Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

3D scene flow estimation based on a variational functional

Artyom Makovetskii

artemmac@csu.ru

Chelyabinsk State University

Vitaly Kober

Center for Scientific Research and Higher Education of Ensenada

Sergei Voronin

Chelyabinsk State University

Aleksei Voronin

Chelyabinsk State University

Research Article

Keywords: 3D scene flow estimation, Motion prediction, Point cloud, Scene reconstruction, Registration, Affne transformation, Spatial regularization

Posted Date: November 23rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11720864/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

3D scene flow estimation based on a variational functional

Artyom Makovetskii[1*], Vitaly Kober[2,3], Sergei Voronin[1], Aleksei Voronin[1]

[1]Department of Mathematics, Chelyabinsk State University, 129 Br. Kashirinykh, Chelyabinsk, 454001, Russian Federation.
[2]Center for Scientific Research and Higher Education of Ensenada, Ensenada, B.C., 22860, Mexico.
[3]Institute for Information Transmission Problems, Russian Academy of Sciences, Moscow, 127051, Russia.

*Corresponding author(s). E-mail(s): artemmac@csu.ru;
Contributing authors: vkober@cicese.mx; voron@csu.ru;
ununus@mail.ru;

Abstract

This paper presents a variational approach to 3D scene flow estimation based on a modified functional originally developed for non-rigid point cloud registration. proposed method associates an individual affine transformation with each point of the source point cloud, allowing local 3D displacements to be estimated independently while maintaining spatial consistency between neighboring points. The variational functional consists of a data term and a regularization term with adaptive weighting coefficients. In particular, different rigidity coefficients are used depending on whether neighboring points belong to static or dynamic regions of the scene. The resulting variational problem is reduced to a system of linear equations, which is solved numerically without iterative nonlinear optimization. The proposed algorithm is evaluated using the Argoverse 2 dataset and compared with the deep-learning-based DeFlow method. The experiments include both consecutive LiDAR scans and point cloud pairs separated by multiple time steps, allowing the performance of the methods to be investigated under different conditions of geometric congruence and correspondence reliability.

Keywords: 3D scene flow estimation; Motion prediction; Point cloud; Scene reconstruction; Registration; Affine transformation; Spatial regularization.

1 Introduction

Point cloud registration is an important component of 3D scene reconstruction, analysis, and dynamic environment perception. It is widely used in robotics, autonomous navigation, object recognition, and 3D mapping. The main objective of point cloud registration is to estimate a geometric transformation that aligns two point clouds representing the same physical scene or object. One of the most widely used approaches to this problem is the Iterative Closest Point (ICP) algorithm [1, 2]. Since its introduction, numerous modifications and extensions of ICP have been proposed to improve its accuracy, robustness, and computational efficiency. These include methods based on different distance metrics, geometric constraints, and transformation models [3–15]. In particular, variational formulations of the registration problem provide a convenient mathematical framework for incorporating additional regularization terms and prior information into the estimation process [10–15].

An important extension of rigid point cloud registration is non-rigid registration, in which different regions of a point cloud are allowed to undergo different geometric transformations. Unlike rigid registration, which assumes that the entire point cloud can be aligned using a single global transformation, non-rigid registration allows local deformations and independent movements of different parts of an object or scene. One of the well-known approaches to this problem is the non-rigid ICP algorithm proposed in [16]. In this method, local affine transformations are associated with individual points, while regularization terms are introduced to ensure spatial consistency between transformations corresponding to neighboring points. Related variational approaches to non-rigid point cloud registration have also been investigated in [17, 18]. Such formulations provide a basis for describing spatially varying geometric transformations and are therefore potentially useful for analyzing dynamic 3D environments.

A closely related problem is 3D scene flow estimation, which aims to determine the three-dimensional motion of points in a dynamic scene. Given two point clouds acquired at different time instants, the objective is to estimate a displacement vector for each point in the first point cloud, describing its motion between the two observations. The scene flow estimation problem can be formulated as finding a set of displacement vectors for each point of the source The resulting set of vectors represents a dense 3D motion field associated with the observed scene.

In contrast to conventional point cloud registration, which generally estimates a single transformation for the entire point cloud, scene flow estimation must account for the possibility that different objects and different regions of the scene move independently. Moreover, the point clouds acquired at successive time instants usually contain different sets of points because of occlusions, changes in sensor visibility, and the non-uniform spatial distribution of LiDAR measurements. Consequently, establishing reliable correspondences between points in successive scans is a challenging part of the problem. Additional difficulties arise from the presence of both static and dynamic objects, different motion patterns, and variations in point cloud density.

Scene flow estimation has become an important research direction in computer vision and robotics, particularly in connection with autonomous driving. The estimated motion field can be used for dynamic object detection, motion segmentation, object tracking, collision prediction, and understanding the behavior of surrounding

objects. In autonomous driving applications, the ability to distinguish stationary structures from independently moving objects is especially important. Unlike object-level motion estimation, which describes the movement of entire objects, point-wise scene flow provides a more detailed representation of local 3D motion.

Recent developments in deep learning have resulted in a number of methods that estimate scene flow directly from point clouds. One of the pioneering approaches is FlowNet3D [19], which employs a neural network to extract hierarchical point cloud features and estimate point-wise motion vectors. Subsequent approaches have introduced different architectures for processing large-scale point clouds and establishing relationships between points in consecutive scans. For example, HPLFlowNet [20] uses permutohedral lattice representations to process unstructured point clouds efficiently. These methods demonstrate the potential of deep neural networks for estimating complex 3D motion fields without explicitly specifying geometric transformations for individual points. However, the problem remains challenging due to the non-uniform distribution of LiDAR measurements, occlusions, large displacements, and the significant difference between the numbers of static and dynamic points in real-world driving scenes. A comprehensive overview of deep learning approaches to point cloud scene flow estimation can be found in [21].

In this paper, we propose a variational approach to 3D scene flow estimation (VF) based on a modification of the functional traditionally used in non-rigid point cloud registration. The proposed method establishes a connection between non-rigid registration and point-wise motion estimation by associating an individual affine transformation with each point in the source point cloud. In this way, the local displacement of a point is determined by the affine transformation assigned to that point, rather than by a single global transformation applied to the entire point cloud.

The proposed algorithm consists of two principal stages. First, correspondences between the source and target point clouds are established using a nearest-neighbor search. Second, a set of local affine transformations is estimated by minimizing a variational functional. The functional contains two terms. The first term measures the discrepancy between the transformed source points and their corresponding points in the target point cloud. The second term penalizes differences between the transformations associated with neighboring points, thereby imposing spatial consistency on the estimated motion field.

An important feature of the proposed formulation is the introduction of weighting coefficients into both terms of the functional. The coefficients associated with the data term reflect the reliability of the established point correspondences: higher values are assigned to correspondences that are considered more reliable. The coefficients associated with the regularization term control the degree of rigidity imposed on neighboring points. The inclusion of individual weighting coefficients for all pairs of corresponding points and all pairs of neighboring points, along with the corresponding changes in the structure of the solution to the variational problem, distinguishes the proposed functional from the original [16].

In particular, a higher rigidity coefficient is used when both neighboring points belong to the dynamic part of the scene, whereas a lower coefficient is assigned when one point belongs to the dynamic part and the other belongs to the static part. This

allows the regularization strength to be adapted to the local structure of the scene and reduces the influence of smoothness constraints across boundaries between static and moving regions. The proposed variational problem admits a closed-form solution in terms of a system of linear equations. Therefore, the estimation of local affine transformations does not require iterative nonlinear optimization. The resulting system is solved numerically, providing the transformation parameters and, consequently, the displacement vectors for the source points. The use of a linear system makes the proposed formulation mathematically tractable.

For experimental evaluation, we use the Argoverse 2 dataset [22], which was developed for research in autonomous driving and 3D scene perception. The dataset provides multimodal sensor information, including LiDAR point clouds, camera images, and vehicle poses, as well as annotations of surrounding objects. In particular, the Argoverse 2 scene flow benchmark provides consecutive LiDAR scans and corresponding ground-truth motion information, enabling quantitative evaluation of point-wise 3D displacement estimates. The dataset includes both static and dynamic regions, which makes it suitable for investigating the influence of different local motion patterns on scene flow estimation.

It should be emphasized that the proposed VF method is not an end-to-end scene flow estimation algorithm. In the experiments presented in this paper, we use ground-truth information about the classification of points into static and dynamic regions provided by the Argoverse 2 scene flow annotations. At first, this information is used to exclude static points from point clouds. At second, the information is used to assign the rigidity coefficients in the regularization term. The automatic identification of static and dynamic points is considered a separate problem and can be addressed using an independent neural-network-based segmentation algorithm. Thus, the present study focuses primarily on the variational estimation of point-wise displacements, assuming that the static/dynamic classification is available.

To evaluate the proposed method, we compare its performance with DeFlow [23], a deep-learning-based scene flow estimation algorithm developed for autonomous driving applications. DeFlow employs a neural network architecture that combines voxel-based feature extraction with point-wise feature refinement and has been evaluated on the Argoverse 2 Scene Flow benchmark. In our experiments, the comparison is performed on several pairs of LiDAR point clouds. In addition to pairs of consecutive scans, we consider point cloud pairs with different degrees of geometric congruence. This makes it possible to investigate the behavior of the proposed variational approach under different conditions of correspondence reliability and geometric overlap. In particular, we apply the algorithms under consideration not only to point clouds with indices n and $n+1$ in a sequence of LiDAR scans, but also to point clouds with indices n and $n+k$, where k is greater than one.

The main objective of this study is to investigate the applicability of variational functionals, originally developed for non-rigid point cloud registration, to the problem of 3D scene flow estimation. Particular attention is paid to the influence of spatial regularization and the use of different rigidity coefficients for static and dynamic regions. The experimental comparison with DeFlow provides an opportunity to assess the effectiveness of the proposed approach relative to a modern neural-network-based method.

2 The Variational Functional Utilizing Locally Affine Transformations

Let $P = \{p_0, \ldots, p_{s_p-1}\}$ be a source point cloud, and $Q = \{q_0, \ldots, q_{s_q-1}\}$ be a target point cloud in $\mathbb{R}^3$, where s_p is the number of points in P, s_q is the number of points in Q. Let us consider the such nearest neighborhoods $\{p_{i_0}, \ldots, p_{i_{N(i)-1}}\}$ of the point p_i that $\| p_i - p_{i_j} \|_{L_2} < R$, where R is the radius of the neighborhood, $N(i)$ is the quantity of points in the neighborhood, $j = 0, \ldots, N(i) - 1$. We shall say that the point p_i is connected by (virtual) edges to points in its neighborhood. Note that if point p_j is a neighbor of point p_i, then p_i is also a neighbor of p_j. Suppose that the relationship between points in P and Q is given in such a manner that for each point p_i exists a corresponding point q_i. Let $J(A_0, \ldots, A_{s_p-1})$ be the following functional:

$$J\left(A_0, \ldots, A_{s_p-1}\right) = J_1\left(A_0, \ldots, A_{s_p-1}\right) + J_2\left(A_0, \ldots, A_{s_p-1}\right) \tag{1}$$

where

$$J_1\left(A_0, \ldots, A_{s_p-1}\right) = \sum_{i=0}^{s_p-1} \alpha_i \|A_i p_i - q_i\|^2 \tag{2}$$

$$J_2\left(A_0, \ldots, A_{s_p-1}\right) = \sum_{\{i,j\} \in E} \beta_{ij} \|A_i - A_j\|^2 \tag{3}$$

A_i is a matrix of an affine transformation in the homogenous coordinates,

$$A_i = \begin{pmatrix} a_i^{11} & a_i^{12} & a_i^{13} & T_i^1 \\ a_i^{21} & a_i^{22} & a_i^{23} & T_i^2 \\ a_i^{31} & a_i^{32} & a_i^{33} & T_i^3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

p_i and q_i are points from the cloud P and Q respectively in the homogenous coordinates

$$p_i = \left(p_i^1 p_i^2 p_i^3 1\right)^t, q_i = \left(q_i^1 q_i^2 q_i^3 1\right)^t \tag{5}$$

$\mathcal{E}$ is the set of (virtual) edges of the point cloud P, $\alpha_i > 0$ and $\beta_{ij} > 0$ are the parameters.

Let us formulate the following variational problem:

$$\left((A_0)_*, \ldots, (A_{s_p-1})_*\right) = \arg \min_{A_0, \ldots, A_{s_p-1}} J\left(A_0, \ldots, A_{s_p-1}\right) \tag{6}$$

Compute the gradient ∇J_1 of the functional J_1

$$\nabla J_1 = \left((\nabla J_1)_0, \ldots, (\nabla J_1)_{s_p-1}\right) \tag{7}$$

Consider the increment of the functional J_1 with respect to the variable A_i

$$J_1\left(A_0, \ldots, A_i + h, \ldots, A_{s_p-1}\right) - J_1\left(A_0, \ldots, A_i, \ldots, A_{s_p-1}\right) \tag{8}$$

The terms $J_1\left(A_0, \ldots, A_i + h, \ldots, A_{s_p-1}\right)$ and $J_1\left(A_0, \ldots, A_i, \ldots, A_{s_p-1}\right)$ are the following sums:

$$J_1\left(A_0,\ldots,A_i+h,\ldots,A_{s_p-1}\right)=\ldots+\alpha_i\langle(A_i+h)p_i-q_i,(A_i+h)p_i-q_i\rangle+\ldots \quad (9)$$

$$J_1\left(A_1,\ldots,A_i,\ldots,A_s\right)=\ldots+\alpha_i\langle A_ip_i-q_i,A_ip_i-q_i\rangle+\ldots \quad (10)$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product. Consider the marked element of the sum separately

$$\begin{aligned}\alpha_i\langle(A_i+h)\,p_i-q_i,(A_i+h)\,p_i-q_i\rangle&=\alpha_i\langle(A_i+h)\,p_i,(A_i+h)\,p_i\rangle\\&-2\alpha_i\langle(A_i+h)\,p_i,q_i\rangle+\alpha_i\langle q_i,q_i\rangle=\\&=\alpha_i\langle A_ip_i,A_ip_i\rangle+2\alpha_i\langle A_ip_i,hp_i\rangle+\alpha_i\langle hp_i,hp_i\rangle\\&-2\alpha_i\langle A_ip_i,q_i\rangle-2\alpha_i\langle hp_i,q_i\rangle+\alpha_i\langle q_i,q_i\rangle\end{aligned} \quad (11)$$

and

$$\alpha_i\langle A_ip_i-q_i,A_ip_i-q_i\rangle=\alpha_i\langle A_ip_i,A_ip_i\rangle-2\alpha_i\langle A_ip_i,q_i\rangle+\alpha_i\langle q_i,q_i\rangle \quad (12)$$

The increment of the functional J_1 with respect to A_i

$$J_1\left(A_1,\ldots,A_i+h,\ldots,A_s\right)-J_1\left(A_1,\ldots,A_i,\ldots,A_n\right)=2\alpha_i\langle A_ip_i-q_i,hp_i\rangle+o(h) \quad (13)$$

The component $\left(\nabla J_1\right)_i$ of the gradient ∇J_1 takes the following form:

$$\left(\nabla J_1\right)_i=2\alpha_i(A_ip_i-q_i)p_i^t \quad (14)$$

Compute the gradient ∇J_2 of the functional J_2

$$\nabla J_2=\left(\left(\nabla J_2\right)_0,\ldots,\left(\nabla J_2\right)_{s_p-1}\right) \quad (15)$$

Consider the increment of the functional J_2 with respect to A_i

$$J_2\left(A_0,\ldots,A_i+h,\ldots,A_{s_p-1}\right)-J_2\left(A_0,\ldots,A_{s_p-1}\right) \quad (16)$$

The terms $J_2\left(A_0,\ldots,A_i+h,\ldots,A_{s_p-1}\right)$ and $J_2\left(A_0,\ldots,A_i,\ldots,A_{s_p-1}\right)$ are the following sums:

$$J_2\left(A_0,\ldots,A_i+h,\ldots,A_{s_p-1}\right)=\ldots+\sum_{j=0}^{N(i)-1}\beta_{i,L(i)_j}\|(A_i+h)-A_{L(i)_j}\|^2+\ldots \quad (17)$$

and

$$J_2\left(A_0,\ldots,A_i,\ldots,A_{s_p-1}\right)=\ldots+\sum_{j=0}^{N(i)-1}\beta_{i,L(i)_j}\|A_i-A_{L(i)_j}\|^2+\ldots \quad (18)$$

where $L(i)$ is the list of neighbors of the point p_i, $L(i)_j$ is the j-th element of the list. Consider the marked element of the sum separately

$$\sum_{j=0}^{N(i)-1} \beta_{i,L(i)_j} \|(A_i + h) - A_{L(i)_j}\|^2$$
$$= \sum_{j=0}^{N(i)-1} \beta_{i,L(i)_j} \langle (A_i + h) - A_{L(i)_j}, (A_i + h) - A_{L(i)_j} \rangle =$$
$$= \sum_{j=0}^{N(i)-1} (\beta_{i,L(i)_j} \langle A_i, A_i \rangle + 2\beta_{i,L(i)_j} \langle A_i, h \rangle + \beta_{i,L(i)_j} \langle h, h \rangle - 2\beta_{i,L(i)_j} \langle A_i, A_{L(i)_j} \rangle$$
$$- 2\beta_{i,L(i)_j} \langle h, A_{L(i)_j} \rangle + \beta_{i,L(i)_j} \langle A_{L(i)_j}, A_{L(i)_j} \rangle) \tag{19}$$

and

$$\sum_{j=0}^{N(i)-1} \beta_{i,L(i)_j} \|A_i - A_{L(i)_j}\|^2 = \sum_{j=0}^{N(i)-1} \beta_{i,L(i)_j} \langle A_i - A_{L(i)_j}, A_i - A_{L(i)_j} \rangle =$$
$$= \sum_{j=0}^{N(i)-1} (\beta_{i,L(i)_j} \langle A_i, A_i \rangle - 2\beta_{i,L(i)_j} \langle A_i, A_{L(i)_j} \rangle + \beta_{i,L(i)_j} \langle A_{L(i)_j}, A_{L(i)_j} \rangle) \tag{20}$$

The increment of the functional J_2 with respect to A_i

$$J_2 \left(A_0, \ldots, A_i + h, \ldots, A_{s_p-1}\right) - J_2 \left(A_0, \ldots, A_i, \ldots, A_{s_p-1}\right) =$$
$$= \sum_{j=0}^{N(i)-1} (2\beta_{i,L(i)_j} \langle A_i, h \rangle + \langle h, h \rangle - 2\beta_{i,L(i)_j} \langle h, A_{L(i)_j} \rangle) =$$
$$= \sum_{j=0}^{N(i)-1} \langle 2\beta_{i,L(i)_j} (A_i - A_{L(i)_j}), h \rangle + o(h) \tag{21}$$
$$= \langle 2 \sum_{j=0}^{N(i)-1} \beta_{i,L(i)_j} (A_i - A_{L(i)_j}), h \rangle + o(h)$$

The component $(\nabla J_2)_i$ of the gradient ∇J_2 has the following form:

$$(\nabla J_2)_i = 2 \sum_{j=0}^{N(i)-1} \beta_{i,L(i)_j} (A_i - A_{L(i)_j}) \tag{22}$$

The component $(\nabla J)_i$ of the gradient ∇J

$$(\nabla J)_i = (\nabla J_1)_i + (\nabla J_2)_i = 2\alpha_i (A_i p_i - q_i) p_i^t + 2 \sum_{j=0}^{N(i)-1} \beta_{i,L(i)_j} (A_i - A_{L(i)_j}) \tag{23}$$

Since $\nabla J = 0$

$$\alpha_i(A_i p_i - q_i)p_i^t + \sum_{j=0}^{N(i)-1} \beta_{i,L(i)_j}(A_i - A_{L(i)_j}) = 0 \tag{24}$$

$$\alpha_i A_i p_i p_i^t + \sum_{j=0}^{N(i)-1} \beta_{i,L(i)_j}(A_i - A_{L(i)_j}) = \alpha_i q_i p_i^t \tag{25}$$

Let matrices P_k and Q_k be defined as

$$P_i = p_i p_i^t, Q_i = \alpha_i q_i p_i^t \tag{26}$$

where $i = 0, \ldots, s_p - 1$. We obtain the following system of equations

$$\begin{cases} \alpha_0 A_0 P_0 + \sum_{j=0}^{N(0)-1} \beta_{0,L(0)_j}(A_0 - A_{L(0)_j}) = Q_0 \\ \ldots \\ \alpha_{s_p-1} A_{s_p-1} P_{s_p-1} + \sum_{j=0}^{N(s_p-1)-1} \beta_{s_p-1,L(s_p-1)_j}(A_{s_p-1} - A_{L(s_p-1)_j}) = Q_{s_p-1} \end{cases} \tag{27}$$

Rewrite the system (27) as

$$\begin{cases} A_0\left(\alpha_0 P_0\right) + A_0 \sum_{j=0}^{N(0)-1} \beta_{0,L(0)_j} - \sum_{j=0}^{N(0)-1} \beta_{0,L(0)_j} A_{L(0)_j} = Q_0 \\ \ldots \\ A_{s_p-1}\left(\alpha_{s_p-1} P_{s_p-1}\right) + A_{s_p-1} \sum_{j=0}^{N(s_p-1)-1} \beta_{s_p-1,L(s_p-1)_j} \\ \quad - \sum_{j=0}^{N(s_p-1)-1} \beta_{s_p-1,L(s_p-1)_j} A_{L(s_p-1)_j} = Q_{s_p-1} \end{cases} \tag{28}$$

Let us denote by $\widehat{P}_i$ and $\widehat{\beta}_i$ the following matrix and number, respectively

$$\widehat{P}_i = \alpha_i P_i, \widehat{\beta}_i = \sum_{j=0}^{N(i)-1} \beta_{i,L(i)_j} \tag{29}$$

where $i = 0, \ldots, s_p - 1$. The system (28) takes the form

$$\begin{cases} A_0\widehat{P_0} + A_0\widehat{\beta_0} - \sum\limits_{j=0}^{N(0)-1} \beta_{0,L(0)_j} A_{L(0)_j} = Q_0 \\ \dots \\ A_{s_p-1}\widehat{P_{s_p-1}} + A_{s_p-1}\widehat{\beta_{s_p-1}} - \sum\limits_{j=0}^{N(s_p-1)-1} \beta_{s_p-1,L(s_p-1)_j} A_{L(s_p-1)_j} = Q_{s_p-1} \end{cases} \tag{30}$$

Let us introduce the following notations

$$P'_i = \widehat{P}_i + \widehat{\beta}_i I, B_{i,L(i)_j} = \beta_{i,L(i)_j} I \tag{31}$$

where $I = diag(1,1,1,1)$. The system takes the form

$$\begin{cases} A_0 P'_0 - \sum\limits_{j=0}^{N(0)-1} A_{L(0)_j} B_{0,L(0)_j} = Q_0 \\ \dots \\ A_{s_p-1} P'_{s_p-1} - \sum\limits_{j=0}^{N(s_p-1)-1} A_{L(s_p-1)_j} B_{s_p-1,L(s_p-1)_j} = Q_{s_p-1} \end{cases} \tag{32}$$

3 The System of Equations and Its Properties

Formula (32) represents a system of linear equations in terms of matrices. Let us define a system of linear equations using numerical variables. The matrix of the system (32) has size $(s_p \times s_p)$. We take the first three rows from all the matrices (these matrices are of size (4×4)) in (32) and obtain a numerical matrix of size $(12s_p \times 12s_p)$. The numerical system can be written as

$$MA = Q \tag{33}$$

where M is matrix of size $(12s_p \times 12s_p)$, A is vector-column of size $12s_p$, Q is vector-column of size $12s_p$,

$$\begin{aligned} A = ((A_0)_{00}\,(A_0)_{01}\,(A_0)_{02}\,(A_0)_{03}\,(A_0)_{10}\,(A_0)_{11} \\ (A_0)_{12}\,(A_0)_{13}\,(A_0)_{20}\,(A_0)_{21}\,(A_0)_{22}\,(A_0)_{23} \\ (A_1)_{00}\,(A_1)_{01}\,(A_1)_{02}(A_1)_{03}(A_1)_{10}\,(A_1)_{11} \\ (A_1)_{12}\,(A_1)_{13}\,(A_1)_{20}\,(A_1)_{21}\,(A_1)_{22}\dots)^t \end{aligned} \tag{34}$$

$$\begin{aligned} Q = ((Q_0)_{00}\,(Q_0)_{01}\,(Q_0)_{02}\,(Q_0)_{03}\,(Q_0)_{10}\,(Q_0)_{11} \\ (Q_0)_{12}\,(Q_0)_{13}\,(Q_0)_{20}\,(Q_0)_{21}\,(Q_0)_{22}\,(Q_0)_{23} \\ (Q_1)_{00}\,(Q_1)_{01}\,(Q_1)_{02}(Q_1)_{03}(Q_1)_{10}\,(Q_1)_{11} \\ (Q_1)_{12}\,(Q_1)_{13}\,(Q_1)_{20}\,(Q_1)_{21}\,(Q_1)_{22}\dots)^t \end{aligned} \tag{35}$$

Note that the matrix of system (32) is symmetric, consequently, the corresponding numerical matrix M is also symmetric. The matrix M is a block matrix, each block contains 12 rows of length $12s_p$. Let us describe a block of the matrix M. Let us introduce the following notations:

$$k = 0, \ldots, s_p - 1, R = 12k + a, a = 0, \ldots, 11, C = 12k + b, b = 0, \ldots, 11 \tag{36}$$

where R is the row index and C is the column index. The elements of matrix M corresponding to indices R and C can be written as follows.

$$\begin{array}{l}
R = 12k + 0\text{:} \quad C = 12k + 0,\ C = 12k + 1,\ C = 12k + 2,\ C = 12k + 3 \\
\quad C = 12n_k(0) + 0,\ \ldots,\ C = 12n_k(e(n_k) - 1) + 0 \mid (Q_k)_{00} \\
\quad (P'_k)_{00}\ (P'_k)_{10}\ (P'_k)_{20}\ (P'_k)_{30} \\
\quad (-B_{k,L(k)_0})_{00}\ \ldots\ (-B_{k,L(k)_{N(k)-1}})_{00} \\
R = 12k + 1\text{:} \quad C = 12k + 0,\ C = 12k + 1,\ C = 12k + 2,\ C = 12k + 3 \\
\quad C = 12n_k(0) + 1,\ \ldots,\ C = 12n_k(e(n_k) - 1) + 1 \mid (Q_k)_{01} \\
\quad (P'_k)_{01}\ (P'_k)_{11}\ (P'_k)_{21}\ (P'_k)_{31} \\
\quad (-B_{k,L(k)_0})_{01}\ \ldots\ (-B_{k,L(k)_{N(k)-1}})_{01} \\
R = 12k + 2\text{:} \quad C = 12k + 0,\ C = 12k + 1,\ C = 12k + 2,\ C = 12k + 3 \\
\quad C = 12n_k(0) + 2,\ \ldots,\ C = 12n_k(e(n_k) - 1) + 2 \mid (Q_k)_{02} \\
\quad (P'_k)_{02}\ (P'_k)_{12}\ (P'_k)_{22}\ (P'_k)_{32} \\
\quad (-B_{k,L(k)_0})_{02}\ \ldots\ (-B_{k,L(k)_{N(k)-1}})_{02} \\
R = 12k + 3\text{:} \quad C = 12k + 0,\ C = 12k + 1,\ C = 12k + 2,\ C = 12k + 3 \\
\quad C = 12n_k(0) + 3,\ \ldots,\ C = 12n_k(e(n_k) - 1) + 3 \mid (Q_k)_{03} \\
\quad (P'_k)_{03}\ (P'_k)_{13}\ (P'_k)_{23}\ (P'_k)_{33} \\
\quad (-B_{k,L(k)_0})_{03}\ \ldots\ (-B_{k,L(k)_{N(k)-1}})_{03} \\
R = 12k + 4\text{:} \quad C = 12k + 4,\ C = 12k + 5,\ C = 12k + 6,\ C = 12k + 7 \\
\quad C = 12n_k(0) + 4,\ \ldots,\ C = 12n_k(e(n_k) - 1) + 4 \mid (Q_k)_{10} \\
\quad (P'_k)_{00}\ (P'_k)_{10}\ (P'_k)_{20}\ (P'_k)_{30} \\
\quad (-B_{k,L(k)_0})_{10}\ \ldots\ (-B_{k,L(k)_{N(k)-1}})_{10} \\
R = 12k + 5\text{:} \quad C = 12k + 4,\ C = 12k + 5,\ C = 12k + 6,\ C = 12k + 7 \\
\quad C = 12n_k(0) + 5,\ \ldots,\ C = 12n_k(e(n_k) - 1) + 5 \mid (Q_k)_{11} \\
\quad (P'_k)_{01}\ (P'_k)_{11}\ (P'_k)_{21}\ (P'_k)_{31} \\
\quad (-B_{k,L(k)_0})_{11}\ \ldots\ (-B_{k,L(k)_{N(k)-1}})_{11}
\end{array}$$

$R = 12k + 6$: $C = 12k + 4,\ C = 12k + 5,\ C = 12k + 6,\ C = 12k + 7$
$C = 12n_k(0) + 6,\ \ldots,\ C = 12n_k(e(n_k) - 1) + 6 \mid (Q_k)_{12}$
$(P'_k)_{02}\ (P'_k)_{12}\ (P'_k)_{22}\ (P'_k)_{32}$
$(-B_{k,L(k)_0})_{12}\ \ldots\ (-B_{k,L(k)_{N(k)-1}})_{12}$
$R = 12k + 7$: $C = 12k + 4,\ C = 12k + 5,\ C = 12k + 6,\ C = 12k + 7$
$C = 12n_k(0) + 7,\ \ldots,\ C = 12n_k(e(n_k) - 1) + 7 \mid (Q_k)_{13}$
$(P'_k)_{03}\ (P'_k)_{13}\ (P'_k)_{23}\ (P'_k)_{33}$
$(-B_{k,L(k)_0})_{13}\ \ldots\ (-B_{k,L(k)_{N(k)-1}})_{13}$
$R = 12k + 8$: $C = 12k + 8,\ C = 12k + 9,\ C = 12k + 10,\ C = 12k + 11$
$C = 12n_k(0) + 8,\ \ldots,\ C = 12n_k(e(n_k) - 1) + 8 \mid (Q_k)_{20}$
$(P'_k)_{00}\ (P'_k)_{10}\ (P'_k)_{20}\ (P'_k)_{30}$
$(-B_{k,L(k)_0})_{20}\ \ldots\ (-B_{k,L(k)_{N(k)-1}})_{20}$
$R = 12k + 9$: $C = 12k + 8,\ C = 12k + 9,\ C = 12k + 10,\ C = 12k + 11$
$C = 12n_k(0) + 9,\ \ldots,\ C = 12n_k(e(n_k) - 1) + 9 \mid (Q_k)_{21}$
$(P'_k)_{01}\ (P'_k)_{11}\ (P'_k)_{21}\ (P'_k)_{31}$
$(-B_{k,L(k)_0})_{21}\ \ldots\ (-B_{k,L(k)_{N(k)-1}})_{21}$
$R = 12k + 10$: $C = 12k + 8,\ C = 12k + 9,\ C = 12k + 10,\ C = 12k + 11$
$C = 12n_k(0) + 10,\ \ldots,\ C = 12n_k(e(n_k) - 1) + 10 \mid (Q_k)_{22}$
$(P'_k)_{02}\ (P'_k)_{12}\ (P'_k)_{22}\ (P'_k)_{32}$
$(-B_{k,L(k)_0})_{22}\ \ldots\ (-B_{k,L(k)_{N(k)-1}})_{22}$
$R = 12k + 11$: $C = 12k + 8,\ C = 12k + 9,\ C = 12k + 10,\ C = 12k + 11$
$C = 12n_k(0) + 11,\ \ldots,\ C = 12n_k(e(n_k) - 1) + 11 \mid (Q_k)_{23}$
$(P'_k)_{03}\ (P'_k)_{13}\ (P'_k)_{23}\ (P'_k)_{33}$
$(-B_{k,L(k)_0})_{23}\ \ldots\ (-B_{k,L(k)_{N(k)-1}})_{23}$

Note that the quantities $(Q_k)_{ij}$ represent the corresponding values on the right-hand side for a given row of matrix M in equation (33) and are provided as a comment. The values $\left(B_{k,L(k)_n}\right)_{ij}$ are non-zero only if $i = j$.

Consider the block of matrix M containing the rows with indices $12k+0$, $12k+1$, $12k+2$ and $12k+3$. The matrix P_k (and $\widehat{P_k}$) has rank 1, but rows in the block are linear independent since diagonal elements has the form: $(P'_k)_{00} = \left(\widehat{P_k}\right)_{00} + \widehat{\beta_k}$, $(P'_k)_{11} = \left(\widehat{P_k}\right)_{11} + \widehat{\beta_k}$, $(P'_k)_{22} = \left(\widehat{P_k}\right)_{22} + \widehat{\beta_k}$ and $(P'_k)_{33} = \left(\widehat{P_k}\right)_{33} + \widehat{\beta_k}$. The matrix P (as well as $\widehat{P_k}$) has rank 1, however, the rows within the block are linearly independent, since the diagonal elements are given by: $(P'_k)_{00} = \left(\widehat{P_k}\right)_{00} + \widehat{\beta_k}$, $(P'_k)_{11} = \left(\widehat{P_k}\right)_{11} + \widehat{\beta_k}$, $(P'_k)_{22} = \left(\widehat{P_k}\right)_{22} + \widehat{\beta_k}$ and $(P'_k)_{33} = \left(\widehat{P_k}\right)_{33} + \widehat{\beta_k}$. The block-diagonal structure of the matrix M leads to the non-singularity of the matrix M.

The matrix M is positive-definite. Let v be vector-column of size $12s_p$

$$v = (v_0 v_1 v_2 \dots v_{12s_p-1})^t \tag{37}$$

Consider the expression $v^t M v$. For each block containing four rows, a similar condition holds

$$\begin{gathered}
(P_0)_0(P_0)_0 v_0 v_0 + (P_0)_1(P_0)_0 v_0 v_1 + (P_0)_2(P_0)_0 v_0 v_2 + (P_0)_0 v_0 v_3 \\
+(P_0)_0(P_0)_1 v_0 v_1 + (P_0)_1(P_0)_1 v_1 v_1 + (P_0)_2(P_0)_1 v_1 v_2 + (P_0)_1 v_1 v_3 \\
+(P_0)_0(P_0)_2 v_0 v_2 + (P_0)_1(P_0)_2 v_1 v_2 + (P_0)_2(P_0)_2 v_2 v_2 + (P_0)_2 v_2 v_3 \\
+(P_0)_0 v_0 v_3 + (P_0)_1 v_1 v_3 + (P_0)_2 v_2 v_3 + 1 \cdot v_3 v_3 = \\
= (v_0 v_1 v_2 v_3) \begin{pmatrix} (P_0)_0(P_0)_0 & (P_0)_0(P_0)_1 & (P_0)_0(P_0)_2 & (P_0)_0 \\ (P_0)_1(P_0)_0 & (P_0)_1(P_0)_1 & (P_0)_1(P_0)_2 & (P_0)_1 \\ (P_0)_2(P_0)_0 & (P_0)_2(P_0)_1 & (P_0)_2(P_0)_2 & (P_0)_2 \\ (P_0)_0 & (P_0)_1 & (P_0)_2 & 1 \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix} = \\
= v^t P_0 {P_0}^t v = (P_0{}^t v)^t P_0{}^t v = (P_0{}^t v)^2 \geq 0
\end{gathered} \tag{38}$$

Since α_0 is positive, $\widehat{\beta_0}$ is positive and terms $\widehat{\beta_0} {v_0}^2$, $\widehat{\beta_0} {v_1}^2$, $\widehat{\beta_0} {v_2}^2$, $\widehat{\beta_0} {v_3}^2$ are non-negative, therefore, the contribution of the block under consideration to the total sum $v^t M v$ is non-negative. Thus, the contribution of each block is non-negative, and the total contribution of these blocks is positive (for actual experimental input data).

Note that

$$\begin{aligned}
\widehat{\beta_0} {v_0}^2 &= \left(\sum_{j=0}^{N(i)-1} \beta_{0,L(0)_j} \right) {v_0}^2 = \beta_{0,L(0)_0} {v_0}^2 + \left(\sum_{j=1}^{N(i)-1} \beta_{0,L(0)_j} \right) {v_0}^2 \\
&= \left(B_{0,L(0)_0}\right)_{00} {v_0}^2 + \left(\sum_{j=1}^{N(i)-1} \beta_{0,L(0)_j} \right) {v_0}^2
\end{aligned} \tag{39}$$

where $\left(\sum_{j=1}^{N(i)-1} \beta_{0,L(0)_j} \right) {v_0}^2 \geq 0$. The coefficient $\left(B_{0,L(0)_0}\right)_{00}$ is included in the total sum $v^t M v$ four times and forms the expression

$$\begin{aligned}
&\left(B_{0,L(0)_0}\right)_{00} {v_0}^2 - 2 \left(B_{0,L(0)_0}\right)_{00} v_0 v_{L(0)_0} + \left(B_{0,L(0)_0}\right)_{00} {v_{L(0)_0}}^2 \\
&= \left(B_{0,L(0)_0}\right)_{00} \left(v_0 - v_{L(0)_0}\right)^2 \geq 0
\end{aligned} \tag{40}$$

This condition is satisfied for all coefficients $\left(B_{k,L(k)_n}\right)_{ii}$ in the matrix M, therefore,

$$v^t M v > 0 \tag{41}$$

for all vectors $v \neq 0$ which means the matrix M is positive definite. In such a way, the matrix M is non-singular, symmetric, and positive-definite. To solve the system of linear equations (33) numerically, we use the conjugate gradient method.

4 Algorithm and Parameter Selection

We use in the proposed paper the dataset Argoverse 2 [22]. In the context of scene flow estimation, points of each cloud in Argoverse 2 can be categorized according to their spatial location and motion characteristics by the following way.

1. Background points belong to the static environment, such as roads, buildings, sidewalks, and other stationary infrastructure. Their apparent motion in the sensor coordinate system is primarily caused by the movement of the observing vehicle. After compensating for the ego-motion of the vehicle, these points are expected to have approximately zero displacement in the world coordinate system.
2. Foreground points belong to objects or regions of interest located in the scene, typically including vehicles, pedestrians, and cyclists. This category describes the spatial or semantic role of points rather than their motion: foreground points may belong to either moving or stationary objects.
3. Dynamic points belong to objects that move independently of the observing vehicle, such as moving vehicles, pedestrians, and cyclists. Their motion cannot be explained solely by the ego-motion of the sensor and must be captured by the estimated scene flow vectors.

The distinction between foreground/background and dynamic/static is important. The evaluation uses three disjoint categories: Foreground Dynamic (FD), Foreground Static (FS), and Background Static (BS). A point is classified as dynamic when its ground-truth flow magnitude, after ego-motion compensation, exceeds the threshold of (0.05) m per 0.1-second interval, corresponding to (0.5) m/s. Points belonging to objects visible in only one of the two sweeps are excluded from flow evaluation because their ground-truth flow cannot be reliably determined.

Suppose that P and Q are two consecutive point clouds in the dataset. We use FD and FS points in the clouds P and Q. The proposed functional (1) is used in the ICP algorithm. The first step of the algorithm is to compute the correspondence between point clouds P and Q. To establish a correspondence, we use a subsets of points FD and FS in P and Q. The second step is the variational problem of the ICP algorithm. The variational problem (6) is solved for a fixed correspondence. The estimated transformations are applied to the points of point cloud P, after which the two steps are repeated iteratively 5 times.

Remark. The proposed method is not an end-to-end algorithm, as it utilizes information about FD and FS points contained in the dataset. The proposed algorithm, based on functional (1), can obtain information about points FD and FS from another algorithm (a neural-network-based one).

Correspondence is determined using the nearest-point method, which is standard for ICP. The weighting coefficients α_i in (2) are determined by the following way. If distance between corresponding points p_i and q_i is equal to d, then

$$\alpha_i = e^{-d^2} \tag{42}$$

The weighting coefficients $\beta_{i,L(i)_j}$ determine the stiffness of the (virtual) edges connecting points p_i and $p_{L(i)_j}$. A higher value of the parameter $\beta_{i,L(i)_j}$ corresponds to a less deformable edge, while a lower value corresponds to a more deformable edge. We use higher values for the parameters $\beta_{i,L(i)_j}$ for edges connecting two points within FD, and lower values for edges connecting a point in FD to a point in FS.

The quality of scene flow estimation is evaluated using the EPE3D metric. Let $(\Delta_k)_{true}$ be a true displacement vector for the point p_k. Let the estimated displacement vector be defined as

Table 1 Accuracy of the proposed algorithm VF for different values of the parameters β.

β within FD	β between FD and FS	EPE3D
10	10	0.62935
100	100	0.57816
100	1	0.57081

$$(\Delta_k)_{est} = (A_k)_* p_k - p_k \tag{43}$$

Then the EPE3D metric for two sequential point clouds is defined by the following expression

$$EPE3D(P,Q) = \frac{1}{s_p} \sum_{k=0}^{s_p-1} \parallel (\Delta_k)_{true} - (\Delta_k)_{est} \parallel_{L_2} \tag{44}$$

For the EPE3D calculation, we use only the FD points, and s_p here is equal to the number of points in the FD.

5 Computer Simulation

The Argoverse 2 dataset contains sequential point clouds with points labelled as types BS, FS, and FD. Note that the existence of a true vector field relating the first point cloud to the second, as well as the second point cloud to the third, makes it possible to determine the true vector field relating the first point cloud to the third. This capability allows to evaluate scene flow between point clouds with different steps within a sequence provided in a dataset. This capability allows for the estimation of scene flow between point clouds at different intervals within a sequence provided in a dataset.

The DeFlow neural network is one of the best-known algorithms for 3D scene flow estimation [23]. The neural network designed for 3D scene flow estimation from consecutive LiDAR point clouds, with particular emphasis on autonomous driving applications. The method was evaluated on the Argoverse 2 scene flow benchmark. Here, we compare the proposed scene flow estimation method VF with DeFlow (note that the proposed algorithm made partial use of ground-truth data).

5.1 Selection of the Values of β

Table 1 shows accuracy of the proposed algorithm VF for different values of the parameters β. Mean EPE3D values are presented for a number of point cloud pairs. The column "β inside of FD" shows the values of all parameters β corresponding to (virtual) edges connected FD points, the column "β between FD and FS" shows the values of parameters β corresponding to (virtual) edges connected FD and FS points.

Increasing the stiffness of the edges improves the mean EPE3D value. Distinguishing between the stiffness of the edges within the FD and the edges connecting the FD and FS also improves the mean EPE3D value.

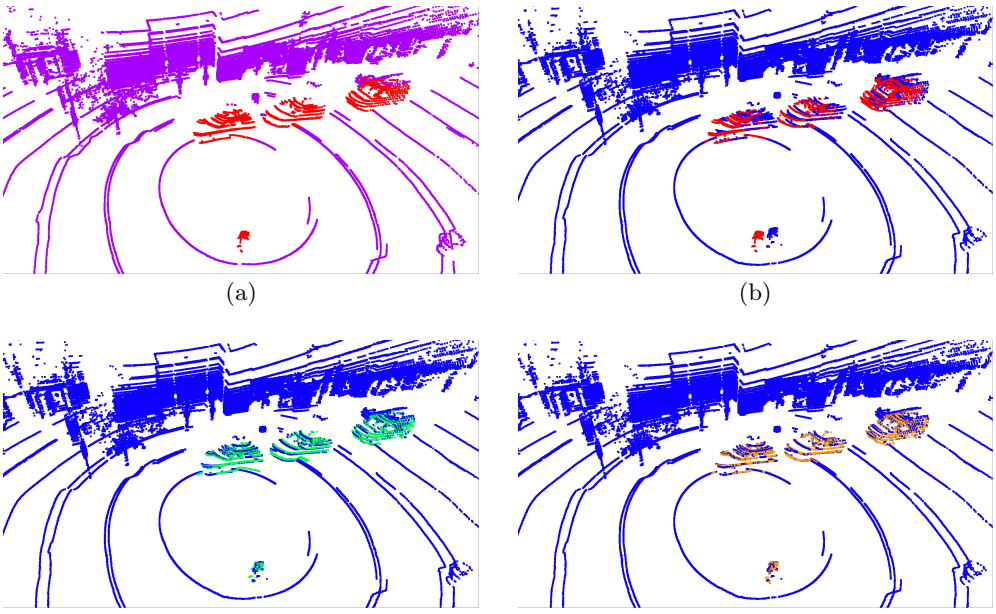


Fig. 1 (a) Purple - source point cloud, red - foreground; (b) blue - target point cloud; (c) green - result of applying DeFlow, blue - target point cloud; (d) orange - result of applying VF, blue - target point cloud.

Table 2 Accuracy of the proposed algorithm VF and DeFlow.

	VF	DeFlow
EPE3D	0.02868	0.02783

5.2 Examples with Step Equal to 1

Here, we consider scene flow estimation for pairs of consecutive point clouds whose indices differ by 1. Fig. 1 (a) shows the source point cloud (purple – background, red – foreground (FD+FS)). Fig. 1 (b) shows target point cloud (blue). The red points are source foreground, positioned within the target's local coordinate system. Fig. 1 (c) shows the result of applying DeFlow to foreground of the source point cloud (green). The target point cloud is indicated in blue. Fig. 1 (d) shows the result of applying VF to foreground of the source point cloud (orange). The target point cloud is indicated in blue.

Table 2 shows accuracy of VF and DeFlow algorithms.

Fig. 2 (a) shows the source point cloud (purple – background, red – foreground (FD+FS)). Fig. 2 (b) shows target point cloud (blue). The red points are source foreground, positioned within the target's local coordinate system. Fig. 2 (c) shows

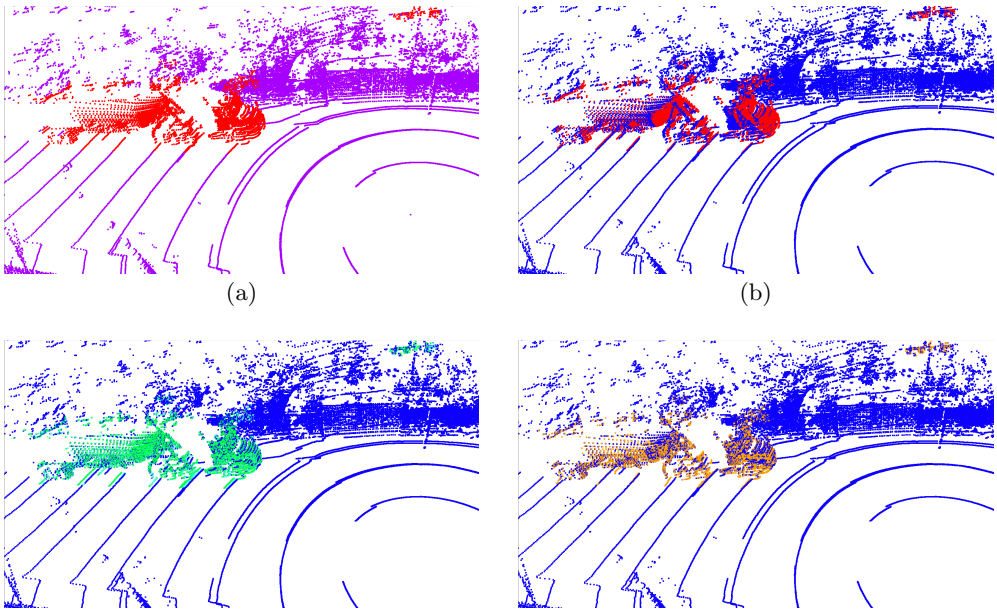


Fig. 2 (a) Purple - source point cloud, red - foreground; (b) blue - target point cloud; (c) green - result of applying DeFlow, blue - target point cloud; (d) orange - result of applying VF, blue - target point cloud.

Table 3 Accuracy of the proposed algorithm VF and DeFlow.

	VF	DeFlow
EPE3D	0.04316	0.03891

the result of applying DeFlow to foreground of the source point cloud (green). The target point cloud is indicated in blue. Fig. 2 (d) shows the result of applying VF to foreground of the source point cloud (orange). The target point cloud is indicated in blue.

Table 3 shows accuracy of VF and DeFlow algorithms.

5.3 Example with Step Equal to 3

Here, we consider scene flow estimation for pair of consecutive point clouds whose indices differ by 3. Fig. 3 (a) shows the source point cloud (purple – background, red – foreground (FD+FS)). Fig. 3 (b) shows target point cloud (blue). The red points are source foreground, positioned within the target's local coordinate system. Fig 3. (c) shows the result of applying DeFlow to foreground of the source point cloud (green). The target point cloud is indicated in blue. Fig. 3 (d) shows the result of applying VF

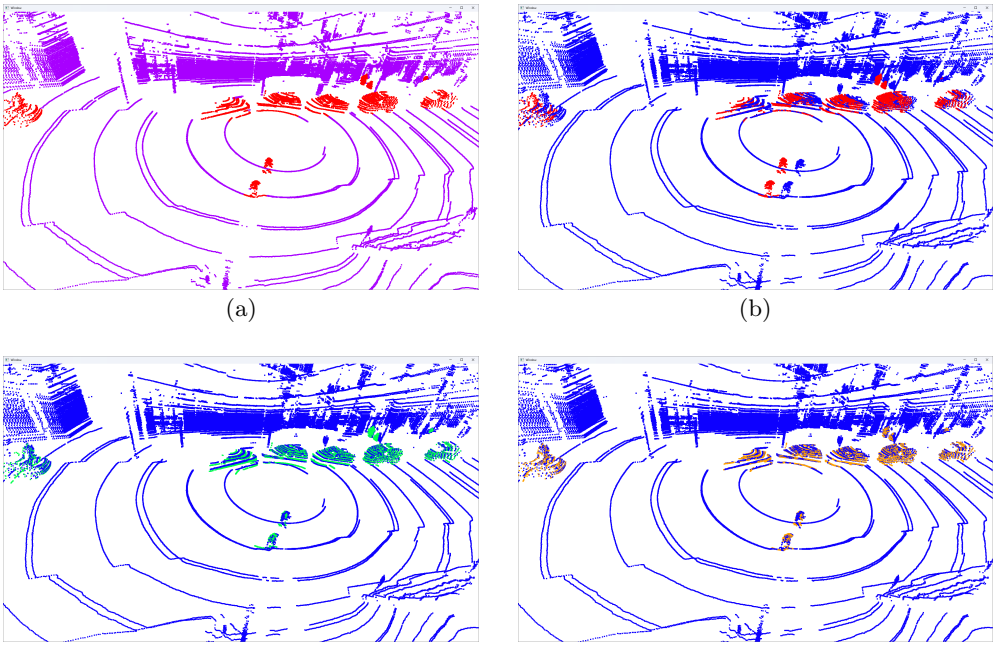


Fig. 3 (a) Purple - source point cloud, red - foreground; (b) blue - target point cloud; (c) green - result of applying DeFlow, blue - target point cloud; (d) orange - result of applying VF, blue - target point cloud.

Table 4 Accuracy of the proposed algorithm VF and DeFlow.

	VF	DeFlow
EPE3D	0.07262	0.13898

to foreground of the source point cloud (orange). The target point cloud is indicated in blue.

Table 4 shows accuracy of VF and DeFlow algorithms.

5.4 Examples with Step Equal to 4

Here, we consider scene flow estimation for pairs of consecutive point clouds whose indices differ by 4. Fig. 4(a) shows the source point cloud (purple – background, red – foreground (FD+FS)). Fig. 4 (b) shows target point cloud (blue). The red points are source foreground, positioned within the target's local coordinate system. Fig. 4 (c) shows the result of applying DeFlow to foreground of the source point cloud (green). The target point cloud is indicated in blue. Fig. 4 (d) shows the result of applying VF

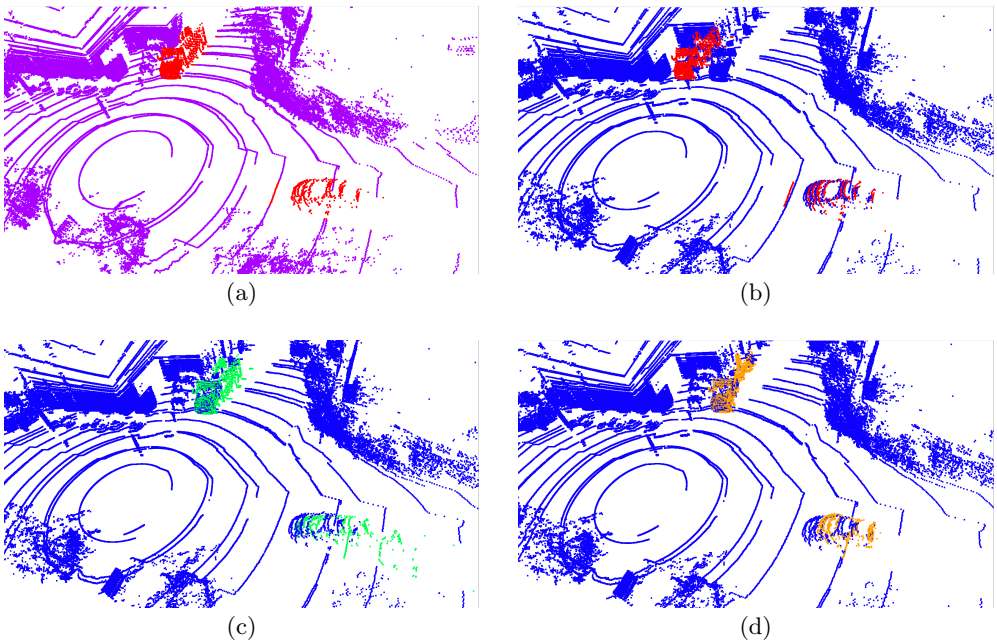


Fig. 4 (a) Purple - source point cloud, red - foreground; (b) blue - target point cloud; (c) green - result of applying DeFlow, blue - target point cloud; (d) orange - result of applying VF, blue - target point cloud.

Table 5 Accuracy of the proposed algorithm VF and DeFlow.

	VF	DeFlow
EPE3D	0.25791	0.26038

to foreground of the source point cloud (orange). The target point cloud is indicated in blue.

Table 5 shows accuracy of VF and DeFlow algorithms.

Fig. 5 (a) shows the source point cloud (purple – background, red – foreground (FD+FS)). Fig. 5 (b) shows target point cloud (blue). The red points are source foreground, positioned within the target's local coordinate system. Fig. 5 (c) shows the result of applying DeFlow to foreground of the source point cloud (green). The target point cloud is indicated in blue. Fig. 5 (d) shows the result of applying VF to foreground of the source point cloud (orange). The target point cloud is indicated in blue.

Table 6 shows accuracy of VF and DeFlow algorithms.

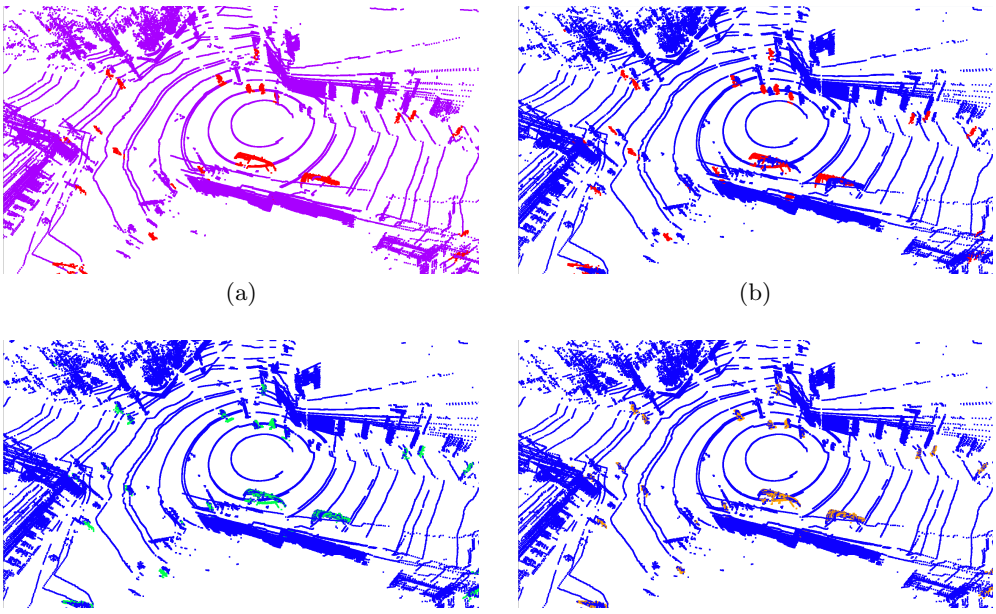


Fig. 5 (a) Purple - source point cloud, red - foreground; (b) blue - target point cloud; (c) green - result of applying DeFlow, blue - target point cloud; (d) orange - result of applying VF, blue - target point cloud.

Table 6 Accuracy of the proposed algorithm VF and DeFlow.

	VF	DeFlow
EPE3D	0.08580	0.13274

5.5 Examples with Step Equal to 5

Here, we consider scene flow estimation for pairs of consecutive point clouds whose indices differ by 5. Fig. 6 (a) shows the source point cloud (purple – background, red – foreground (FD+FS)). Fig. 6 (b) shows target point cloud (blue). The red points are source foreground, positioned within the target's local coordinate system. Fig. 6 (c) shows the result of applying DeFlow to foreground of the source point cloud (green). The target point cloud is indicated in blue. Fig. 6 (d) shows the result of applying VF to foreground of the source point cloud (orange). The target point cloud is indicated in blue.

Table 7 shows accuracy of VF and DeFlow algorithms.

Fig. 7 (a) shows the source point cloud (purple – background, red – foreground (FD+FS)). Fig. 7 (b) shows target point cloud (blue). The red points are source foreground, positioned within the target's local coordinate system. Fig. 7 (c) shows

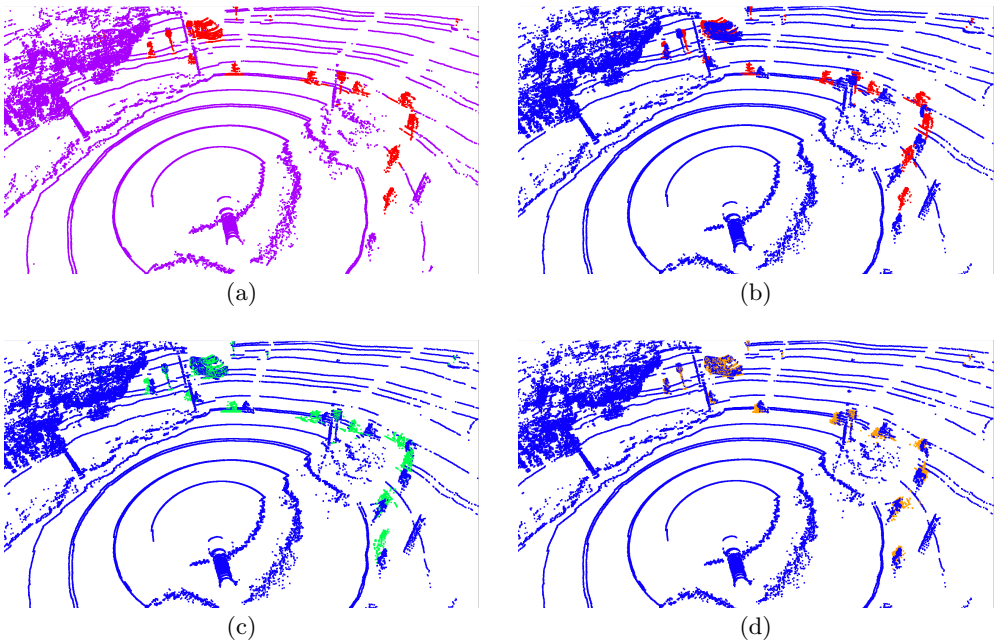


Fig. 6 (a) Purple - source point cloud, red - foreground; (b) blue - target point cloud; (c) green - result of applying DeFlow, blue - target point cloud; (d) orange - result of applying VF, blue - target point cloud.

Table 7 Accuracy of the proposed algorithm VF and DeFlow.

	VF	DeFlow
EPE3D	0.46556	0.72371

Table 8 Accuracy of the proposed algorithm VF and DeFlow.

	VF	DeFlow
EPE3D	1.10391	2.16174

the result of applying DeFlow to foreground of the source point cloud (green). The target point cloud is indicated in blue. Fig. 7 (d) shows the result of applying VF to foreground of the source point cloud (orange). The target point cloud is indicated in blue.

Table 8 shows accuracy of VF and DeFlow algorithms.

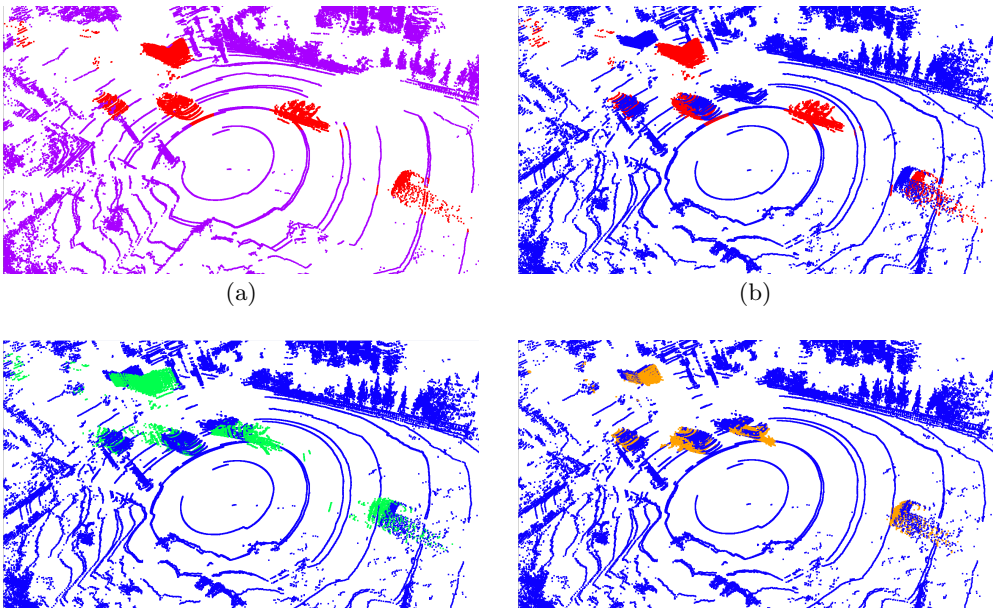


Fig. 7 (a) Purple - source point cloud, red - foreground; (b) blue - target point cloud; (c) green - result of applying DeFlow, blue - target point cloud; (d) orange - result of applying VF, blue - target point cloud.

Table 9 Accuracy of the proposed algorithm VF and DeFlow.

	VF	DeFlow
EPE3D	0.14214	0.14744

Fig. 8 (a) shows the source point cloud (purple – background, red – foreground (FD+FS)). Fig. 8 (b) shows target point cloud (blue). The red points are source foreground, positioned within the target's local coordinate system. Fig. 8 (c) shows the result of applying DeFlow to foreground of the source point cloud (green). The target point cloud is indicated in blue. Fig. 8 (d) shows the result of applying VF to foreground of the source point cloud (orange). The target point cloud is indicated in blue.

Table 9 shows accuracy of VF and DeFlow algorithms.

6 Conclusion

In this paper, we apply the 3D scene flow algorithm for real world data. The algorithm is based on a variational functional with parameters that allow for adjusting the stiffness of a source point cloud as it is being aligned with a target point cloud. The

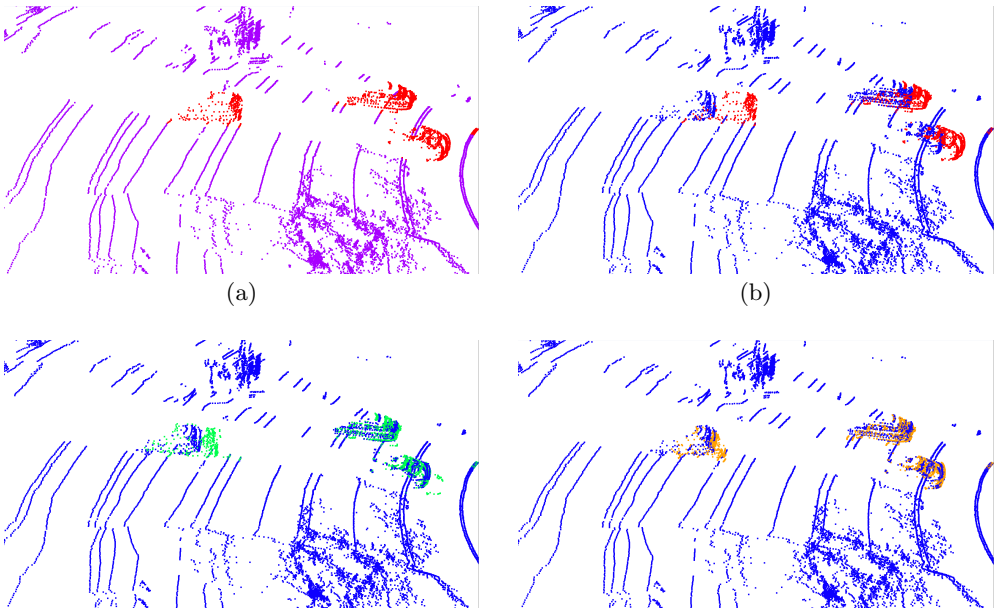


Fig. 8 (a) Purple - source point cloud, red - foreground; (b) blue - target point cloud; (c) green - result of applying DeFlow, blue - target point cloud; (d) orange - result of applying VF, blue - target point cloud.

proposed algorithm is compared with the efficient neural network algorithm DeFlow. Experiments show that the proposed algorithm yields results comparable to those of DeFlow when dealing with point clouds that exhibit minor differences (within the context of the Argoverse 2 dataset), and outperforms it in cases where the differences between the point clouds are more significant.

Acknowledgements. The work was supported by Russian Science Foundation grant 25-11-20031.

Author Contributions. A.M., S.V. and V.K. wrote the main manuscript text and A.V. prepared computer simulation. All authors reviewed the manuscript

Data availability. No datasets were generated during the current study. The Argoverse 2 dataset was analysed during the current study.

Competing interests. The authors declare no competing interests.

References

1. Besl, P. J., McKay, N. D.: A Method for Registration of 3-D Shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**, 239–256 (1992)
2. Chen, Y., Medioni, G.: Object Modelling by Registration of Multiple Range Images. Image Vis. Comput. **10**, 145–155 (1992)

3. Segal, A., Haehnel, D., Thrun, S.: Generalized-ICP. Proc. Robotics: Science and Systems V, pp. 161–168 (2009)
4. Serafin, J., Grisetti, G.: Using Extended Measurements and Scene Merging for Efficient and Robust Point Cloud Registration. Robot. Auton. Syst. **92**, 91–106 (2017)
5. Makovetskii, A., Voronin, S., Kober, V., Tihonkih, D.: Affine Registration of Point Clouds Based on Point-to-Plane Approach. Procedia Eng. **201**, 322–330 (2017)
6. Horn, B. K. P.: Closed-Form Solution of Absolute Orientation Using Unit Quaternions. J. Opt. Soc. Am. A **4**, 629–642 (1987)
7. Horn, B. K. P., Hilden, H. M., Negahdaripour, S.: Closed-Form Solution of Absolute Orientation Using Orthonormal Matrices. J. Opt. Soc. Am. A **5**, 1127–1135 (1988)
8. Umeyama, S.: Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. IEEE Trans. Pattern Anal. Mach. Intell. **13**, 376–380 (1991)
9. Mellado, N., Aiger, D., Mitra, N. J.: Super 4PCS Fast Global Pointcloud Registration via Smart Indexing. Comput. Graph. Forum **33**(5), 205–215 (2014)
10. Makovetskii, A., Voronin, S., Kober, V., Voronin, A.: Point Cloud Registration Based on Multiparameter Functional. Mathematics **9**, 2589 (2021)
11. Makovetskii, A., Voronin, S., Kober, V., Voronin, A.: Coarse Point Cloud Registration Based on Variational Functionals. Mathematics **11**, 35 (2023)
12. Makovetskii, A., Voronin, S., Kober, V., Voronin, A.: A Regularized Point Cloud Registration Approach for Orthogonal Transformations. J. Glob. Optim. **83**(3), 497–519 (2022)
13. Makovetskii, A., Voronin, S., Kober, V., Voronin, A.: A Non-Iterative Method for Approximation of the Exact Solution to the Point-to-Plane Variational Problem for Orthogonal Transformation. Math. Methods Appl. Sci. **41**, 9218–9230 (2018)
14. Makovetskii, A., Voronin, S., Kober, V., Voronin, A.: Registration Algorithm for Incongruent Point Clouds. Proc. VIII Int. Conf. Inf. Technol. Nanotechnol. (ITNT), pp. 1–8 (2022)
15. Makovetskii, A., Voronin, S., Kober, V., Voronin, A.: A Multiparameter Functional for Rotation and Translation Estimation in Point Cloud Registration Task. Comput. Math. Model. (2025)
16. Amberg, B., Romdhani, S., Vetter, T.: Optimal Step Nonrigid ICP Algorithms for Surface Registration. Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2007)
17. Makovetskii, A., Voronin, S., Kober, V., Voronin, A.: An Efficient Algorithm for Non-Rigid Object Registration. Comput. Opt. **44**(1), 67–73 (2020)
18. Voronin, S., Kober, V., Makovetskii, A., Voronin, A.: Non-Rigid ICP and 3D Facial Models. Proc. SPIE **11137**, 111372K (2019)
19. Liu, X., Qi, C. R., Guibas, L. J.: FlowNet3D: Learning Scene Flow in 3D Point Clouds. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 529–537 (2019)
20. Gu, X., Wang, Y., Wu, C., Lee, Y. J., Wang, P.: HPLFlowNet: Hierarchical Permutohedral Lattice FlowNet for Scene Flow Estimation on Large-Scale Point Clouds. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern

Recognition (CVPR), pp. 3254–3263 (2019)
21. Li, Z., Xiang, N., Chen, H., Zhang, J., Yang, X.: Deep Learning for Scene Flow Estimation on Point Clouds: A Survey and Prospective Trends. Comput. Graph. Forum **42**(6), e14795 (2023)
22. Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J. K., Ramanan, D., Carr, P., Hays, J.: Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting. Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks) (2021)
23. Zhang, Q., Yang, Y., Fang, H., Geng, R., Jensfelt, P.: DeFlow: Decoder of Scene Flow Network in Autonomous Driving. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 2105–2111 (2024)