulations for RGB-D scene flow and leverage novel advances in processor design for real-time implementations. We consider scene flow approaches which consist of data terms enforcing intensity consistency between consecutive images, together with regularisation terms which impose smoothness over the flow field. To achieve real-time operation, previous systems leveraged GPUs and implemented regularisation only between variables corresponding to neighbouring pixels. Such systems could estimate continuously deforming flow fields but the lack of global regularisation over the whole field made them ineffective for visual odometry. We leverage the GraphCore Intelligence Processing Unit (IPU) graph processor chip, which consists of 1216 independent cores called tiles, each with 256kB local memory. The tiles are connected to an ultrafast all-to-all communication fabric which enables efficient data transmission between the tiles in an arbitrary communication pattern. We propose a distributed formulation for dense RGB-D scene flow based on Gaussian Belief Propagation which leverages the architecture of this processor to implement both local and global regularisation. Local regularisation is enforced for pairs of flow estimates whose corresponding pixels are neighbours, while global regularisation is defined for flow estimate pairs whose corresponding pixels are far from each other on the image plane. Using both types of regularisation allows our algorithm to handle a variety of in-scene motion and makes it suitable for estimating deforming scene flow, piece-wise rigid scene flow and visual odometry within the same system.**

***Index Terms*—RGB-D Perception; Visual Tracking; Hardware-Software Integration in Robotics**

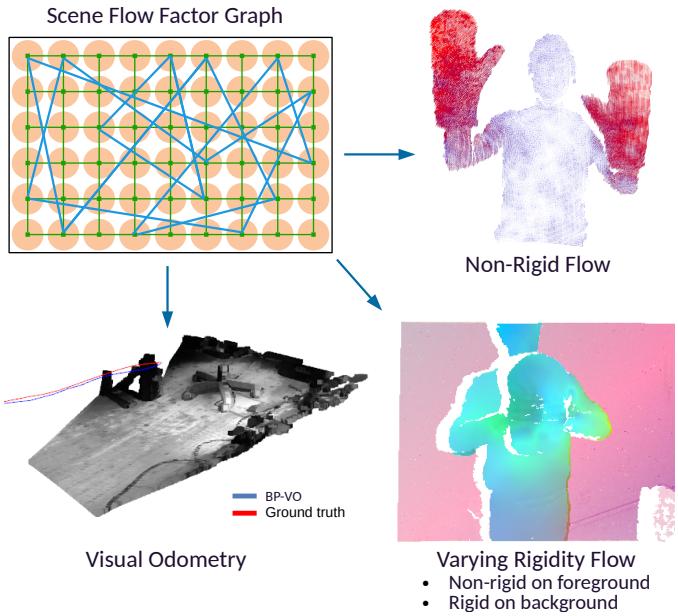

Fig. 1: **Top left:** Scene flow factor graph, with per-pixel flow variables (orange), local regularisation (green) and global regularisation (blue). **Top right:** Non-rigid scene flow, with flow vectors coloured by magnitude (large in red, small in blue). **Bottom left:** Visual odometry, showing our estimated trajectory (blue), ground truth trajectory (red) and 100 reprojected RGB-D frames from *fr1/floor* [1]. Visual odometry is sampled from the rigid flow field. **Bottom right:** Varying rigidity scene flow: continuously deforming on the foreground and nearly constant on the background, where a segmentation mask of the foreground is assumed to be provided.

I. Introduction

SCENE flow estimates the 3D non-rigid motion field of a scene between consecutive images. Its potential uses include helping robots navigate while avoiding moving obstacles, enabling SLAM (simultaneous localisation and mapping) in dynamic environments or as input to motion segmentation.

Scene flow received significant research interest thanks to commodity 3D cameras and improvements in deep learning (DL). While DL-based methods demonstrate compelling results, variational formulations remain relevant. They readily generalise and are low cost to develop, not requiring training data or large computational resources for training.

However, variational formulations for scene flow are ill-posed and require regularisation to be solved. Although scene flow measures the motion of every 3D point, it cannot directly be used as visual odometry even in static scenes. This lead to different types of motion estimation algorithms to support different kinds of regularisation, from fully non-rigid scene flow to piece-wise rigidity and visual odometry.

We propose a pairwise Markov Random Field (MRF)-type method to bridge the gap between non-rigid scene flow and rigid visual odometry by leveraging novel advances in processor design. While GPUs have traditionally been used to solve scene flow, real-time variational approaches mainly implemented local regularisation between neighboring pixels by

Manuscript received: September, 9, 2021; Revised December, 7, 2021; Accepted January, 25, 2022.

This paper was recommended for publication by Editor Cesar Cadena upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by Dyson Technology Ltd. and EPSRC under a prosperity partnership award.

H. Matsuki and A. Davison are with the Dyson Robotics Laboratory at Imperial College London. R. Scona is currently with the Advanced Technology Division of Ocado Technology, Hatfield, UK. This research was done when R. Scona was with the Dyson Robotics Laboratory at Imperial College London. raluca.scona@gmail.com

Digital Object Identifier (DOI): see top of this page.

leveraging fast shared memory access [2]. Recently, unconventional architectures have been developed for computer vision applications, and an important example is the GraphCore IPU [3]. With its distributed per-tile memory and ultrafast all-to-all communication fabric, the IPU can support greater flexibility in the communication patterns of algorithms and we leverage this to implement local as well as global regularisation.

We formulate scene flow as a factor graph where the variables are per-pixel 6D motion estimates and with two types of factors: 1. unary photometric factors which warp a pixel between consecutive frames to minimise the photometric error and 2. pairwise smoothness factors which penalise the difference between specified pairs of motion estimates.
To support both local and global regularisation, we implement two kinds of smoothing factors:

- Local smoothing between neighbouring variables under the assumption of local rigidity.
- Global smoothing between pairs of variables that are located far from each other on the image plane, to synchronise the motion estimate across the image and simulate visual odometry.

Regarding global regularisation, we observe that sampling N (4 to 8) smoothing connections uniformly at random on the image plane for each pixel causes the scene flow distribution to become very tight and consistent with the camera motion. We interpret this as a 'small world' property, where flow vectors receive information from different image regions, converging to a globally consistent solution.

We propose that a combination of these smoothing factors enables our algorithm to tackle a continuous range of problems from pure scene flow to pure rigid visual odometry (Fig. 1). To summarise, our contributions are:

1) A factor graph formulation for RGB-D scene flow, solved using the Gaussian Belief Propagation message passing algorithm which runs in real-time on the IPU.
2) We implement both local and global regularisation and show that depending on how this is set, the behaviour of the algorithm changes significantly, such that the same method is used to estimate deforming scene flow, piecewise rigid scene flow and visual odometry.
3) To the best of our knowledge, ours is the first approach for decentralised visual odometry, where the global motion of the camera is computed through purely local computations using the global smoothing factors.

II. Related Works

a) Local regularisation: Scene flow is the 3D extension of the optical flow problem and was pioneered for multi-view stereo systems through formulations solving optical and range flow constraints [4], [5]. Following works explored variational formulations and methods for regularising the flow field, where notably TV (Total Variation) worked well in enforcing discontinuities between moving objects [6], [7]. Herbst *et al*. [8] proposed one of the first RGB-D scene flow techniques combining optical, range flow and TV regularisation weighted by depth, surface normals and colour. Jaimez *et al*. [2] proposed the first real-time RGB-D scene flow system with a primal-dual formulation. The solver was designed with pixel-wise computation in mind which made it suitable for the GPU.

Although belief propagation has been applied to optical flow [9], [10], the computational cost of this algorithm for standard processors was significant for real-time applications and we are not aware of further studies applied to scene flow.

b) Piece-wise rigidity: Further research has explored different types of regularisation over the flow field. Some techniques extract superpixels assuming they correspond to rigid planar regions [11], [12] Sun *et al*. [13] focus on robust occlusion handling by decomposing the scene into moving layers ordered by depth. In these methods, motion estimation can be coupled with a labeling task, where each pixel is assigned an estimated motion and a patch or layer.

Jaimez *et al*. [14] decompose depth images into clusters and estimate whether each cluster is static or dynamic. Dynamic clusters are assigned independent rigid motions while static clusters are used for visual odometry. Quiroga *et al*. [15] represent flow estimates using 6D twists and simultaneously estimate the camera motion and scene flow. They formulate separate cost functions for both quantities which are solved iteratively and in alternation. The difference between our method and these contributions is that we use the same formulation for both scene flow and visual odometry and also support deforming objects whose motion cannot be well captured through rigid decompositions.

c) Learned Features: GPUs have enabled many DL techniques for scene flow. These can be more robust to large motions by relying on complex features extracted through deep convolutional layers [16]. DL methods have also been explored for point clouds, such as FlowNet3D [17] which extracts deep features from point clouds to estimate per-point 3D motion vectors. Tishchenko *et al*. [18] first learn a rigid transformation between a pair of point clouds followed by non-rigid flow after registering the point clouds.

While not directly related, techniques like DynamicFusion [19] and VolumeDeform [20] focus on reconstructing a single deforming foreground object which is tracked using sparser deformation graphs. A dense flow field is computed from the deformation graphs through motion interpolation. Novel directions include learning dense non-rigid correspondences [21] and improving real-time efficiency [22].

III. Scene Flow Factor Graph

The variables in the factor graph are the per-pixel scene flow estimates $V = \{\boldsymbol{v}_i\}_{i\in\Omega}$, where Ω is the image plane. We formulate: 1. unary measurement factors m which compute the photometric error of a pixel between consecutive images given the estimated flow and 2. pairwise smoothing factors s defined over a set of pairs N, which penalise the difference between these motion estimates. Smoothing factors are implemented between neighboring variables and also variables that are far from each other on the image plane. The structure of this factor graph is shown in Fig. 2.

We maximise the posterior of the flow estimates given the measurements and smoothness constraints:

$$V^* = \arg\max_V \prod_{\boldsymbol{v}_i\in V} m_i(\boldsymbol{v}_i) \prod_{(j,k)\in N} s_{j,k}(\boldsymbol{v}_j, \boldsymbol{v}_k). \quad (1)$$

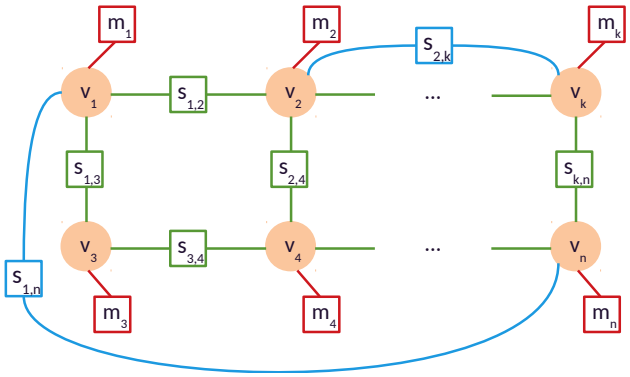

Fig. 2: The structure of the scene flow factor graph. Per-pixel variables $\boldsymbol{v}_i$ are each associated a measurement factor m_i. Smoothing factors $s_{i,j}$ are implemented between neighbouring variables (green) as well as variables that are far from each other (blue).

We formulate each factor as a least-squares error term with zero mean Gaussian noise. Next, we describe the state parametrisation and the factors.

A. State Parametrisation

Many RGB-D scene flow methods represent the motion of a point using a 3D vector which is the translation of this point between consecutive frames. This parametrisation works well in scenarios where local smoothing is enforced and where rotation estimation is not required [2]. However, our system tackles both scene flow and visual odometry. This requires the estimation of full 3D motion, so we represent each flow value as a column vector $\boldsymbol{v} = [\boldsymbol{t}, \boldsymbol{\omega}] \in \mathbb{R}^6$ composed of a stacked translation vector $\boldsymbol{t} \in \mathbb{R}^3$ and a rotation twist $\boldsymbol{\omega} \in \mathbb{R}^3$. A point $\boldsymbol{p} \in \mathbb{R}^3$ is warped from frame i to frame $i+1$ as follows:

$$\boldsymbol{p}_{i+1} = \exp_M(\boldsymbol{\omega}^\times)\boldsymbol{p}_i + \boldsymbol{t}, \tag{2}$$

where $\boldsymbol{\omega}^\times$ is the 3×3 skew-symmetric matrix representation of $\boldsymbol{\omega}$ and $\exp_M : \boldsymbol{so}(3) \rightarrow \mathbb{SO}(3)$ maps an element of Lie Algebra to a rotation matrix.

B. Gaussian Photometric Factor

This factor computes the likelihood of a scene flow estimate given the photometric error of its corresponding pixel as it is warped between the current and previous intensity images (I_1 and I_0 respectively). For every pixel i with 2D coordinates $\boldsymbol{x}_i \in \Omega$ and corresponding flow estimate $\boldsymbol{v}_i = [\boldsymbol{t}_i, \boldsymbol{\omega}_i]^T$, we formulate a Gaussian photometric factor:

$$m_i(\boldsymbol{v}_i) \propto \exp\left(-\frac{1}{2}||z_{m_i} - h_{m_i}(\boldsymbol{v}_i)||^2_{\Sigma_m}\right), \tag{3}$$

where $z_{m_i} = I_0[\boldsymbol{x}_i]$ and with the measurement function:

$$h_{m_i}(\boldsymbol{v}_i) = I_1\left[\pi\left(\exp_M(\boldsymbol{\omega}_i)\pi^{-1}\left(\boldsymbol{x}_i, D_0\left[\boldsymbol{x}_i\right]\right) + \boldsymbol{t}_i\right)\right], \tag{4}$$

where D_0 is the depth image corresponding to I_0 and $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ projects a 3D point onto a 2D plane using the pinhole camera model. We minimise this photometric error (Eq. (4)) to compute flow fields which represent the scene motion between consecutive images.

The measurement function h_{m_i} is non-linear due to the rotational component of $\boldsymbol{v}_i$ and in Section IV-A2 we describe how this factor is relinearised during optimisation to be expressed as a Gaussian distribution.

C. Gaussian Smoothness Factor

Gaussian smoothing factors constrain that pairs of flow vectors should have a similar value:

$$s_{j,k}(\boldsymbol{v}_j, \boldsymbol{v}_k) \propto \exp\left(-\frac{1}{2}\left|\left|\begin{bmatrix}\boldsymbol{t}_k\\ \boldsymbol{\omega}_k\end{bmatrix} - \begin{bmatrix}\boldsymbol{t}_j\\ \boldsymbol{\omega}_j\end{bmatrix}\right|\right|^2_{\Sigma_s}\right). \tag{5}$$

These factors reduce the influence of outlier flow estimates and induce smoothness in the flow field.

D. MAP Estimation

Given the Gaussian forms of the factors specified above, we reformulate the problem as least-squares minimisation:

$$V^* = \arg\min_V \left[\sum_{\boldsymbol{v}_i \in V} ||z_{m_i} - h_{m_i}(\boldsymbol{v}_i)||^2_{\Sigma_m} + \sum_{(j,k)\in N}\left|\left|\begin{bmatrix}\boldsymbol{t}_k\\ \boldsymbol{\omega}_k\end{bmatrix} - \begin{bmatrix}\boldsymbol{t}_j\\ \boldsymbol{\omega}_j\end{bmatrix}\right|\right|^2_{\Sigma_s}\right]. \tag{6}$$

Where Σ_m and Σ_s are the covariance matrices of the photometric and smoothing factors respectively.

IV. Gaussian Belief Propagation

We optimise the factor graph formulation in Eq. 6 using Gaussian Belief Propagation, a message-passing-based solver which can be efficiently implemented on the IPU [23] [24].

A. Message Passing

We represent Gaussian distributions in information form [25], using information vector $\boldsymbol{\eta}$ and precision matrix $\boldsymbol{\Lambda}$:

$$\mathcal{N}(\boldsymbol{v}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto \mathcal{N}^{-1}(\boldsymbol{v}, \boldsymbol{\eta}, \boldsymbol{\Lambda}), \text{ with} \tag{7}$$

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}\ ;\ \boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}. \tag{8}$$

This aids implementation as multiplication of distributions is equivalent to adding information vectors and matrices. Next, we describe variable and factor message computations [26].

1) Variable to Factor Message: A variable belief at time t is represented in Gaussian information form: ${}^t b_i(\boldsymbol{v}_i) = \mathcal{N}^{-1}(\boldsymbol{v}_i; {}^t\boldsymbol{\eta}_{b_i}, {}^t\boldsymbol{\Lambda}_{b_i})$ and is computed by taking the product of incoming messages from the photometric and smoothing factors, which is equivalent to summing the information vectors and precision matrices respectively:

$$\begin{aligned} {}^{t+1}\boldsymbol{\eta}_{b_i} &= {}^t\boldsymbol{\eta}_{m_i \rightarrow i} + \sum_{(i,j)\in N} {}^t\boldsymbol{\eta}_{s_{i,j}\rightarrow i} \\ {}^{t+1}\boldsymbol{\Lambda}_{b_i} &= {}^t\boldsymbol{\Lambda}_{m_i \rightarrow i} + \sum_{(i,j)\in N} {}^t\boldsymbol{\Lambda}_{s_{i,j}\rightarrow i} \end{aligned} \tag{9}$$

Variables send messages only to smoothing factors, as photometric factors are unary. A message from a variable i to a factor $s_{i,j}$ is computed by multiplying incoming messages from all factors except for the message from factor $s_{i,j}$:

$$\begin{aligned} {}^t\boldsymbol{\eta}_{i\rightarrow s_{i,j}} &= {}^{t+1}\boldsymbol{\eta}_{b_i} - {}^t\boldsymbol{\eta}_{s_{i,j}\rightarrow i}, \\ {}^t\boldsymbol{\Lambda}_{i\rightarrow s_{i,j}} &= {}^{t+1}\boldsymbol{\Lambda}_{b_i} - {}^t\boldsymbol{\Lambda}_{s_{i,j}\rightarrow i}. \end{aligned} \tag{10}$$

2) Factor to Variable Message:

a) Photometric Factor: To compute the information form of the photometric factor (Sec. III-B), the measurement function h_{m_i} must first be linearised around a state variable $\boldsymbol{v}_{i,0}$. The information vector and precision matrix of a linearised factor take the form [23], [27]:

$$\begin{aligned} {}^{t}\boldsymbol{\eta}_{m_i \to i} &= \boldsymbol{J}^T \Sigma_m^{-1} \left(\boldsymbol{J}\boldsymbol{v}_{i,0} + z_{m_i} - h_{m_i}(\boldsymbol{v}_{i,0}) \right), \\ {}^{t}\boldsymbol{\Lambda}_{m_i \to i} &= \boldsymbol{J}^T \Sigma_m^{-1} \boldsymbol{J}, \end{aligned} \tag{11}$$

with the 1×6 Jacobian $\boldsymbol{J} = \left[\frac{\partial h_{m_i}}{\partial \boldsymbol{t}_i}, \frac{\partial h_{m_i}}{\partial \boldsymbol{\omega}_i} \right] \Big|_{\boldsymbol{t}_i = \boldsymbol{t}_{i,0}, \boldsymbol{\omega}_i = \boldsymbol{\omega}_{i,0}}$.

The measurement nodes are leaf nodes and at every time step t they are relinearised around the current state estimate and send the message: $\mathcal{N}^{-1}(\boldsymbol{v}_i; {}^{t}\boldsymbol{\eta}_{m_i \to i}, {}^{t}\boldsymbol{\Lambda}_{m_i \to i})$.

b) Smoothness Factor: This is a linear factor and computing its information vector and precision matrix following Eq. 11 needs only to be done once. The structure of the factor, partitioned according to state, is as follows:

$$s_{j,k}(\boldsymbol{v}_j, \boldsymbol{v}_k) = \mathcal{N}^{-1} \left(\begin{bmatrix} \boldsymbol{v}_j \\ \boldsymbol{v}_k \end{bmatrix}; \begin{bmatrix} \boldsymbol{\eta}_j \\ \boldsymbol{\eta}_k \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Lambda}_{jj} & \boldsymbol{\Lambda}_{jk} \\ \boldsymbol{\Lambda}_{kj} & \boldsymbol{\Lambda}_{kk} \end{bmatrix} \right). \tag{12}$$

For multi-variate factors, computing the message for a variable requires first multiplying the factor distribution with all other incoming variable messages and then marginalising out these other variables. For example, to compute the message from factor $s_{j,k}$ to variable j, we first add the incoming message from variable k:

$$\boldsymbol{\eta}_{s_{j,k}} = \begin{bmatrix} \boldsymbol{\eta}_j \\ \boldsymbol{\eta}_k + {}^{t}\boldsymbol{\eta}_{k \to s_{j,k}} \end{bmatrix}; \boldsymbol{\Lambda}_{s_{j,k}} = \begin{bmatrix} \boldsymbol{\Lambda}_{jj} & \boldsymbol{\Lambda}_{jk} \\ \boldsymbol{\Lambda}_{kj} & \boldsymbol{\Lambda}_{kk} + {}^{t}\boldsymbol{\Lambda}_{k \to s_{j,k}} \end{bmatrix}. \tag{13}$$

We apply the Schur complement as proposed in [25] to the upper part of the system to marginalise out variables dependent on k and obtain a message for variable j:

$$\begin{aligned} {}^{t+1}\boldsymbol{\eta}_{s_{j,k} \to j} &= \boldsymbol{\eta}_j - \boldsymbol{\Lambda}_{jk} \left(\boldsymbol{\Lambda}_{kk} + {}^{t}\boldsymbol{\Lambda}_{k \to s_{j,k}} \right)^{-1} \left(\boldsymbol{\eta}_k + {}^{t}\boldsymbol{\eta}_{k \to s_{j,k}} \right), \\ {}^{t+1}\boldsymbol{\Lambda}_{s_{j,k} \to j} &= \boldsymbol{\Lambda}_{jj} - \boldsymbol{\Lambda}_{jk} \left(\boldsymbol{\Lambda}_{kk} + {}^{t}\boldsymbol{\Lambda}_{k \to s_{j,k}} \right)^{-1} \boldsymbol{\Lambda}_{kj}. \end{aligned} \tag{14}$$

The same procedure is used to send messages to variable k, however the information vector and precision matrix are reordered before marginalisation so the output variable is located at the top.

B. Implementation

1) Robust Factors: To reduce the influence of large residuals which could originate from outliers, it is common in practice to use M-estimators instead of Gaussian distributions to model the factor measurement functions. We replicate the behaviour of the Huber loss, which is quadratic around the minimum and grows linearly beyond a specified threshold N_{i_σ}. It is possible to replicate this behaviour using Gaussian distributions by rescaling the noise covariance of the Gaussian measurement models for residuals larger than N_{i_σ} [28]. This reduces the information of messages from these factors in a manner equivalent to the Huber norm [23], [27]:

$$f_i(x_i) = \begin{cases} \exp\left(-\frac{1}{2}M_i^2\right), & M_i \leq N_{i_\sigma} \\ \exp\left(-\frac{1}{2}M_i^2 \left[\frac{2N_\sigma}{M_i} - \frac{N_\sigma^2}{M_i^2}\right]\right), & M_i > N_{i_\sigma} \end{cases}, \tag{15}$$

where M_i is the Mahalanobis distance for the measurement and smoothing factors: $M_{m_i}(\boldsymbol{v}_i) = ||z_{m_i} - h_{m_i}(\boldsymbol{v}_i)||_{\Sigma_m}$ and $M_{s_{j,k}}(\boldsymbol{v}_j, \boldsymbol{v}_k) = ||\boldsymbol{v}_k - \boldsymbol{v}_j||_{\Sigma_s}$ respectively. We set different thresholds N_{m_σ} and N_{s_σ} for the two types of factors.

2) Image Pyramid: To improve the robustness of our system to larger motions, we implement scene flow estimation using a coarse-to-fine pyramid scheme. We use 4 levels in the pyramid with the resolutions: 30×40, 60×80, 120×160 and 240×320. We formulate separate factor graph optimisation problems for each level and use the solution from a coarser level of a pyramid to initialise the optimisation at the following finer level.

V. Factor Graph Design and Implementation

The IPU chip is massively parallel, consisting of 1216 tiles that are connected to an ultrafast all-to-all communication fabric which enables them to send data between each other at high speed. Each tile has 256kB local memory with a total of 300MB memory for the whole chip. Each tile can run 6 threads and execute different programs independently.

A. Factor Graph Design

Graph programs must be compiled before they can be run on the IPU. Compiled graphs can be saved to disk and reused when necessary. To make best use of the IPU, we design the structure of the factor graph to fit the problem and keep this structure fixed. We focus on three configurations for our system which differ in terms of the distribution of their smoothing factors (Fig. 3):

1) **Non-Rigid Scene Flow (BP-Flow)**: each variable is connected via 4 local smoothing factors to its direct neighbours in the directions: up, down, left and right. This configuration estimates non-rigid scene flow.
2) **Rigid Scene Flow (BP-VO)**: each variable is connected through global smoothing factors to 6 other variables that are sampled uniformly at random on the image plane. This is used to estimate rigid scene flow, which in static scenes is equivalent to visual odometry.
3) **Scene Flow with Varying Rigidity (BP-VRFlow)**: combines both formulations described above: each variable is connected to its 4 immediate neighbors and 6 other variables sampled from the image plane. This configuration supports estimating both rigid and deforming motion in a single image and can also replicate the behaviour of the other two configurations at increased computational cost (Table III).

Regarding BP-VRFlow, to disable regularisation in a particular region of the graph, we 'disable' unwanted smoothing factors by sending empty smoothing messages between the variables in that region. BP-VRFlow can simulate both BP-Flow and BP-VO by sending empty messages through global or local smoothing factors respectively. BP-VRFlow requires as input a segmentation mask to distinguish between non-rigidly moving objects and the rigid background.

In BP-VO, we experimented with other types of global smoothing factor configurations, such as hand-designed patterns. However, we found that these designs induce bias in the optimisation which results in unwanted symmetries in the

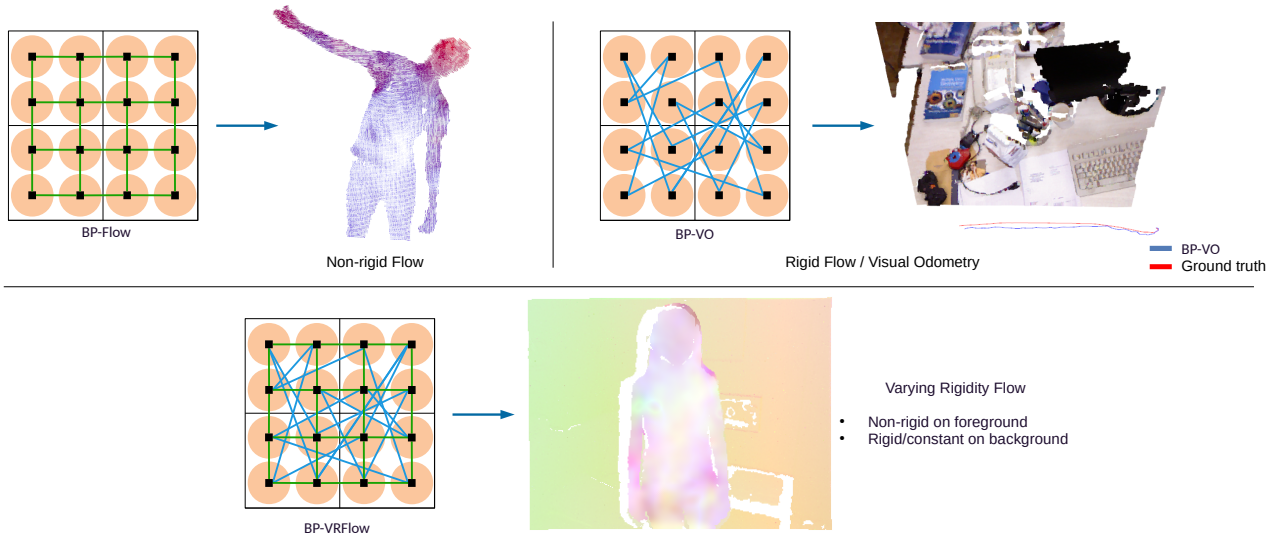

Fig. 3: The factor graph configurations we consider (with variables in orange, local smoothing in green, long range smoothing in blue): **BP-Flow** tackles non-rigid scene flow by implementing smoothing between neighbouring variables; **BP-VO** tackles visual odometry using smoothing factors between variables which are far from each other on the image plane. This constrains the flow field to be rigid, which in static scenes is equivalent to the camera motion. **BP-VRFlow** combines both local and global smoothing factors to estimate both rigid background motion and non-rigid foreground flow. BP-VRFlow requires a segmentation mask to control where regularisation is disabled (eg. between pixels belonging to the person and to the background). For illustration purposes, factor graphs are shown here to only consist of 4×4 variables. We implement multi-resolution factor graphs with a maximum resolution of 240×320 (Sec. IV-B2).

flow field. Ideally, we would implement smoothing factors between one 'anchor' variable and all the others. However, this produces an all-to-one communication pattern which is not feasible for real-time operation. Instead, uniform random sampling of smoothing factors is a practical solution which produces a smooth and uniform flow field.

To handle pixels with missing depth, we compute a validity mask representing variables with valid depth and who are connected to at least two other variables with valid depth. We then 'disable' smoothing factors with at least one invalid variable by sending empty messages.

B. IPU Implementation

When mapping the problem to the IPU, we minimise inter-tile communication by splitting the variables into small patches and mapping each patch to a different tile. Factor graphs in Fig. 3 show this, where the underlying grid represents a processor with 2×2 tiles, and patches of 2×2 variables are mapped to each tile. This way, smoothing factors between variables within the same patch perform all computation on the same tile. To compute the image warping required in the photometric factor (Eq. 4), we use a multislice function provided in the PopLibs™library which splits an input image and coordinates of warped pixels evenly across the tiles to retrieve the warped image efficiently.

Our scene flow estimation problem is significant, consisting of 6D variables for each pixel at every level of the pyramid. Most memory is spent on smoothing factors that invert two 6×6 matrices (Eq. 14). We use two IPU chips, as they are both located on the same board supporting fast data exchange. The new IPU MK2 has 900MB total memory, compared to the two MK1 IPUs we use which have 600MB in total, and we are confident that the next generation of this processor will enable us to run our method on a single chip.

VI. Results

We evaluate the scene flow configurations discussed in Section V-A: BP-Flow, BP-VO and BP-VRFlow. We evaluate each method on its capability: 1. BP-Flow on non-rigid scene flow, 2. BP-VO on visual odometry and 3. we demonstrate that BP-VRFlow supports both rigid and non-rigid motions in the same image assuming a segmentation mask of the non-rigidly moving objects in the scene is provided.

A. Non-Rigid Scene Flow (BP-Flow)

We compare BP-Flow to PD-Flow [2], a primal-dual scene flow method which is real-time on the GPU. PD-Flow is similar to our approach: it minimises the photometric error and uses local regularisation. Some differences are: 1. PD-Flow estimates one 3D motion vector per pixel, neglecting rotation; 2. both photometric and geometric errors are minimised.

We evaluate both approaches on the DeepDeform training set [21] which provides dense scene flow fields for a selection of RGB-D pairs of foreground objects undergoing deformations. Scene flow was computed through a combination of crowd sourcing of sparse correspondences and dense non-rigid flow estimation which is consistent with the sparse annotations. While this flow field is not perfect ground truth, it is of high accuracy and valuable for real data evaluation.

The dataset targets generic scene flow, including small and large inter-frame motion. We focus on real-time performance and make the assumption of small motion between consecutive frames. Both BP-Flow and PD-Flow implement the photometric error which has a small basin of convergence and requires small inter-frame motion to work well.

To support this, we process the DeepDeform training set and select RGB-D pairs with maximum scene flow values of 0.05, 0.1, 0.15, 0.2 and $0.25m$. We then compute the average per-point difference between the estimated flow field

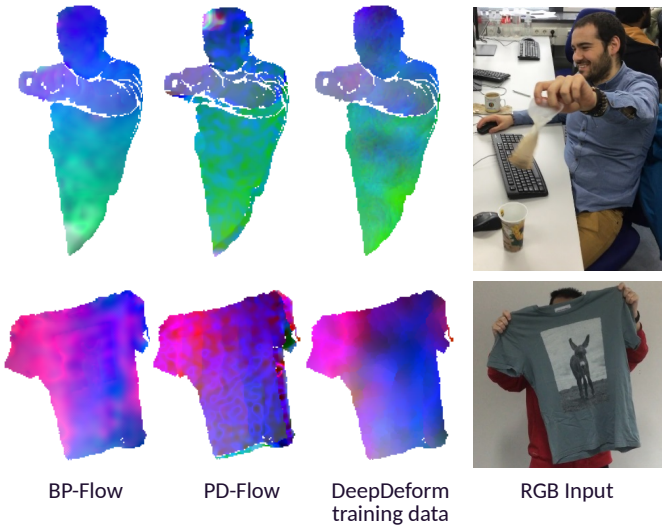

Fig. 4: Sample motion estimates of BP-Flow and PD-Flow on two DeepDeform sequences. BP-Flow and PD-Flow well recover the deformation fields of the foreground objects and produce motion fields which are consistent to the DeepDeform training data.

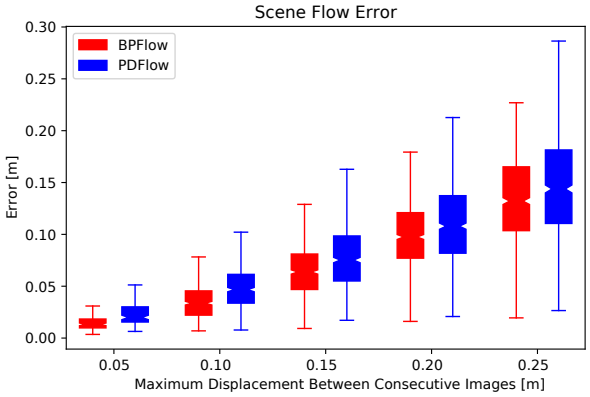

Fig. 5: Average per-point scene flow errors on DeepDeform sequences as a function of inter-frame motion. PD-Flow and BP-Flow perform similarly and are accurate for small inter-frame motions.

and the training set flow field for each image and report the performance of these two systems as a function of the inter-frame motion in Tab. I and in more detail in Fig. 5.

The two approaches achieve similar accuracy on this dataset, with best performance being achieved for small inter-frame motions. This means that our factor graph formulation is a viable alternative to similar variational methods for scene flow which use the photometric error and local regularisation.

Fig. 4 shows example flow fields from PD-Flow and BP-Flow. The methods provide similar flow fields which are consistent to reference flow fields from DeepDeform.

B. Rigid Scene Flow (BP-VO)

We first qualitatively compare BP-Flow and BP-VO to show the effect of the long range smoothing factors on the resulting flow field. In Fig. 6 we warp an RGB-D image pair with known $-0.02m$ motion on the X axis and plot histograms of the flow fields from BP-Flow and BP-VO for the translational degrees of freedom. The long range factors induce a much tighter distribution in the flow field from BP-VO, which is desirable for applying this method to rigid scenes. The motion

Average per-point error (m)					
	$< 0.05m$	$< 0.1m$	$< 0.15m$	$< 0.2m$	$< 0.25m$
PD-Flow	0.024	0.050	0.078	0.112	0.147
BP-Flow	**0.015**	**0.034**	**0.062**	**0.101**	**0.133**
Imgs. no.	113	439	690	841	862

TABLE I: Average per-point scene flow errors for image pairs with increasing inter-frame motion. We also state the number of RGB-D pairs within the DeepDeform training set identified for each category of inter-frame motion and used during evaluation.

estimate from BP-VO is also more accurate as the median translational quantity on the X axis is $-0.015m$. This is due to the global information exchange enabled by the long range smoothing factors, which aids the solver in achieving a consistent estimate.

We quantitatively evaluate BP-VO on the TUM dataset [1]. We report results for: 1. the mean scene flow estimate by taking the average of all estimated flow vectors (BP-VO (mean)) and 2. a single estimate from the central pixel (BP-VO (sample)). The second evaluation shows that our method can be used as visual odometry, without further post-processing.

We compare the accuracy of our method against related approaches: 1. DVO [29], a visual odometry method designed for static scenes.; 2. VO-SF [14] an odometry method designed for dynamic scenes which rigidly tracks moving objects; 3. StaticFusion [30], a visual SLAM method which segments moving objects and reconstructs the background, 4. FlowFusion [31], a visual SLAM method which segments moving objects using optical flow and reconstructs the background.

We report the translational relative pose error RMS over static scenes in Table II. Although our method is slightly less accurate compared to dedicated visual odometry, this is a significant result, as we are not aware of any other scene flow method that is comparable to dedicated visual odometry.

The accuracy of BP-VO could be improved through a better rotation parametrisation similar to visual odometry formulations. While we estimate the full rotation, visual odometry methods repeatedly warp one image towards the other and relinearise the solver around zero motion. Replicating this in Belief Propagation is left for future work.

C. Scene Flow with Varying Rigidity (BP-VRFlow)

BP-VRFlow has a flexible structure that supports both rigid and deforming motions in the same image. It requires a segmentation mask to disable regularisation between foreground and background. Fig. 7 shows motion estimates from BP-VRFlow in a scene with a moving person. We simulate a change in the smoothing factors by sending empty smoothing messages through all global factors connected to foreground variables. As a result, motion estimates on the wall are much smoother than those produced by PD-Flow and BP-Flow, while foreground objects maintain non-rigidity.

We evaluate BP-VRFlow quantitatively on the TUM dataset (Tab. II). In static environments, BP-VRFlow simulates BP-VO by disabling all local regularisation factors. In dynamic environments, we use MaskRCNN [32] to segment the moving people and chairs which enables BP-VRflow to estimate visual odometry using background variables and also non-rigid flow for the foreground. Segmentations are computed offline for each frame and read during run-time with the RGB-D input.

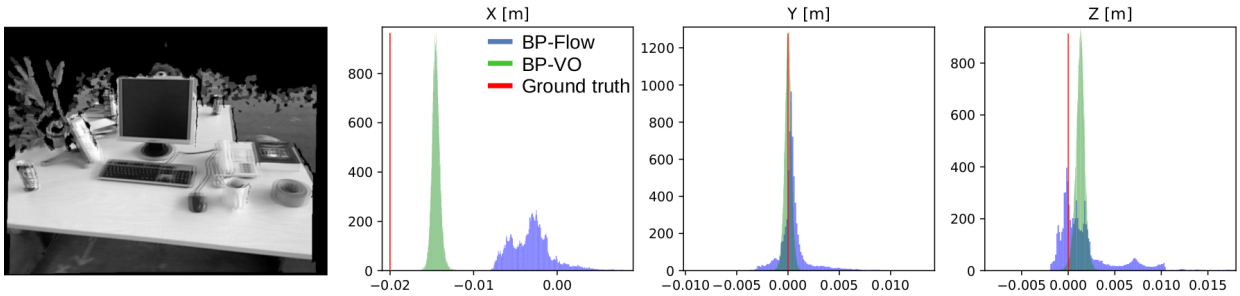

Fig. 6: Qualitative example of the impact of long range smoothing factors on the distribution of the flow field. **Left:** we warp an RGB-D image with a known motion of $-0.02m$ in the lateral X direction and display the overlaid original and warped image. **Right:** a histogram of the distribution of the flow fields from BP-Flow and BP-VO for the translational degrees of freedom, where red marks the ground truth motion estimate. The distribution from BP-VO is tighter and closer to the actual solution. This is because long distance smoothing messages increase the information that each pixel has access to and allow the method to converge to a solution which is globally consistent.

Trans. RPE RMSE (m/s)								
		IPU			CPU/GPU			
Seq.		BP-VO (mean)	BP-VO (sample)	BP-VRFlow (sample)	DVO [29]	VO-SF [14]	StaticFusion [30]	FlowFusion [31]
Static Env.	fr1/desk	0.077	0.077	0.077	0.04	**0.021**	0.03	-
	fr1/floor	0.073	0.075	0.075	0.09	**0.058**	0.08	-
	fr1/plant	0.065	0.068	0.068	**0.036**	0.06	0.104	-
	fr1/room	0.076	0.076	0.076	0.058	**0.052**	0.072	-
	fr1/teddy	0.08	0.083	0.083	0.096	**0.065**	0.10	-
	fr1/xyz	0.056	0.056	0.056	0.026	**0.021**	0.023	0.023
	fr2/desk	0.031	0.035	0.035	**0.018**	0.02	**0.018**	-
Dyn. Env.	fr3/w_static	0.32	0.34	0.015	0.31	0.11	**0.013**	0.03
	fr3/w_xyz	0.46	0.46	**0.079**	0.48	0.27	0.121	0.21

TABLE II: Translational RMSE. BP-VO is slightly less accurate compared to dedicated visual odometry approaches, however the results are significant given the distributed nature of the approach. BP-VRFlow can simulate BP-VO in static scenes, while in dynamic scenes it can estimate both visual odometry and scene flow assuming it is provided a segmentation of the foreground.

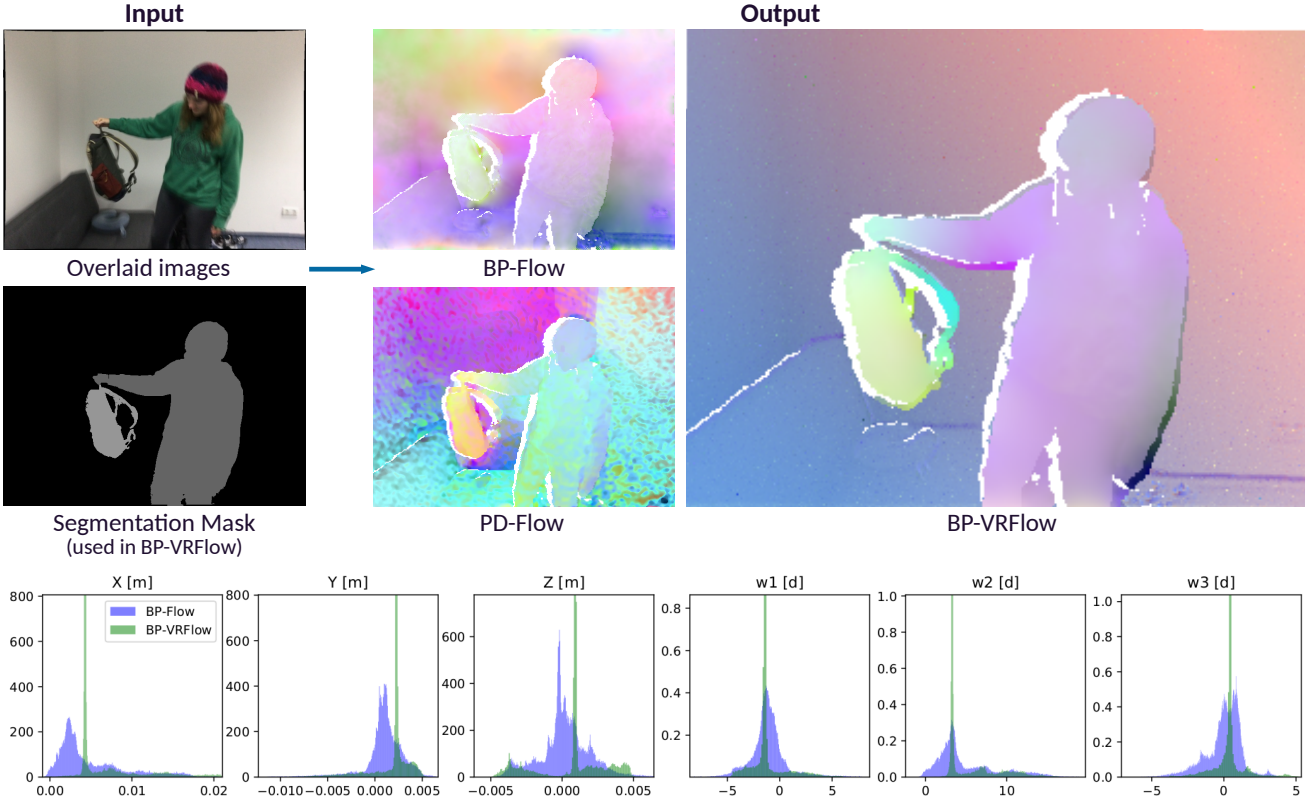

Fig. 7: Qualitative results of our flow system running with varying rigidity. **Top left**: overlaid input images, segmentation mask from DeepDeform, flow fields estimated using BP-Flow and PD-Flow. **Top right**: The flow estimated by BP-VRFlow is smooth in the background and non-rigid on the foreground. This is enabled by our flexible factor graph structure, where global smoothing messages are only sent for variables belonging to the background. **Bottom**: Flow distribution histogram for BP-Flow and BP-VRFlow for each degree of freedom (rotation is expressed in degrees). While BP-Flow has a broad distribution, BP-VRFlow shows a clear peak corresponding to the camera motion and smaller broad distributions representing the motions of the people.

Run-time						
	IPU			GPU		CPU
	BP-Flow	BP-VO	BP-VRFlow	PD-Flow	SF	VO-SF
Hz	17	14	10	30	30	12

TABLE III: Run-time of our system and related approaches.

D. Timing Analysis

Table III states the runtimes of the different configurations of our system compared to related approaches. The BP-VRFlow processing times do not include the MaskRCNN segmentation computation. The graph program must be re-compiled if its structure is changed during run-time. We maintain the structures of our graphs fixed, but also state their compilation timings for completeness: $\sim$ 2mins for BP-Flow, $\sim$ 3mins for BP-VO and $\sim$ 5mins for BP-VRFlow. After compilation, programs are written to file and read once at the start of the application, which takes around 1 second.

VII. Discussion and Conclusion

We demonstrated that a simple MRF-type formulation with only unary measurement and pairwise smoothing factors can be used to estimate non-rigid, rigid and piece-wise rigid scene flow. The behaviour of the algorithm can be altered by changing the configuration of the smoothing factors and global properties such as the camera motion can be estimated through purely local computations. This is an important step in the direction of distributed motion estimation and in future work we will improve the accuracy of BP-VO through better rotation parametrisaion.

Our method is appealing due to its simplicity. Other smoothing factor configurations could also be considered, for example if the scene is known to contain mostly piece-wise rigid objects, local smoothing could be extended to a larger neighbourhood beyond the immediate four neighbours. We also aim to integrate motion segmentation into the method to explicitly segment differently moving objects in the scene based on similarities in the flow field and use this information to enable or disable global smoothing factors as required.

References

[1] J. Stürm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2012.

[2] M. Jaimez, M. Souiai, J. Gonzalez-Jimenez, and D. Cremers, "A primal-dual framework for real-time dense RGB-D scene flow," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[3] GraphCore IPU, https://www.graphcore.ai/products/ipu.

[4] S. Vedula, P. Rander, R. Collins, and T. Kanade, "Three-dimensional scene flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 27, no. 3, pp. 475–480, 2005.

[5] A. Wedel, T. Brox, T. Vaudrey, C. Rabe, U. Franke, and D. Cremers, "Stereoscopic scene flow computation for 3D motion understanding," *International Journal of Computer Vision (IJCV)*, vol. 95, no. 1, pp. 29–51, 2011.

[6] T. Basha, Y. Moses, and N. Kiryati, "Multi-view scene flow estimation: A view centered variational approach," *International Journal of Computer Vision (IJCV)*, vol. 101, no. 1, pp. 6–21, 2013.

[7] F. Huguet and F. Devernay, "A variational method for scene flow estimation from stereo sequences," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2007.

[8] E. Herbst, X. Ren, and D. Fox, "RGB-D Flow: Dense 3-D motion estimation using color and depth," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[9] M. Isard and J. MacCormick, "Dense motion and disparity estimation via loopy belief propagation," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2006.

[10] M. Hornáček, F. Besse, J. Kautz, A. Fitzgibbon, and C. Rother, "Highly overparameterized optical flow using patchmatch belief propagation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[11] C. Vogel, K. Schindler, and S. Roth, "3D scene flow estimation with a piecewise rigid scene model," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 1, pp. 1–28, 2015.

[12] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[13] D. Sun, E. B. Sudderth, and H. Pfister, "Layered RGBD scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[14] M. Jaimez, C. Kerl, J. Gonzalez-Jimenez, and D. Cremers, "Fast odometry and scene flow from RGB-D cameras based on geometric clustering," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[15] J. Quiroga, T. Brox, F. Devernay, and J. Crowley, "Dense semi-rigid scene flow estimation from RGBD images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[16] W. Ma, S. Wang, R. Hu, Y. Xiong, and R. Urtasun, "Deep rigid instance scene flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[17] X. Liu, C. R. Qi, and L. J. Guibas, "Flownet3D: Learning scene flow in 3D point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[18] I. Tishchenko, S. Lombardi, M. R. Oswald, and M. Pollefeys, "Self-supervised learning of non-rigid residual flow and ego-motion," *Proceedings of the International Conference on 3D Vision (3DV)*, 2020.

[19] R. A. Newcombe, D. Fox, and S. M. Seitz, "DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[20] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger, "VolumeDeform: Real-time volumetric non-rigid reconstruction," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[21] A. Božič, M. Zollhofer, C. Theobalt, and M. Nießner, "DeepDeform: Learning non-rigid RGB-D reconstruction with semi-supervised data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[22] A. Božič, P. Palafox, M. Zollhöfer, A. Dai, J. Thies, and M. Nießner, "Neural non-rigid tracking," in *Neural Information Processing Systems (NIPS)*, 2021.

[23] A. J. Davison and J. Ortiz, "FutureMapping 2: Gaussian Belief Propagation for Spatial AI," *arXiv:arXiv:1910.14139*, 2019.

[24] J. Ortiz, T. Evans, and A. J. Davison, "A visual introduction to gaussian belief propagation," *arXiv preprint arXiv:2107.02308*, 2021.

[25] R. M. Eustice, H. Singh, and J. J. Leonard, "Exactly Sparse Delayed State Filters," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[26] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

[27] J. Ortiz, M. Pupilli, S. Leutenegger, and A. J. Davison, "Bundle adjustment on a graph processor," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[28] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[29] C. Kerl, J. Stürm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[30] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "StaticFusion: Background reconstruction for dense rgb-d slam in dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[31] T. Zhang, H. Zhang, Y. Li, Y. Nakamura, and L. Zhang, "FlowFusion: Dynamic dense RGB-D slam based on optical flow," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[32] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.