



Article

Accurate Global Point Cloud Registration Using GPU-Based Parallel Angular Radon Spectrum

Ernesto Fontana ^{1,*}  and Dario Lodi Rizzini ^{1,2} 

¹ Department of Engineering and Architecture, University of Parma, Parco Area delle Scienze 181/A, 43124 Parma, Italy; dario.lodirizzini@unipr.it

² Interdepartmental Center for Energy and Environment (CIDEA), University of Parma, Parco Area delle Scienze 95, 43124 Parma, Italy

* Correspondence: ernesto.fontana@unipr.it

Abstract: Accurate robot localization and mapping can be improved through the adoption of globally optimal registration methods, like the Angular Radon Spectrum (ARS). In this paper, we present Cud-ARS, an efficient variant of the ARS algorithm for 2D registration designed for parallel execution of the most computationally expensive steps on Nvidia™ Graphics Processing Units (GPUs). Cud-ARS is able to compute the ARS in parallel blocks, with each associated to a subset of input points. We also propose a global branch-and-bound method for translation estimation. This novel parallel algorithm has been tested on multiple datasets. The proposed method is able to speed up the execution time by two orders of magnitude while obtaining more accurate results in rotation estimation than state-of-the-art correspondence-based algorithms. Our experiments also assess the potential of this novel approach in mapping applications, showing the contribution of GPU programming to efficient solutions of robotic tasks.

Keywords: registration; mapping; parallel processing; GPU



Citation: Fontana, E.; Lodi Rizzini, D. Accurate Global Point Cloud Registration Using GPU-Based Parallel Angular Radon Spectrum. *Sensors* **2023**, *23*, 8628. <https://doi.org/10.3390/s23208628>

Academic Editor: Frantisek Duchon

Received: 18 September 2023

Revised: 16 October 2023

Accepted: 17 October 2023

Published: 22 October 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In recent years, robot localization and mapping research has been focused on globally optimal registration of point clouds. Registration is the problem of finding the best rigid transformation that links multiple overlapping measurements acquired from different viewpoints. This primitive operation is essential in robot localization, motion tracking systems, and shape reconstruction from partial point clouds. Standard registration methods such as iterative closest point (ICP) [1] are often referred to as local algorithms, as they rely on an accurate initial guess, e.g., provided by robot odometry, to find the transformation that locally minimizes their objective function. Local alignment is usually achieved through associations between corresponding points in the input point clouds. Point matching is feasible when point clouds are close, through a raw assessment of their relative transformation. When a reliable initial estimation is not available, local algorithms may fail to compute consistent and accurate solutions.

Global registration methods [2–5] compute the aligning pose corresponding to the global minimum of the objective function. These algorithms are often referred to as *certifiable*, as they do not depend (or are less dependent on) an initial guess of the rigid transformation among the input point clouds, and are robust to a large amount of outliers. Global registration usually relies either on robust outlier rejection algorithms for detection of globally consistent correspondences, or on effective global descriptors of point clouds. In order to guarantee global optimality, global registration generally requires computationally intensive operations.

The last decade has been characterized by the rapid and continuous development of graphic cards in terms of performance and application domains. As a matter of fact, the most advanced graphic cards today can be considered additional processing units; they are

commonly called Graphic Processing Units (GPUs). As such, global registration methods that are composed of highly parallelizable operations can greatly benefit from GPU processing. In particular, correspondence-less registration methods like Angular Radon Spectrum (ARS) [2,3] separately operate on each point cloud. ARS is a descriptor that captures collinearity among a set of planar points, possibly capturing the simplest and strongest invariant property to rigid motion. The rotation between two point clouds is accurately estimated by finding the maximum correlation between their corresponding spectra. The main limitation of ARS lies in its quadratic complexity, and using GPUs could significantly speed up their computations operating on independent parts. Mathematical frameworks built to deal with similar problems using GPUs have already been proposed [6–8], but they tend to lack in guidance for implementation.

In this paper, we propose *Cud-ARS*, a novel parallel algorithm for the registration of planar point clouds using ARS, implemented for execution on Nvidia GPUs through CUDA. ARS descriptors are represented by Fourier coefficients and depended on each point pair. The pairwise assessment of coefficients has been decomposed into independent tasks and assigned to different GPU cores. More specifically, the grid-like structure originates from splitting data into blocks. Computations are then performed among in-block point pairs by parallel threads, then by inter-block comparisons, and finally by a follow-up summation of partial results. In order to limit the computational load on each CUDA thread while also coping with the limited memory capabilities on each said thread, a matrix-like structure, adaptive to the size of the problem to be dealt with, has been implemented and is discussed in this paper. Our self-contained implementation also limits unnecessary dependencies and improves reusability for other projects and frameworks. *Cud-ARS* has been integrated into a full registration pipeline including translation estimation to perform pairwise scan alignment. Our experiments show a significant reduction in execution time guaranteed by GPU-based assessments of rotation. Moreover, the mapping experiments on standard datasets show that the performance of *Cud-ARS* pairwise registration is comparable with tools performing state-of-the-art scan-to-map registration. In summary, the contributions of this paper are the following:

1. Introducing the parallel algorithm *Cud-ARS* for computation of the Fourier coefficients of ARS suitable for parallel execution of GPUs.
2. The implementation of *Cud-ARS* using the Nvidia CUDA library and conducting experiments comparing performance with state-of-the-art registration methods on benchmarks.
3. A branch-and-bound (B&B)-based translation estimation method improves accuracy over previous versions used in ARS, completing the pose estimation pipeline.

This paper is organized as follows: Section 2 presents the related literature. Section 3 illustrates ARS and its application to point cloud registration. Section 4 presents novel algorithmic contributions, and in Section 5, we discuss our experimental assessment. Finally, Section 6 presents our final remarks.

2. Related Work

The scientific literature on registration is extensive, covering several application domains [9] and including different formulations for a variety of problems. Despite it being a problem that has been investigated for over three decades, there is still room for better generalization and overall improvement in areas that are today considered state-of-the-art solutions [10]. A general classification criterion divides registration methods for point clouds into *correspondence-based* and *correspondence-less* methods. Correspondence-based methods rely on the estimation of corresponding points between two point clouds to be aligned. As point association is usually achieved through a rough initial assessment of the transformation between the point clouds, correspondence-based algorithms usually achieve locally optimal registration. Iterative Closest Point (ICP) [1,11] is perhaps the most popular registration algorithm that iteratively refines the transformation by matching each point of the source point set with its closest point in the target set. Notable variants use other

cost functions like point-to-plane and plane-to-plane distance [12]. Several approaches have been proposed to address uncertainty in correspondences. The Normal Distributions Transform (NDT) [13,14] performs a soft association based on a probability distribution instead of points. Stein-ICP [15] explicitly evaluates transform uncertainty using Stein variational gradient descent to achieve consistent associations. Recently, Kolpakov and Werman [16] propose an algorithm to assess the initial guess of ICP's explicit reasoning on point covariances.

Some registration techniques have been proposed to specifically fit widely used sensors like LIDARs. A recent work from Jaimez et al. [17] deals with odometry oriented methods for registration of planar LIDAR data using range flow constraint equations. LIDAR odometry and mapping (LOAM) [18] and its successive variants [19,20] estimate the transformation through detection and association of sparse features like edges and planar patches in point clouds. Customized systems like LIO [21] also integrate other sensory data, such as inertial measurements, in order to improve accuracy.

Correspondence-less and globally optimal methods are less common in the literature. Strong outlier rejection algorithms can make registration less dependent on initial guess or on correspondences. Coherence point drift (CPD) [22] and Vector Field Consensus (VFC) exploit the relative position of points belonging to the same cloud, with the goal of filtering outlier correspondences. TEASER [4] is able to perform registration through a truncated least squares formulation. It also presents sophisticated outlier rejection techniques for dealing with noisy input data. Although they are robust to a high percentage of outliers, these approaches rely on an initial set of correspondences; hence, they cannot be considered *fully* global methods. In contrast, GO-ICP [5] is the clearest example of globally optimal registration using the branch-and-bound method. Although it exploits approximations such as the Euclidean distance transform for closest point computation, this algorithm is computationally intensive and practically unsuitable for online estimation. Another way of addressing correspondence-less registration is by using strong global features in point clouds. The Hough Spectrum (HS) [23,24] and ARS [2,3] are collinearity descriptors for planar point set that can be used in rotation estimation, decoupling it from translation. ARS has the advantages to overcome the discretization issues of HS and to accurately evaluate rotation. Zhang et al. [25] have recently proposed a formulation based on uncertain landmark data. The common element between [25] and ARS stands in basing for the SLAM formulation on working with uncertain data, which should naturally help them in performing better on real sensory data. Other estimation frameworks that are based on the factor graph formulation [26] move in a similar direction. A recent example of this, encompassing an even broader span of applications, is WOLF by Solà et al. [27].

Even if the usage of SLAM methods in real-time applications has been a long-time concern [28,29], the exploitation of GPUs to speed up perception and sensor processing [30] is less frequent in the literature. It is instead more common to see it paired with computer vision primitives or straight-up deep learning methods [31–33]. Milioto et al. have proposed Bonnet [34] and RangeNet++ [35] for performing segmentation based on deep learning. Collet et al. [36,37] propose a series of works leading to MOPED, which is a framework for estimating the pose of objects based on recognizing feature keypoints. A typical application for some of these frameworks is robotic manipulation [38]. Furthermore, Titan [39] is a library comprising parallel algorithms to handle geometry in soft-body and multi-robot physics simulations. This approach to Nvidia CUDA parallel processing closely resembles the proposed operational decomposition of ARS. GPU-related literature also includes a class of works that focuses on high-level formal analysis of computational optimization and parallelization. Ha et al. [40] present an optimal parallel scan method, showing experiments on throughput and MIPS on a data-intensive simulation of a prefix sum problem. The goal of this paper is to help fill the gap between parallelization analysis of benchmark problems and deep learning-related applications, specifically in the field of robotic registration and mapping.

3. Angular Radon Spectrum for Registration

3.1. Angular Radon Spectrum of a Gaussian Mixture Model

ARS is a function suitable for the estimation of rotation between two point clouds. It has been introduced in [2,3]. Input point clouds must be represented as a Gaussian Mixture Model (GMM). Although other solutions are possible, a straightforward choice to convert a point cloud into a GMM is to associate each point to a Gaussian kernel centered on the point, with a covariance matrix representing the uncertainty about the point position. We let $\mathcal{P} = \{\boldsymbol{\mu}_i\}_{i=1,\dots,n_p}$ with $\boldsymbol{\mu}_i \in \mathbb{R}^2$ be the estimated position vectors of the points. It is convenient to define the *density function* $f : \mathbb{R}^2 \rightarrow \mathbb{R}_{\geq 0}$ that represents the point density in the plane and is proportional to the probability density function (PDF) of finding a point. Then, the GMM density function is defined as

$$f(\mathbf{r}) = \sum_{i=1}^{n_p} w_i f_i(\mathbf{r}) = \sum_{i=1}^{n_p} w_i n(\mathbf{r} - \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \quad (1)$$

where the sum of positive weights w_i is equal to one, and symbol $n(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ is used for a Gaussian kernel of mean value $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$. The *Radon Transform* (RT) [41] of $f(\mathbf{r})$ enables measuring the alignment of the point set with a given line $\mathcal{F}_{\mathbf{q}}$ represented by parameters $\mathbf{q} = [\theta, \rho]^\top$. The RT is defined as

$$\mathcal{R}[f](\mathbf{q}) = \int_{\mathcal{F}_{\mathbf{q}}} f(\mathbf{r}) d\mathbf{r} = \sum_{i=1}^{n_p} w_i \int_{\mathcal{F}_{\mathbf{q}}} n(\mathbf{r} - \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) d\mathbf{r}. \quad (2)$$

In our case, the integral of line $\mathcal{F}_{\mathbf{q}}$ can be solved through the parametric equation of points $\mathbf{r}(\mathbf{t})$ lying on line

$$\mathbf{r}(\mathbf{t}) = t_1 \mathbf{u}_1 + t_2 \mathbf{u}_2 = \mathbf{U}\mathbf{t}, \quad (3)$$

where $t_1 = \rho$ is a fixed constant, t_2 is the varying parameter associated to the points on the line, $\mathbf{u}_1 = \hat{\mathbf{u}}(\theta) = [\cos \theta, \sin \theta]^\top$ is the unitary vector orthogonal to the line, and $\mathbf{u}_2 = \hat{\mathbf{u}}(\theta + \pi/2) = [-\sin \theta, \cos \theta]^\top$ corresponds to the line direction. Since \mathbf{r} is a linear transformation, the integral of each Gaussian kernel has an elegant closed-form expression

$$\mathcal{R}[f_i](\theta, \rho) = n\left(\rho - \tilde{\mu}_{i,1}, \tilde{\sigma}_{i,1}^2\right), \quad (4)$$

where $\tilde{\mu}_{i,1} = \mathbf{u}_1^\top \boldsymbol{\mu}_i$ and $\tilde{\sigma}_{i,1}^2 = \mathbf{u}_1^\top \boldsymbol{\Sigma}_i \mathbf{u}_1$.

The ARS is a function applied to RT to detect patterns of points collinear to a given direction measured by θ . Given a superadditive concentration function $\kappa(\cdot)$, the ARS is defined as

$$\mathcal{S}[f](\theta) = \int_{-\infty}^{+\infty} \kappa(\mathcal{R}[f](\theta, \rho)) d\rho. \quad (5)$$

A standard concentration function is $\kappa(x) = x^2$, which is implicitly used in the remaining. Thus, the square of a sum of Gaussian kernels $\kappa(\mathcal{R}[f])$ in Equation (1) consists of double products of Gaussians that can be integrated. The equation of the ARS of a GMM has the form

$$\mathcal{S}[f](\theta) = \sum_{i=1}^{n_p} \sum_{j=1}^{n_p} w_i w_j \psi_{ij}(\theta). \quad (6)$$

The ARS kernel functions $\psi_{ij}(\theta)$ are equal to the Gaussian-like function

$$\psi_{ij}(\theta) = n\left(\mathbf{u}_1^\top(\theta)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j), \mathbf{u}_1^\top(\theta)(\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j)\mathbf{u}_1(\theta)\right). \quad (7)$$

The ARS $\mathcal{S}[f](\theta)$ is π -periodic and can be expanded into Fourier series. The Fourier series of ARS kernel ψ_{ij} is

$$\psi_{ij}(\theta) = a_0^{(ij)} + \sum_{k=1}^{+\infty} \left(a_k^{(ij)} \cos(2k\theta) + b_k^{(ij)} \sin(2k\theta) \right). \quad (8)$$

In the *isotropic* case, the Fourier coefficients $a_k^{(ij)}$ and $b_k^{(ij)}$ have closed-form equations [2]

$$a_k^{(ij)} = 2 I_k(\lambda_{ij}) e^{-\lambda_{ij}} (-1)^k \cos(2k\theta_{ij}), \quad (9)$$

$$b_k^{(ij)} = 2 I_k(\lambda_{ij}) e^{-\lambda_{ij}} (-1)^k \sin(2k\theta_{ij}), \quad (10)$$

where $I_k()$ is the modified Bessel function of the first kind, $\lambda_{ij} = \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2 / (8\sigma^2)$ and $\theta_{ij} = \angle(\boldsymbol{\mu}_j - \boldsymbol{\mu}_i)$. In the anisotropic case, the coefficients are numerically evaluated from samples. The Fourier expansion allows compact representation of the whole ARS in the form of a weighted sum of Fourier coefficients as the one in Equation (6). The advantages of this formulation emerge in the rotation estimation method presented in Section 3.2.

The main property of ARS lies in its invariance to translation \mathbf{t} and angular shift. If translation \mathbf{t} and rotation \mathbf{R} with angle δ are applied to a point set represented by density function $f(\mathbf{r})$, then the spectrum of the transformed point set satisfies the equation

$$\mathcal{S}[f(\mathbf{R}(\delta) \mathbf{r} + \mathbf{t})](\theta) = \mathcal{S}[f(\mathbf{r})](\theta + \delta). \quad (11)$$

Thus, the spectrum of a transformed point cloud is a shifted copy of the spectrum of the original point cloud, where the shift corresponds to the rotation angle δ . Rotation can be estimated using a proper metric and procedure for comparing spectra. In the next section, we show how this important property can be exploited to achieve this goal.

3.2. Registration Algorithm

ARS can be effectively applied to the estimation of the rigid transformation between two point clouds that represent the same scene from different viewpoints. Rotation is generally the more difficult part of the registration problem, and ARS translation-invariance allows decoupling of rotation estimation from translation estimation. As previously stated, when there is a rotation with angle δ between a source and a target point set represented by density functions, respectively, $f_S(\mathbf{r})$ and $f_T(\mathbf{r})$, spectrum $\mathcal{S}[f_S]$ is the shifted copy of $\mathcal{S}[f_T]$. The shift angle can be computed by searching the maximum of the following correlation function,

$$C[f_S, f_T](\delta) = \frac{1}{\pi} \int_0^\pi \mathcal{S}[f_S](\theta + \delta) \mathcal{S}[f_T](\theta) d\theta. \quad (12)$$

between the source and destination spectra. Since each ARS can be represented as a Fourier series, the correlation function is elegantly expressed in the form of convolution. The global maximum δ^* of $C[f_S, f_T]$ can be efficiently found through a branch-and-bound procedure on the angular domain. More details can be found in [2,3]. It can be observed that, since ARS is π -periodic, the real rotation angle is either δ or $\delta + \pi$. The assessment of translation enables disambiguation between the two candidate values of rotation.

To complete global registration, a branch-and-bound procedure inspired by [42] has been chosen. The objective function to be maximized is the number of overlapping point pairs between destination and the translated source point clouds. A point pair is overlapping if the distance in it is less than tolerance ε . Given a closed box $\mathcal{B} \subset \mathbb{R}^2$, the lower and upper bounds of the number of matching pairs are estimated. The lower bound is computed by counting the number of source points with a corresponding destination point belonging to the box \mathcal{B} centered on the source point. The upper bound excludes from this

counting the points clearly without matching. The translation is estimated as the center of the optimal box \mathcal{B}_{opt} which has the largest number of inliers.

4. Cud-ARS

4.1. Parallelization Setup and Enhancement

The ARS computational complexity is dominated by the evaluation of a spectrum kernel for each possible pair of points, as it is clear from Equation (6). Since $\psi_{ij}(\theta) = \psi_{ji}(\theta)$, the final spectrum does not depend on the processing order of the points with indices i and j . Conventionally, the point with an index (say, i) in the external loop is called *source point*, and the one with an index (say, j) in the nested loop is the *destination point*. The simple idea behind the GPU enhancement of isotropic ARS is to execute the largest number of computations in parallel.

In order to maximize computational throughput across the GPU kernels and to distribute computation in an efficient manner by doing so, a virtual grid-like structure to compute ARS spectra has been introduced. The goal has been to keep the number of threads a power of 2, starting from a minimum of 32. To preserve the square shape of the grid, its last cells still perform computation on padding data as part of ARS spectra computation pipeline, even though the padding data are not related to real pairs of points.

Parameter *max_chunk_size* corresponds to the maximum number of points taken as input into one Cud-ARS processing CUDA grid. If the size of the input source or destination data is greater than *max_chunk_size*, the Cud-ARS coefficient update is iterated until all the source-to-destination point comparisons are processed. This step is necessary as the internal memory of modern GPUs is rather large, but still finite. An additional check is performed to avoid Cud-ARS coefficient computation steps with small data chunks. When chunks of data slightly surpass the maximum size allowed, but they still fit the GPU memory, they are processed in one step to avoid the additional iteration that would slow down the whole pipeline (especially as less CPU-GPU transfers with high throughput are more efficient than multiple transfers with less data).

Cud-ARS is implemented as a three-step procedure. First, an indexed table of ARS coefficients is computed. The *tid*th element of this table corresponds to the evaluation of ARS on points with indices i, j , with i, j computed as explained in the *getIJfromTID()* method presented in Algorithm 1.

Algorithm 1 Obtain I and J from TID

```

nId ← n − 1; //indices vary between 0 and n−1
tid_tmp ← tid
while tid_tmp ≥ 0 do
    tid_tmp −= (nId − i);
    i ← i + 1;
end while
return i ← i − 1;
nId ← n − 1; //indices vary between 0 and n−1
tid_tmp ← tid
while i > 0 do
    tid_tmp −= (nId − i);
    i ← i − 1;
end while
return j ← tid + 1;

```

The *tid* indexing has been introduced in order to avoid excessive memory usage for storing useless computation outputs. As a matter of fact, the ARS coefficient matrix stores only the evaluation of ARS between points corresponding to non-null elements of the *strictly triangular* cost/matching matrix of the two datasets. Outputs i and j of Algorithm 1

correspond to the couple of point indices from source and, respectively, destination sets to be processed.

The most significant and computationally expensive step of this first part of the algorithm is the ARS kernel computation, which is on its own composed of two steps: an evaluation of the *PNEBI* (Product of Negative Exponential and Bessel functions on the first kind) which is defined as

$$PNEBI(k, x) = 2.0 * \exp(-x) * besseli(k, x), \quad (13)$$

where $besseli(k, x)$ is the modified Bessel function of the first kind of order k . The evaluation of $besseli(k, x)$ is based on recurrence. Hence, it is convenient to evaluate all coefficients for $k = 0, \dots, arsOrder$ and to store them into vector *pnebis*. Said vector is used to update the coefficient matrix, as illustrated in the discussion of Algorithm 2.

Algorithm 2 ARS Coefficient Downward Update

```

coffs = coffsMat = matrix.empty()

factor ← weight =  $\frac{1.0}{(numPts^2) * \sqrt{(4 * \pi * sigma^2)}}$ 

rowIdx ← tid = TODO

firstIdx = rowIdx * ncols + 0
coffs[firstIdx] += 0.5 * factor * pnebis0

delta ← meansj − meansi
phi ← atan2(delta.y, delta.x)
cth2 ← cos(2.0 * phi)
sth2 ← sin(2.0 * phi)
sgn ← −1.0
cth ← cth2
sth ← sth2

for k = 1 : fourierOrder do
  evenIdx ← rowIdx * ncols + 2k
  oddIdx ← rowIdx * ncols + (2k + 1)
  coffs[evenIdx] += factor * pnebis[k] * sgn * cth
  coffs[oddIdx] += factor * pnebis[k] * sgn * sth

  sgn ← −sgn
  ctmp ← cth2 * cth − sth2 * sth
  stmp ← sth2 * cth + cth2 * sth
  cth ← ctmp
  sth ← stmp
end for

```

It can be noted that a large part of the computational load is due to the estimation of such 20 ÷ 30-sized vector for each pair of points to be evaluated with ARS. The aforementioned Algorithm 2 runs on the GPU in a for-stride loop guarded by the following instructions:

$$\begin{aligned}
 index &= blockIdx.x * blockDim.x + threadIdx.x \\
 stride &= blockDim.x * gridDim.x \\
 tot &= gridDim.x * blockDim.x \\
 for(tid = index; tid < tot; tid += stride)
 \end{aligned} \quad (14)$$

where *index* runs through a single block, while *stride* is supposedly the total number of threads in the grid. The goal is to fit as many coefficient computations as possible into one single grid.

Then, ARS computation for rotation estimation proceeds by summing the coefficients across each Fourier order, i.e., summing them along each (virtual) column of the coefficient matrix. However, due to the large number of rows to be summed for each column, it has appeared more profitable to subdivide this summing procedure into two steps: first, a partial column-wise sum of the coefficient matrix entries in chunks of consecutive rows (each having a fixed number of rows *part_sum_numrows*) is been computed; then, the sum of these partial sums is computed (still column-wise).

One last important consideration to be made is on how to approach large input datasets. The considerable amount of data needed for each thread of the kernels' computation quickly fills the central memory of the GPUs, which for *general purpose* personal PCs rarely goes over 25 GB. Going even further into the exploitation of the separability of ARS coefficient parallel computation, the natural way for presented Cud-ARS to solve problems with input point sets with a size over ~ 5000 points is subdivision of the datasets in chunks, and then the processing of each of the chunks separately. The 4096 value reported in Table 1 has been chosen as the appropriate maximum chunk size by empirical testing. The value of 256 representing the number of threads in each block has been selected in a similar fashion. Since kernel parallelization parameters vary depending on the input dataset chunk size, when the number of input points is greater than *chunk_max_size*, an iterative procedure resembling the subdivision into partial sums and total sums explained for computing ARS coefficients of each Fourier order is deployed. First, the parallelization parameters are updated according to input data chunk size. Then, the computation of ARS coefficients is summed for each combination of data chunks among source and destination point sets. Finally, it has to be noted that even during this process, the strictly triangular indexing for the ARS coefficient matrices is kept even when the need arises to evaluate ARS across multiple different chunk combinations, coming from source and destination point sets.

Table 1. Parameter configuration.

Description	Symbol	Value
ARS Fourier order	n_f	32
ARS stdev	σ_{min}	1.0 (mpeg7), 0.05 (maps)
ARS tolerance on B&B	$\Delta\theta$	0.5°
Coeff Matrix Rows	n_{rows}	$\frac{num_pts*(num_pts-1)}{2}$
Coeff Matrix Cols	n_{cols}	$2n_f + 2$
Prlz Grid Size	$grid_sz$	n_{rows}
Number of Blocks	$blks$	$\lfloor \frac{gridTotalSize}{pp.blockSz} \rfloor + 1$
Number of Threads	$threads$	256
Coeff Matrix Tot Size	$cffs_mat_tot_sz$	$grid_sz * n_{cols}$
Max Chunk Size	max_chunk_size	4096

4.2. Full Registration and Mapping

While the precise estimation of rotation between two point clouds is an important task/primitive for plenty of robotics applications, it is even more important to integrate it in a more complete pipeline.

The generally accurate rotation estimation of ARS is used as the initial guess for the full pose estimation, paired with the translation estimation procedure presented in Section 3.2. The registration algorithm has been used to build maps by accumulating point clouds aligned with scan-to-scan matching. This solution, although simple, allows the appreciation of the effectiveness of the proposed pose estimation algorithm. Results are discussed in the next section.

5. Experiments

In order to evaluate Cud-ARS, multiple and diverse sets of experiments have been conducted. The first goal has been to assess whether Cud-ARS is able to obtain the same accuracy performances as ARS and other state-of-the-art methods on commonly used datasets (see Section 5.1). Second, the proposed methods for scan-matching based registration have been tested on real-world robotic simulations, assessing their ability in reconstructing the trajectory and their usability in building an occupancy grid map of the robot's movements (see Section 5.2). An implementation of Cud-ARS is available in a public repository (available online at: <https://github.com/ErnestF22/cudars/>, accessed on 16 October 2023).

5.1. Cud-ARS Rotation Estimation

Cud-ARS evaluation has been performed on three datasets (namely *mpeg7*, *map* and *scan*) used also in previous works. These datasets contain a good range of the characteristics that can be found in robotic tasks with heterogeneous sensors and setup.

The *Mpeg7* datasets [43] are composed of images of more than 1000 different shapes that are sampled as point clouds. Three major transformations have been applied for testing the robustness of Cud-ARS:

- *Noise tests.* This transformation adds Gaussian noise with a given standard deviation σ to the points coordinates. The value of σ is varied in the interval of $0 \div 50$, with the maximum dimension of a point set varying from 300 to 900.
- *Occlusion tests.* An occluded version of a point set is constructed by removing all points lying inside a randomly generated circle. The center of the circle is a randomly chosen point of the dataset, while the radius is proportional to the size of the point set. In particular, if the points are contained in a bounding box of size $b_x \times b_y$, the radius is equal to $\beta \sqrt{b_x b_y}$, where $0 \leq \beta \leq 1$ is the occlusion rate. Occlusion rate β is varied up to 50%.
- *Rand tests.* This transformation adds γn_{in} random points to an input point set of n_{in} points, where γ is the random point rate. The random points are uniformly distributed on a circle centered on the point set's mean point, and with a radius double the size of bounding box. The maximum value of the random point rate γ used in the tests is 300, i.e., random points are at most three times the number of shape points.

The different transformations applied to assess the robustness of the rotation estimation are further explained in [2]. The *map* dataset is composed of occupancy grid maps obtained using the Cartographer [44] ROS tool on laser scans acquired at the University of Parma (available online: https://www.ce.unipr.it/~rizzini/papers/datasets/VLP-16_Unipr_DepartmentHall_dataset_20171102/, accessed on 16 October 2023), and on a public Deutsche Museum dataset (available online: <https://google-cartographer-ros.readthedocs.io/en/latest/data.html>, accessed on 16 October 2023). The *scan* dataset is made of laser scans traditionally used within the SLAM community, named after the place of acquisition: *fr-log*, *fr079*, *intel-lab* and *mit-csail* (URL to the *scan* dataset can be found in the Data Availability section). Each of these datasets contains about 5000 scans.

As expected, experiments on all these common robotic datasets show the same results for pose estimation as isotropic ARS, but with a substantial improvement in terms of speed. In these tests, Cud-ARS is compared against two previous versions of ARS (CPU Isotropic and Anisotropic), and the Hough Spectrum [23] from Censi et al.

Results are shown in Figures 1–3. The speed-up of the newly introduced Cud-ARS is easily noticeable across all experiments. The same can be said for the limited growth in execution time when other algorithms heavily slow down instead. It can be observed that the one algorithm performing similarly to Cud-ARS is Hough Spectrum (HS). While HS slightly outperforms Cud-ARS on some tests, we can still see that their speed is always very similar, and that they are constantly much faster than their counterparts. Cud-ARS achieves the same accuracy and precision as the original Isotropic ARS. This happens because, despite performing most of the computation in parallel on the GPU, Cud-ARS computes

the same exact spectrum values in a more efficient way. Furthermore, the variance in terms of mean execution time is much higher for previous versions of ARS, which means that the newly introduced Cud-ARS can see a great increase in the number of potential applications. As a matter of fact, its performance is reasonably constant when dealing with diverse types of datasets (for example, in terms of the number of input points) whose elaboration may have previously required an excessive amount of time for online processing.

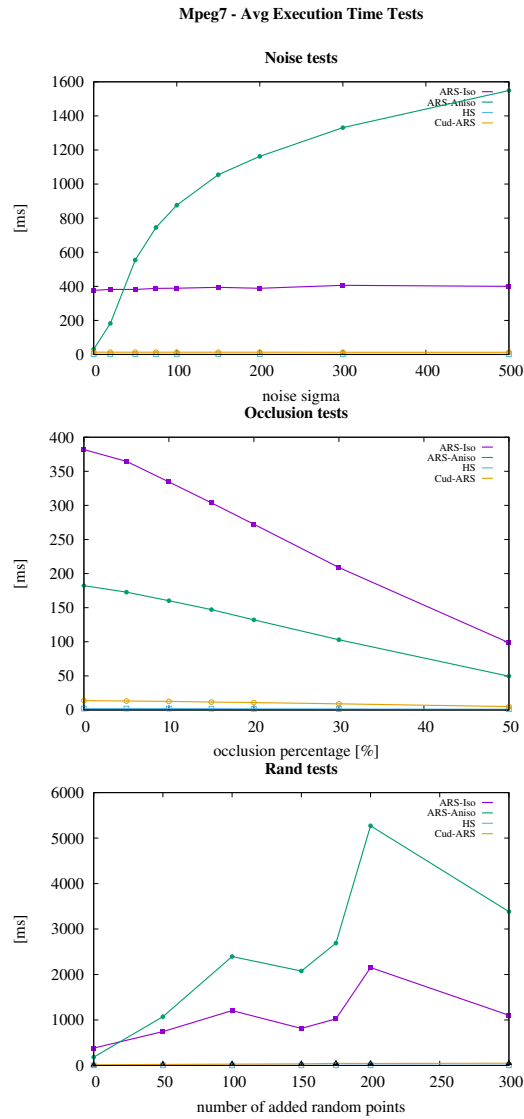


Figure 1. MPEG7 dataset execution time results.

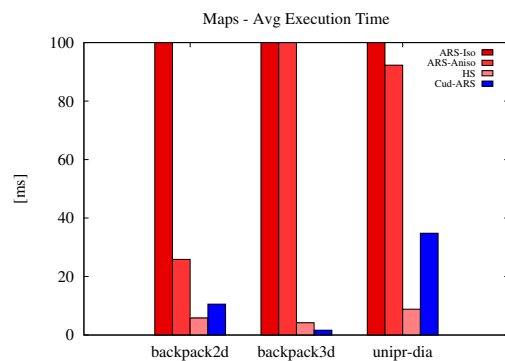


Figure 2. Maps dataset execution time results.

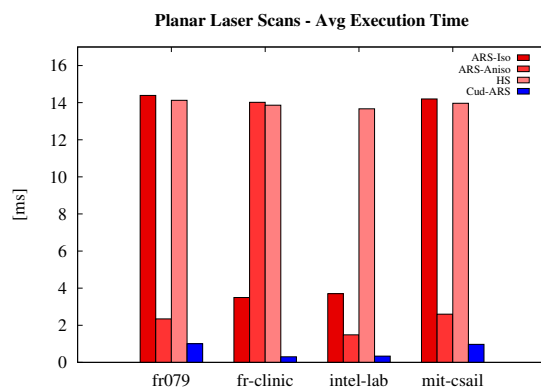


Figure 3. Scan dataset execution time results.

5.2. Registration and Mapping

In general, ARS registration has been able to keep track of the trajectories, even if working more similarly to a scan matcher rather than a classic registration method. To attest this, the proposed registration pipeline has been tested using a dataset acquired by a Pioneer 3DX robot in the main hallway of the Department of Engineering and Architecture at the University of Parma, which consists of a more than 100 m long corridor with branches and tables. The measurements included in the dataset are the robot odometry, the laser scans collected by Sick LMS100, and the point clouds collected by multilayer LIDAR Velodyne VLP16. In particular, we use two sequences called *uniprdia_0* and *uniprdia_1*.

The proposed ARS registration algorithm has been compared with Cartographer [44] and Hector SLAM (briefly Hector) [45]. At each iteration, the estimation given by ARS is based only on the alignment of the current laser scan with the previous one, without any initial guess. Conversely, Hector and Cartographer are full mapping systems that align and merge each new scan with the current map. Moreover, Hector uses the initial guess provided by the robot odometry and Cartographer also integrates the 3D measurements from the LIDAR. The goal of this tests is to display the robustness of ARS estimation, albeit based on scan-to-scan comparison.

Figures 4 and 5 display the occupancy grid maps, as well as the estimated trajectories obtained with the three methods. The occupancy grid maps are obtained through online collection and merging of the aligned laser scans using Octomap [46], without removing inconsistencies, for a fair comparison of the three approaches. Even though it uses less data, ARS is able to estimate locally accurate trajectories. As it can be seen, though, in dataset *uniprdia_0*, after the robot performs a U-turn, the algorithm loses track of the real orientation. The loss in orientation might be due to a non sufficient rate of consecutive scan matching during the turn, or, even more trivially, to a simple badly acquired scan. More accurate results have been obtained on dataset *uniprdia_1*, where our method has been able to keep track of a more complex trajectory. It is well known that the absence of any kind of memory of previous states and maps while performing registration can lead to effects like this. However, these tests still show the stability of ARS' scan-to-scan rotation and translation estimation, even if the need for adding a more complex mapping pipeline to the ARS project still appears necessary.

Table 2 reports the Average Translational Error (ATE) and Average Rotational Error (ARE) for Hector and ARS with respect to the trajectory of Cartographer (used as groundtruth) in the two sequences. As expected, ARS errors are larger, but significantly limited for a scan matching algorithm.

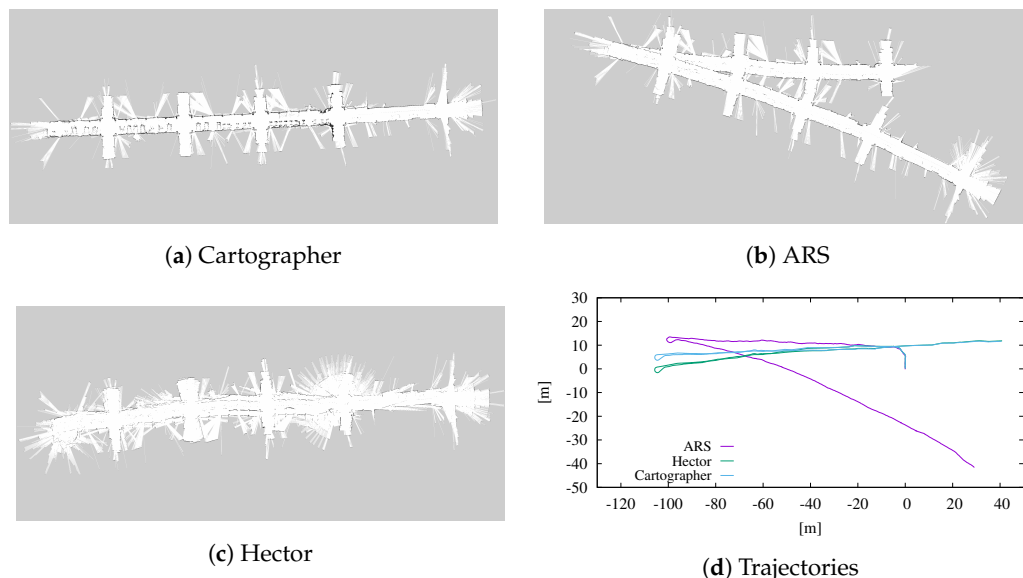


Figure 4. Occupancy grid maps and estimated trajectories of dataset *uniprdia_0* obtained using Cartographer, Hector SLAM and ARS-based registration. The occupancy grid maps are computed using Octomap that overlaps online raw laser scan data. Error propagation after the U-turn is visible.

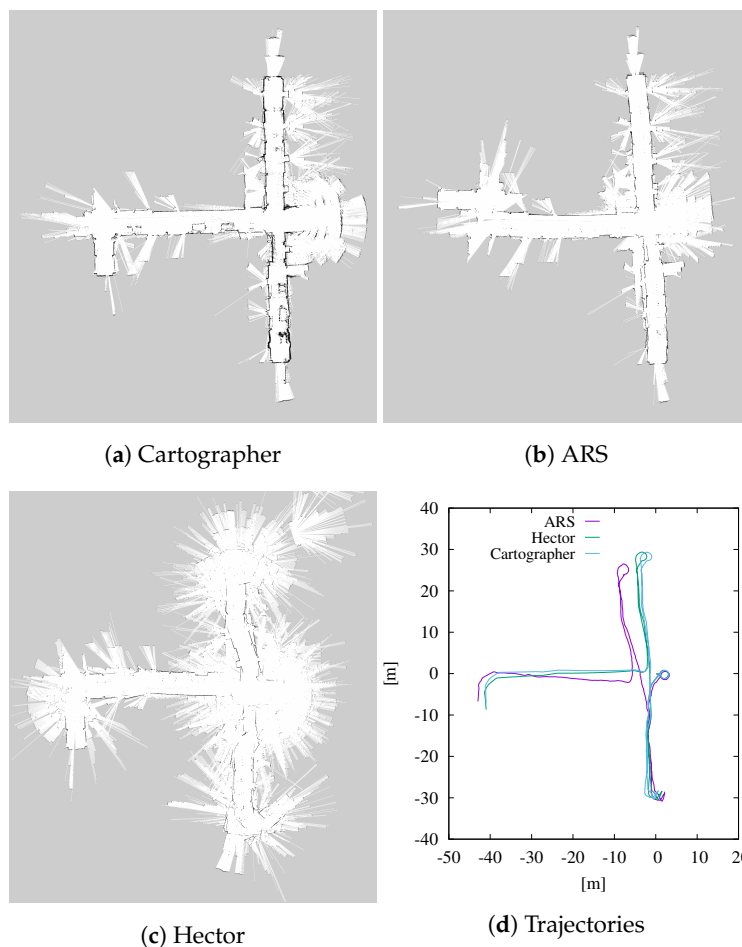


Figure 5. Occupancy grid maps and estimated trajectories of dataset *uniprdia_1* obtained using Cartographer, Hector SLAM and ARS-based registration. The occupancy grid maps are computed using Octomap that overlaps online raw laser scan data. Here, a more complex trajectory is kept track of with limited error.

Table 2. Average Translational Error (ATE) and Average Rotational Error (ARE) obtained by Hector and ARS on the given sequences of datasets *uniprdia_0* and *uniprdia_1*.

Dataset	Length (m)	Hector		ARS	
		ATE (%)	ARE ($10^{-2\circ/m}$)	ATE (%)	ARE ($10^{-2\circ/m}$)
<i>uniprdia_0</i>	262.28	3.87	7.18	19.78	31.66
<i>uniprdia_1</i>	180.34	3.14	6.66	11.06	32.58

Another set of experiments has been conducted on the *scan* dataset. The goal of these experiments has been the one of assessing the ability to correctly estimate rigid transformations between subsequent scans in each dataset. Pose estimation tests have been separated into rotation and translation estimation, respectively. For rotation, the results obtained with the six methods are compared against the ground truth information contained in the datasets, as explained in [3]. Translation has been estimated after rotating the point clouds by an angle estimated through Isotropic ARS. The results are reported in Figure 6. ARS methods achieve an error on par with or lower than the other rotation estimation methods, while just a bit over the 1 cm scan resolution parameter when estimating translations. It has to be noted that several failed estimations are due to non-overlapping scans.

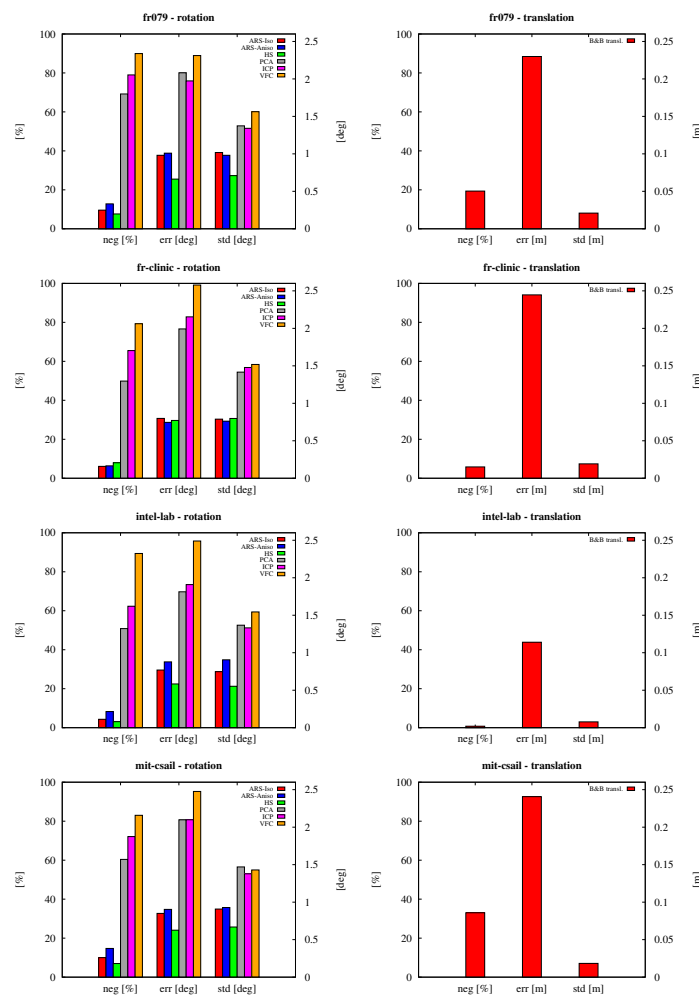


Figure 6. Registration accuracy in estimation of rotation (top) and of translation (bottom) on scan datasets *fr079*, *fr-clinic*, *intel-lab* and *mit-csail* (from left to right). For each set of tests, negative (failed) estimation percentage, average rotation error, standard deviation and translation mean error ($^{\circ}$) and (m), respectively) are reported.

5.3. Discussion

The rotation estimation results presented in Sections 5.1 and 5.2 show that the proposed method is constantly able to estimate rotation between pairs of point clouds with a < 1 [deg] accuracy. These results have been verified on multiple standard datasets, comprising images, occupancy grid maps, and planar laser scans. Cud-ARS achieves a speed up in the execution time of up to two orders of magnitude (i.e., up to $100\times$), rendering it at least on par with the grid-based HS in most tests. The scan-based mapping tests also assess the potential application of the proposed method to mapping tasks, even if one case shows that the incomplete representation provided by a pair of scans may lead the algorithm to losing track of the real robot orientation. However, this set of experiments has been designed to show the adequacy of the parallel rotation estimation and of the B&B-based translation algorithm as building blocks of a real-world mapping system.

6. Conclusions

This work has presented Cud-ARS, an algorithm able to perform fast and globally optimal registration on 2D point clouds. Cud-ARS is designed as a parallel algorithm for efficient computation of the Radon Spectrum. The original ARS has been reformulated in order to efficiently run on Nvidia GPUs. The ARS-based pipeline has been improved in its capability to perform full registration due to a B&B-based translation estimation method. Experiments conducted to compare and evaluate the presented method against state-of-the-art algorithms show the large improvements in computational speed of Cud-ARS, as well as the high accuracy of the method. The code is available on a public repository. The registration accuracy has been tested against more complex state-of-the-art mapping frameworks, and despite its current lack in place recognition capabilities, shows good potential even in real-world applications. Future work will include a more stable mapping pipeline, with online updates that will be performed on the GPU, in order to exploit the computational advantages even further.

Author Contributions: E.F.: Conceptualization, Methodology, Software, Validation, Writing—original draft, Writing—review and editing. D.L.R.: Conceptualization, Methodology, Software, Validation, Writing—original draft, Writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Code is available in public repository: <https://github.com/ErnestF22/cudars/>, accessed on 16 October 2023; MPEG7 dataset is available at: https://www.ce.unipr.it/~rizzini/papers/datasets/mpeg7_dataset/, accessed on 16 October 2023; scan dataset is available at: https://www.ce.unipr.it/~rizzini/papers/datasets/laser2d_dataset/, accessed on 16 October 2023.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
GPU	Graphics Processing Unit
ARS	Angular Radon Spectrum

References

1. Besl, P.; McKay, H. A method for registration of 3-D shapes. *IEEE Trans. Pat. Anal. Mach. Intel* **1992**, *14*, 239–256. [CrossRef]
2. Lodi Rizzini, D. Angular Radon Spectrum for Rotation Estimation. *Pattern Recognit.* **2018**, *84*, 182–196. [CrossRef]
3. Lodi Rizzini, D.; Fontana, E. Rotation Estimation Based on Anisotropic Angular Radon Spectrum. *IEEE Robot. Autom. Lett.* **2022**, *7*, 7279–7286. [CrossRef]

4. Yang, H.; Shi, J.; Carlone, L. Teaser: Fast and certifiable point cloud registration. *IEEE Trans. Robot.* **2020**, *37*, 314–333. [[CrossRef](#)]
5. Yang, J.; Li, H.; Campbell, D.; Jia, Y. Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *38*, 2241–2254. [[CrossRef](#)]
6. Huang, Y.; Ling, K.V.; See, S. Solving quadratic programming problems on graphics processing unit. *ASEAN Eng. J.* **2011**, *1*, 76–86.
7. Frasch, J.V.; Sager, S.; Diehl, M. A parallel quadratic programming method for dynamic optimization problems. *Math. Program. Comput.* **2015**, *7*, 289–329. [[CrossRef](#)]
8. Olson, E. Real-Time Correlative Scan Matching. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, 12–17 May 2009; pp. 4387–4393.
9. Premebida, C.; Ambrus, R.; Marton, Z.C. Intelligent robotic perception systems. In *Applications of Mobile Robots*; IntechOpen: London, UK, 2018.
10. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [[CrossRef](#)]
11. Vizzo, I.; Guadagnino, T.; Mersch, B.; Wiesmann, L.; Behley, J.; Stachniss, C. KISS-ICP: In Defense of Point-to-Point ICP—Simple, Accurate, and Robust Registration if Done the Right Way. *IEEE Robot. Autom. Lett.* **2023**, *8*, 1029–1036. [[CrossRef](#)]
12. Rusinkiewicz, S.; Levoy, M. Efficient variants of the ICP algorithm. In Proceedings of the International Conference on 3-D Digital Imaging and Modeling, Quebec, QC, Canada, 28 May–1 June 2001.
13. Biber, P.; Straßer, W. The normal distributions transform: A new approach to laser scan matching. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, MI, USA, 1–5 October 2003; pp. 2743–2748.
14. Magnusson, M.; Lilienthal, A.; Duckett, T. Scan registration for autonomous mining vehicles using 3D-NDT. *J. Field Robot.* **2007**, *24*, 803–827. [[CrossRef](#)]
15. Maken, F.; Ramos, F.; Ott, L. Stein ICP for Uncertainty Estimation in Point Cloud Matching. *IEEE Robot. Autom. Lett.* **2022**, *7*, 1063–1070. [[CrossRef](#)]
16. Kolpakov, A.; Werman, M. An approach to robust ICP initialization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 1–9. [[CrossRef](#)] [[PubMed](#)]
17. Jaimez, M.; Monroy, J.; Lopez-Antequera, M.; Gonzalez-Jimenez, J. Robust Planar Odometry Based on Symmetric Range Flow and Multiscan Alignment. *IEEE Trans. Robot.* **2018**, *34*, 1623–1635. [[CrossRef](#)]
18. Zhang, J.; Singh, S. Low-drift and Real-time Lidar Odometry and Mapping. *Auton. Robot.* **2017**, *41*, 401–416. [[CrossRef](#)]
19. Shan, T.; Englot, B. LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 4758–4765.
20. Wang, H.; Wang, C.; Chen, C.; Xie, L. F-LOAM: Fast LiDAR Odometry and Mapping. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 4390–4396.
21. Shan, T.; Englot, B.; Meyers, D.; Wang, W.; Ratti, C.; Rus, D. LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 5135–5142. [[CrossRef](#)]
22. Myronenko, A.; Song, X. Point Set Registration: Coherent Point Drift. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 2262–2275. [[CrossRef](#)]
23. Censi, A.; Iocchi, L.; Grisetti, G. Scan Matching in the Hough Domain. In Proceedings of the IEEE International Conference on Robotics & Automation (ICRA), Barcelona, Spain, 18–22 April 2005.
24. Censi, A.; Carpin, S. HSM3D: Feature-less global 6DOF scan-matching in the Hough/Radon domain. In Proceedings of the IEEE International Conference on Robotics & Automation (ICRA), Kobe, Japan, 12–17 May 2009; pp. 3899–3906.
25. Zhang, Y.; Severinsen, O.A.; Leonard, J.J.; Carlone, L.; Khosoussi, K. Data-Association-Free Landmark-based SLAM. *arXiv* **2023**, arXiv:2302.13264.
26. Dellaert, F.; Kaess, M. Factor graphs for robot perception. *Found. Trends Robot.* **2017**, *6*, 1–139. [[CrossRef](#)]
27. Sola, J.; Vallvé, J.; Casals, J.; Deray, J.; Fourmy, M.; Atchuthan, D.; Corominas-Murtra, A.; Andrade-Cetto, J. WOLF: A modular estimation framework for robotics based on factor graphs. *IEEE Robot. Autom. Lett.* **2022**, *7*, 4710–4717. [[CrossRef](#)]
28. Roussillon, C.; Gonzalez, A.; Solà, J.; Codol, J.M.; Mansard, N.; Lacroix, S.; Devy, M. RT-SLAM: A generic and real-time visual SLAM implementation. In Proceedings of the International Conference on Computer Vision Systems, Sophia Antipolis, France, 20–22 September 2011; pp. 31–40.
29. Ila, V.; Polok, L.; Solony, M.; Svoboda, P. SLAM++—A highly efficient and temporally scalable incremental SLAM framework. *Int. J. Robot. Res.* **2017**, *36*, 210–230. [[CrossRef](#)]
30. Abouzahir, M.; Latif, R.; Ramzi, M.; Sbihi, M. OpenCL and OpenGL Implementation of Simultaneous Localization and Mapping Algorithm using High-End GPU. In Proceedings of the ITM Web of Conferences. EDP Sciences, Craiova, Romania, 29 June–2 July 2022; Volume 46, p. 04001.
31. Kumar, A.; Chellappa, R. Disentangling 3d pose in a dendritic cnn for unconstrained 2d face alignment. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 430–439.

32. Chen, H.; Manhardt, F.; Navab, N.; Busam, B. TexPose: Neural Texture Learning for Self-Supervised 6D Object Pose Estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 4841–4852.
33. Kehl, W.; Manhardt, F.; Tombari, F.; Ilic, S.; Navab, N. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 1521–1529.
34. Milioto, A.; Stachniss, C. Bonnet: An open-source training and deployment framework for semantic segmentation in robotics using crns. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 7094–7100.
35. Milioto, A.; Vizzo, I.; Behley, J.; Stachniss, C. RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), The Venetian Macao, Macau, 4–8 November 2019; pp. 4213–4220. [[CrossRef](#)]
36. Collet, A.; Srinivasa, S.S. Efficient multi-view object recognition and full pose estimation. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, Alaska, 3–8 May 2010; pp. 2050–2055.
37. Collet, A.; Martinez, M.; Srinivasa, S.S. The moped framework: Object recognition and pose estimation for manipulation. *Int. J. Robot. Res.* **2011**, *30*, 1284–1306. [[CrossRef](#)]
38. Deng, X.; Xiang, Y.; Mousavian, A.; Eppner, C.; Bretl, T.; Fox, D. Self-supervised 6d object pose estimation for robot manipulation. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Virtual, 31 May–31 August 2020; pp. 3665–3671.
39. Austin, J.; Corrales-Fatou, R.; Wyetzner, S.; Lipson, H. Titan: A parallel asynchronous library for multi-agent and soft-body robotics using nvidia cuda. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Virtual, 31 May–31 August 2020; pp. 7754–7760.
40. Ha, S.W.; Han, T.D. A scalable work-efficient and depth-optimal parallel scan for the GPGPU environment. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *24*, 2324–2333. [[CrossRef](#)]
41. Deans, S. Radon and Abel Transforms. In *The Transforms and Applications Handbook*, 2nd ed.; Poularikas, A., Ed.; CRC Press: Boca Raton, FL, USA, 2000; pp. 1–95.
42. Liu, Y.; Wang, C.; Song, Z.; Wang, M. Efficient global point cloud registration by matching rotation invariant features through translation search. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 448–463.
43. Bai, X.; Yang, X.; Latecki, L.; Liu, W.; Tu, Z. Learning Context-Sensitive Shape Similarity by Graph Transduction. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 861–874. [[CrossRef](#)] [[PubMed](#)]
44. Hess, W.; Kohler, D.; Rapp, H.; Andor, D. Real-Time Loop Closure in 2D LIDAR SLAM. In Proceedings of the IEEE International Conference on Robotics & Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1271–1278. [[CrossRef](#)]
45. Kohlbrecher, S.; Meyer, J.; von Stryk, O.; Klingauf, U. A Flexible and Scalable SLAM System with Full 3D Motion Estimation. In Proceedings of the IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), Kyoto, Japan, 1–5 November 2011.
46. Hornung, A.; Wurm, K.; Bennewitz, M.; Stachniss, C.; Burgard, W. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Auton. Robots* **2013**, *34*, 189–206. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.