Discriminative Optimization: Theory and Applications to Point Cloud Registration

Jayakorn Vongkulbhisal[†,‡], Fernando De la Torre[‡], João P. Costeira[†]
[†]ISR - IST, Universidade de Lisboa, Lisboa, Portugal
[‡]Carnegie Mellon University, Pittsburgh, PA, USA
jvongkul@andrew.cmu.edu, ftorre@cs.cmu.edu, jpc@isr.ist.utl.pt

Abstract

Many computer vision problems are formulated as the optimization of a cost function. This approach faces two main challenges: (1) designing a cost function with a local optimum at an acceptable solution, and (2) developing an efficient numerical method to search for one (or multiple) of these local optima. While designing such functions is feasible in the noiseless case, the stability and location of local optima are mostly unknown under noise, occlusion, or missing data. In practice, this can result in undesirable local optima or not having a local optimum in the expected place. On the other hand, numerical optimization algorithms in high-dimensional spaces are typically local and often rely on expensive first or second order information to guide the search. To overcome these limitations, this paper proposes Discriminative Optimization (DO), a method that learns search directions from data without the need of a cost function. Specifically, DO explicitly learns a sequence of updates in the search space that leads to stationary points that correspond to desired solutions. We provide a formal analysis of DO and illustrate its benefits in the problem of 2D and 3D point cloud registration both in synthetic and range-scan data. We show that DO outperforms state-of-the-art algorithms by a large margin in terms of accuracy, robustness to perturbations, and computational efficiency.

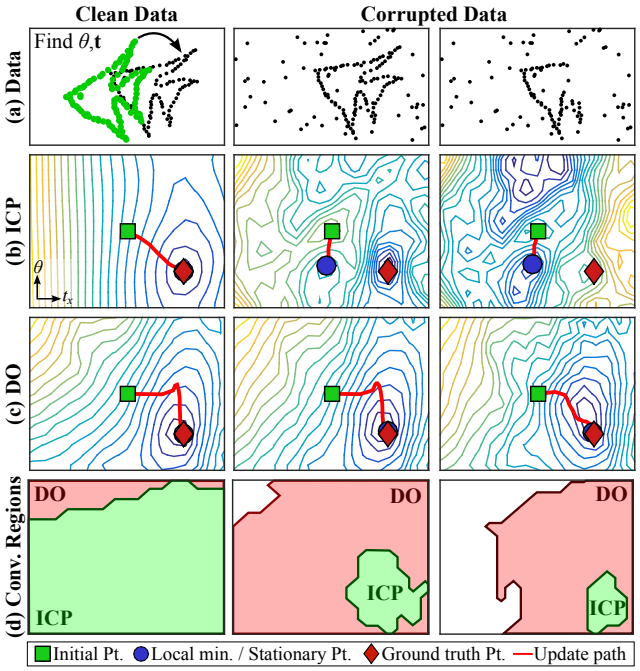


Figure 1. 2D point alignment with ICP and DO. (a) Data. (b) Level sets of the cost function for ICP. (c) Inferred level sets for the proposed DO. (d) Regions of convergence for ICP and DO. See text for detailed description (best seen in color).

1. Introduction

Mathematical optimization is fundamental to solving many problems in computer vision. This fact is apparent as demonstrated by the plethora of papers that use optimization techniques in any major computer vision conference. For instance, camera calibration, structure from motion, tracking, or registration are traditional computer vision problems that are posed as optimization problems. Formulating computer vision problems as optimization problems faces two main challenges: (1) Designing a cost function that has a local optimum that corresponds to a suitable solution. (2) Selecting an efficient and accurate algorithm for searching the parameter space. Traditionally these two steps have been treated independently, leading to different cost functions and search algorithms. However, in the presence of noise, missing data, or inaccuracies of the model, this traditional approach can lead to having undesirable local optima or even not having an optimum in the expected solution.

Consider Fig. 1a-left which illustrates a 2D alignment problem in a case of noiseless data. A good cost function for this problem should have a global optimum when the

two shapes overlap. Fig. 1b-left illustrates the level sets of the cost function for the Iterative Closest Point (ICP) algorithm [4] in the case of complete and noiseless data. Observe that there is a well-defined optimum and that it coincides with the ground truth. Given a cost function, the next step is to find a suitable algorithm that, given an initial configuration (green square), finds a local optimum. For this particular initialization, the ICP algorithm will converge to the ground truth (red diamond in Fig. 1b-left), and Fig. 1d-left shows the convergence region for ICP in green. However, in realistic scenarios with the presence of perturbations in the data, there is no guarantee that there will be a good local optimum in the expected solution, while the number of local optima can be large. Fig. 1b-center and Fig. 1b-right show the level set representation for the ICP cost function in the cases of corrupted data. We can see that the shape of cost functions have changed dramatically: there are more local optima, and they do not necessarily correspond to the ground truth (red diamond). In this case, the ICP algorithm with an initialization in the green square will converge to wrong optima. It is important to observe that the cost function is *only* designed to have an optimum at the correct solution in the ideal case, but little is known about the behavior of this cost function in the *surroundings* of the optimum and how it will change with noise. 主要的问题：无法预测cost function 在退化环境下的变化

To address the aforementioned problems, this paper proposes Discriminative Optimization (DO). DO exploits the fact that we often know from the training data where the solutions should be, whereas traditional approaches formulate optimization problems based on an ideal model. Rather than following a descent direction of a cost function, DO directly learns a sequence of update directions leading to a stationary point. These points are placed "by design" in the desired solutions from training data. This approach has three main advantages. First, since DO's directions are learned from training data, they take into account the perturbations in the *neighborhood* of the ground truth, resulting in more robustness and a larger convergence region. This can be seen in Fig. 2, where we show DO's update directions for the same examples of Fig. 1. Second, because DO does not optimize any explicit function (*e.g.*, ℓ_2 registration error), it is less sensitive to model misfit and more robust to different types of perturbations. Fig. 1c illustrates the contour level inferred[1] from the update directions learned by DO. It can be seen that the curve levels have a local optimum on the ground truth and fewer local optima than ICP in Fig. 1b. Fig. 1d shows that the convergence regions of DO change little despite the perturbations, and always include the regions of ICP. Third, to compute update directions, traditional approaches require the cost function to be differentiable or continuous, whereas DO's directions can always be computed. We also provide a proof of DO's convergence in the training set. We named our approach DO to reflect the idea of learning to find a stationary point directly rather than optimizing a "generative" cost function.

[1]Recall that DO does not use a cost function. The contour level is approximately reconstructed using the surface reconstruction algorithm [13] from the update directions of DO. For ICP, we used the optimal matching at each parameter value to compute the ℓ_2 cost.

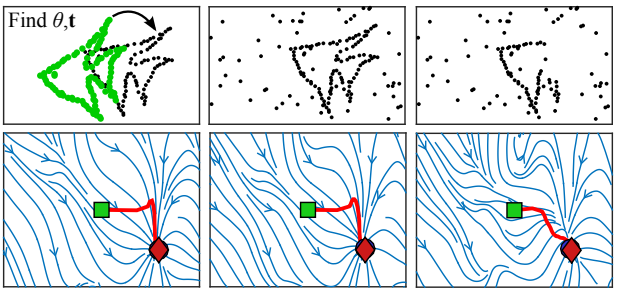


Figure 2. Update directions of DO in Fig. 1c.

We demonstrate the potential of DO in problems of rigid 2D and 3D point cloud registration. Specifically, we aim to solve for a 2D/3D rotation and translation that registers two point clouds together. Using DO, we learn a sequence of directions leading to the solution for each specific shape. In experiments on synthetic data and cluttered range-scan data, we show that DO outperforms state-of-the-art local registration methods such as ICP [4], GMM [14], CPD [17] and IRLS [3] in terms of computation time, robustness, and accuracy. In addition, we show how DO can be used to track 3D objects.

2. Related Work

2.1. Point cloud registration

Point cloud registration has been an important problem in computer vision for the last few decades. Arguably, Iterative Closest Point (ICP) [4] and its variants [9, 18] are the most well-known algorithms. These approaches alternate between solving for the correspondence and the geometric transformation until convergence. A typical drawback of ICP is the need for a good initialization to avoid a bad local minimum. To alleviate this problem, Robust Point Matching (RPM) [10] uses soft assignment instead of binary assignment. Recently, Iteratively Reweighted Least Squares (IRLS) [3] proposes using various robust cost functions to provide robustness to outliers and avoid bad local minima.

In contrast to the above point-based approaches, density-based approaches model each point as the center of a density function. Kernel Correlation [20] aligns the densities of the two point clouds by maximizing their correlation. Coherent Point Drift (CPD) [17] assumes the point cloud of one shape is generated by the density of the other shape, and solves for the parameters that maximize their likelihood. Gaussian Mixture Model Registration (GMM-

Reg) [14] minimizes the L_2 error between the densities of the two point clouds. More recently, [6] uses Support Vector Regression to learn a new density representation of each point cloud before minimizing L_2 error, while [11] models point clouds as particles with gravity as attractive force, and solves differential equations to obtain the registration.

In summary, previous approaches tackle the registration problem by first defining different cost functions, and then solving for the optima using iterative algorithms (e.g., expectation maximization, gradient descent). Our approach takes a different perspective by *not* defining new cost functions, but directly learning a sequence of updates of the rigid transformation parameters such that the stationary points match the ground truths from a training set.

2.2. Supervised sequential update (SSU) methods

Our approach is inspired by the recent practical success of supervised sequential update (SSU) methods for body pose estimation and facial feature detection. Cascade regression [8] learns a sequence of maps from features to refinement parameters to estimate the pose of target objects in single images. Explicit shape regression [7] learns a sequence of boosted regressors that minimizes error in search space. Supervised descent method (SDM) [23, 24] learns a sequence of descent maps as the averaged Jacobian matrices for solving non-linear least-squares functions. More recent works include learning both Jacobian and Hessian matrices [22]; running Gauss-Newton algorithm after SSU [1]; and using different maps in different regions of the parameter space [25]. Most of these works focus on facial landmark alignment and tracking.

Building upon previous works, we provide a new interpretation to SSU methods as a way of learning update steps such that the stationary points correspond to the problem solutions. This leads to several novelties. First, we allow the number of iterations on the learned maps to be adaptive rather than constant as in previous works. Second, we apply DO to the new problem setting of point cloud registration, where we show that the updates can be learned from only synthetic data. In addition, we provide a theoretical result on the convergence of training data and an explanation for why the maps should be learned in a sequence, which has been previously treated as a heuristic [24].

3. Discriminative Optimization (DO)

3.1. Intuition from gradient descent

DO aims to learn a sequence of update maps (SUM) to update an initial estimate of the parameter to a stationary point. The intuition of DO can be understood when compared with the underlying principle of gradient descent.

Let[2] $J : \mathbb{R}^p \to \mathbb{R}$ be a differentiable cost function. The gradient descent algorithm for minimizing J can be written as,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mu_k \frac{\partial}{\partial \mathbf{x}} J(\mathbf{x}_k), \tag{1}$$

where $\mathbf{x}_k \in \mathbb{R}^p$ is the parameter at step k, and μ_k is a step size. This update is performed until the gradient vanishes, *i.e.*, until a stationary point is reached [5].

In contrast to gradient descent where the updates are derived from a cost function, DO learns the updates from the training data. A major advantage is that no cost function is explicitly optimized and the *neighborhoods* around the solutions of perturbed data are taken into account when the maps are learned.

3.2. Sequence of update maps (SUM)

DO uses an update rule that is similar to (1). Let $\mathbf{h} : \mathbb{R}^p \to \mathbb{R}^f$ be a function that encodes a representation of the data (*e.g.*, $\mathbf{h}(\mathbf{x})$ extracts features from an image at positions $\mathbf{x}$). Given an initial parameter $\mathbf{x}_0 \in \mathbb{R}^p$, DO iteratively updates $\mathbf{x}_k, k = 0, 1, \ldots,$ using:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{D}_{k+1}\mathbf{h}(\mathbf{x}_k), \tag{2}$$

until convergence to a stationary point. The matrix[3] $\mathbf{D}_{k+1} \in \mathbb{R}^{p \times f}$ maps the feature $\mathbf{h}(\mathbf{x}_k)$ to an update vector. The sequence of matrices $\mathbf{D}_{k+1}, k = 0, 1, \ldots$ learned from training data forms the SUM.

Learning a SUM: Suppose we are given a training set as a set of triplets $\{(\mathbf{x}_0^{(i)}, \mathbf{x}_*^{(i)}, \mathbf{h}^{(i)})\}_{i=1}^N$, where $\mathbf{x}_0^{(i)} \in \mathbb{R}^p$ is the initial parameter for the i^{th} problem instance (*e.g.*, the i^{th} image), $\mathbf{x}_*^{(i)} \in \mathbb{R}^p$ is the ground truth parameter (*e.g.*, position of the object on the image), and $\mathbf{h}^{(i)} : \mathbb{R}^p \to \mathbb{R}^f$ provides information of the i^{th} problem instance. The goal of DO is to learn a sequence of update maps $\{\mathbf{D}_k\}_k$ that updates $\mathbf{x}_0^{(i)}$ to $\mathbf{x}_*^{(i)}$. In order to learn $\mathbf{D}_k$, we use the following least-square regression:

$$\mathbf{D}_{k+1} = \arg\min_{\tilde{\mathbf{D}}} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_*^{(i)} - \mathbf{x}_k^{(i)} + \tilde{\mathbf{D}}\mathbf{h}^{(i)}(\mathbf{x}_k^{(i)})\|^2. \tag{3}$$

After we learn a map $\mathbf{D}_{k+1}$, we update each $\mathbf{x}_k^{(i)}$ using (2), then proceed to learn the next map. This process is repeated until some terminating conditions, such as until the error does not decrease much, or until a maximum number of iterations. To see why (3) learns stationary points, we can see that for i with $\mathbf{x}_k^{(i)} \approx \mathbf{x}_*^{(i)}$, (3) will force $\mathbf{D}\mathbf{h}^{(i)}(\mathbf{x}_k^{(i)})$ to

[2] Bold capital letters denote a matrix $\mathbf{X}$, bold lower-case letters a column vector $\mathbf{x}$. All non-bold letters represent scalars. $\mathbf{0}_n \in \mathbb{R}^n$ is the vector of zeros. Vector $\mathbf{x}_i$ denotes the i^{th} column of $\mathbf{X}$. Bracket subscript $[\mathbf{x}]_i$ denotes the i^{th} element of $\mathbf{x}$. $\|\mathbf{x}\|$ denotes ℓ_2-norm $\sqrt{\mathbf{x}^\top \mathbf{x}}$.

[3] Here, we use linear maps for their computational efficiency, but other non-linear regression functions can be used in a straightforward manner.

Algorithm 1 Training a sequence of update maps (SUM)

Input: $\{(\mathbf{x}_0^{(i)}, \mathbf{x}_*^{(i)}, \mathbf{h}^{(i)})\}_{i=1}^N, K, \lambda$
Output: $\{\mathbf{D}_k\}_{k=1}^K$

for $k = 0$ **to** $K-1$ **do**
  Compute $\mathbf{D}_{k+1}$ with (4).
  for $i = 1$ **to** N **do**
    Update $\mathbf{x}_{k+1}^{(i)} := \mathbf{x}_k^{(i)} - \mathbf{D}_{k+1}\mathbf{h}^{(i)}(\mathbf{x}_k^{(i)})$.
  end for
end for

Algorithm 2 Searching for a stationary point

Input: $\mathbf{x}_0, \mathbf{h}, \{\mathbf{D}_k\}_{k=1}^K, maxIter, \epsilon$
Output: $\mathbf{x}$

Set $\mathbf{x} := \mathbf{x}_0$
for $k = 1$ **to** K **do**
  Update $\mathbf{x} := \mathbf{x} - \mathbf{D}_k\mathbf{h}(\mathbf{x})$
end for
Set $iter := K + 1$.
while $\|\mathbf{D}_K\mathbf{h}(\mathbf{x})\| \geq \epsilon$ **and** $iter \leq maxIter$ **do**
  Update $\mathbf{x} := \mathbf{x} - \mathbf{D}_K\mathbf{h}(\mathbf{x})$
  Update $iter := iter + 1$
end while

be close to zero, thereby inducing a stationary point around $\mathbf{x}_*^{(i)}$. In practice, to prevent overfitting, ridge regression is used to learn the maps:

$$\min_{\tilde{\mathbf{D}}} \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_*^{(i)} - \mathbf{x}_k^{(i)} + \tilde{\mathbf{D}}\mathbf{h}^{(i)}(\mathbf{x}_k^{(i)})\|^2 + \frac{\lambda}{2}\|\tilde{\mathbf{D}}\|_F^2, \quad (4)$$

where λ is a hyperparameter. The pseudocode for training a SUM is shown in Alg. 1.

Solving a new problem instance: When solving a new problem instance, we are given an unseen function $\mathbf{h}$ and an initialization $\mathbf{x}_0$. To solve this new problem instance, we update $\mathbf{x}_k, k = 0, 1, \ldots$ with the obtained SUM using (2) until a stationary point is reached. However, in practice, the number of maps is finite, say K maps. We observed that many times the update at the K^{th} iteration is still large, which means the stationary point is still not reached, and also the result parameter $\mathbf{x}_K$ is far from the true solution. For the registration task, this is particularly the problem when there is a large rotation angle between the initialization and the solution. To overcome this problem, we keep updating $\mathbf{x}$ using the K^{th} map until the update is small or the maximum number of iterations is reached. This approach makes DO different from previous works in Sec. 2.2, where the updates are only performed up to the number of maps. Alg. 2 shows the pseudocode for updating the parameters.

3.3. Theoretical analysis

This section provides theoretical analysis for DO. Specifically, we show that under a weak assumption on $\mathbf{h}^{(i)}$, it is possible to learn a SUM that strictly decreases training error in each iteration. First, we define the *monotonicity at a point* condition:

Definition 1 *(Monotonicity at a point) A function $\mathbf{f} : \mathbb{R}^p \to \mathbb{R}^p$ is monotone at a point $\mathbf{x}_* \in \mathbb{R}^p$ if it satisfies $(\mathbf{x} - \mathbf{x}_*)^\top \mathbf{f}(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^p$. $\mathbf{f}$ is strictly monotone if the equality holds only at $\mathbf{x} = \mathbf{x}_*$.*[4]

With the above definition, we can show the following result:

Theorem 1 (Strict decrease in training error under a sequence of update maps (SUM)) *Given a training set $\{(\mathbf{x}_0^{(i)}, \mathbf{x}_*^{(i)}, \mathbf{h}^{(i)})\}_{i=1}^N$, if there exists a linear map $\hat{\mathbf{D}} \in \mathbb{R}^{p\times f}$ such that, for each i, $\hat{\mathbf{D}}\mathbf{h}^{(i)}$ is strictly monotone at $\mathbf{x}_*^{(i)}$, and if $\exists i : \mathbf{x}_k^{(i)} \neq \mathbf{x}_*^{(i)}$, then the update rule:*

$$\mathbf{x}_{k+1}^{(i)} = \mathbf{x}_k^{(i)} - \mathbf{D}_{k+1}\mathbf{h}^{(i)}(\mathbf{x}_k^{(i)}), \quad (5)$$

with $\mathbf{D}_{k+1} \in \mathbb{R}^{p\times f}$ obtained from (3), guarantees that the training error strictly decreases in each iteration:

$$\sum_{i=1}^{N} \|\mathbf{x}_*^{(i)} - \mathbf{x}_{k+1}^{(i)}\|^2 < \sum_{i=1}^{N} \|\mathbf{x}_*^{(i)} - \mathbf{x}_k^{(i)}\|^2. \quad (6)$$

The proof of Thm. 1 is provided in the supplementary material. In words, Thm. 1 says that if each instance i is similar in the sense that each $\hat{\mathbf{D}}\mathbf{h}^{(i)}$ is strictly monotone at $\mathbf{x}_*^{(i)}$, then sequentially learning the optimal maps with (3) guarantees that training error strictly decreases in each iteration. Note that $\mathbf{h}^{(i)}$ is not required to be differentiable or continuous. The SDM theorem [24] also presents a convergence result for a similar update rule, but it shows the convergence of a *single* function under a *single ideal* map. It also requires an additional condition called 'Lipschitz at a point.' This condition is necessary for bounding the norm of the map, otherwise the update can be too large, preventing the convergence to the solution. In contrast, Thm. 1 explains the convergence of *multiple* functions under the same SUM learned from the data, where the learned maps $\mathbf{D}_k$ can be different from the ideal map $\hat{\mathbf{D}}$. Thm. 1 also does not require the 'Lipschitz at a point' condition to bound the norms of the maps since they are adjusted based on the training data. Not requiring this Lipschitz condition has an important implication as it allows robust discontinuous features, such as HOG [24], to be used as $\mathbf{h}^{(i)}$. In this work, we will also propose a discontinuous function $\mathbf{h}$ for point-cloud registration. Lastly, we wish to point out that Thm. 1 does not guarantee that the error of each instance i reduces in each iteration, but guarantees the reduction in the average error.

[4] The strict version is equivalent to the one used in the proof in [24].

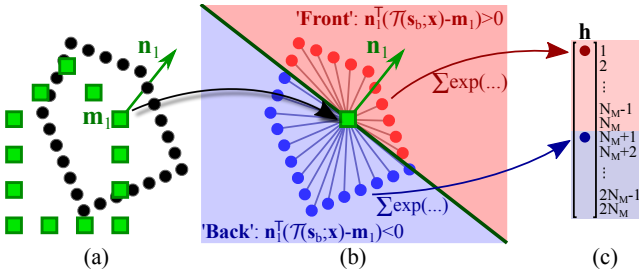


Figure 3. Feature $\mathbf{h}$ for registration. (a) Model points (square) and scene points (circle). (b-c) Weights of $\mathbf{s}_b$ that are on the 'front' or 'back' of model point $\mathbf{m}_1$ are assigned to different indices in $\mathbf{h}$.

4. DO for Point Cloud Registration

This section describes how to apply DO to register point clouds under rigidity. For simplicity, this section discusses the case for registering two 3D shapes with different numbers of points, but the idea can be simply extended to 2D cases. Let $\mathbf{M} \in \mathbb{R}^{3\times N_M}$ be a matrix containing 3D coordinates of one shape ('model') and $\mathbf{S} \in \mathbb{R}^{3\times N_S}$ for the second shape ('scene'). Our goal is to find the rotation and translation that registers $\mathbf{S}$ to $\mathbf{M}$[5]. Recall that the correspondence between points in $\mathbf{S}$ and $\mathbf{M}$ is unknown.

4.1. Parametrization of the transformations

Rigid transformations are usually represented in matrix form with nonlinear constraints. Since DO does not admit constraints, it is inconvenient to parametrize the transformation parameter $\mathbf{x}$ in such matrix form. However, the matrix representation of rigid transformation forms a Lie group, which associates with a Lie algebra [12, 15]. In essence, the Lie algebra is a linear vector space with the same dimensions as the degrees of freedom of the transformation; for instance, $\mathbb{R}^6$ is the Lie algebra of the 3D rigid transformation. Each element in the Lie algebra is associated with an element in the Lie group via exponential and logarithm maps, where closed form computations exists (provided in supplementary material). Being a linear vector space, Lie algebra provides a convenient parametrization for $\mathbf{x}$ since it requires no constraints to be enforced. Note that multiple elements in the Lie algebra can represent the same transformation in Lie group, *i.e.*, the relation is not one-to-one. However, the relation is one-to-one locally around the origin of the Lie algebra, which is sufficient for our task. Previous works that use Lie algebra include motion estimation and tracking in images [2, 21].

4.2. Features for registration

The function $\mathbf{h}$ encodes information about the problem to be solved, *e.g.*, it extracts features from the input data.

[5]The transformation that register $\mathbf{M}$ to $\mathbf{S}$ can be found by inversing the transformation that registers $\mathbf{S}$ to $\mathbf{M}$.

For point cloud registration, we observe that most shapes of interest are comprised of points that form a surface, and good registration occurs when the surfaces of the two shapes are aligned. To align surfaces of points, we design $\mathbf{h}$ to be a histogram that indicates the weights of scene points on the 'front' and the 'back' sides of each model point (see Fig. 3). This allows DO to learn the parameters that update the point cloud in the direction that aligns the surfaces. Let $\mathbf{n}_a \in \mathbb{R}^3$ be a normal vector of the model point $\mathbf{m}_a$ computed from neighboring points; $\mathcal{T}(\mathbf{y};\mathbf{x})$ be a function that applies rigid transformation with parameter $\mathbf{x}$ to vector $\mathbf{y}$; $S_a^+ = \{\mathbf{s}_b : \mathbf{n}_a^\top(\mathcal{T}(\mathbf{s}_b;\mathbf{x}) - \mathbf{m}_a) > 0\}$ be the set of scene points on the 'front' of $\mathbf{m}_a$; and S_a^- contains the remaining scene points. We define $\mathbf{h} : \mathbb{R}^6 \times \mathbb{R}^{3\times N_S} \to \mathbb{R}^{2N_M}$ as:

$$[\mathbf{h}(\mathbf{x};\mathbf{S})]_a = \frac{1}{z}\sum_{\mathbf{s}_b\in S_a^+} \exp\left(-\frac{1}{\sigma^2}\|\mathcal{T}(\mathbf{s}_b;\mathbf{x}) - \mathbf{m}_a\|^2\right), \quad (7)$$

$$[\mathbf{h}(\mathbf{x};\mathbf{S})]_{a+N_M} = \frac{1}{z}\sum_{\mathbf{s}_b\in S_a^-} \exp\left(-\frac{1}{\sigma^2}\|\mathcal{T}(\mathbf{s}_b;\mathbf{x}) - \mathbf{m}_a\|^2\right), \quad (8)$$

where z normalizes $\mathbf{h}$ to sum to 1, and σ controls the width of the exp function. The exp term calculates the weight depending on the distance between the model and the scene points. The weight due to $\mathbf{s}_b$ is assigned to index a or $a + N_M$ depending on the side of $\mathbf{m}_a$ that $\mathbf{s}_b$ is on. Note that $\mathbf{h}$ is specific to a model $\mathbf{M}$, and it returns a fixed length vector of size $2N_M$. This is necessary since $\mathbf{h}$ is to be multiplied to $\mathbf{D}_k$, which are fixed size matrices. Thus, *the SUM learned is also specific to the shape* $\mathbf{M}$. However, $\mathbf{h}$ can take the scene shape $\mathbf{S}$ with an arbitrary number of points to use with the SUM. Although we do not prove that this $\mathbf{h}$ complies with the condition in Thm. 1, we show empirically in Sec. 5 that it can be effectively used for our task.

4.3. Fast computation of feature

Empirically, we found that computing $\mathbf{h}$ directly is slow due to pairwise distance computations and the evaluation of exponentials. To perform fast computation, we quantize the space around the model shape into uniform grids, and store the value of $\mathbf{h}$ evaluated at the center of each grid. When computing features for a scene point $\mathcal{T}(\mathbf{s}_b;\mathbf{x})$, we simply return the precomputed feature of the grid center that is closest to $\mathcal{T}(\mathbf{s}_b;\mathbf{x})$. Note that since the grid is uniform, finding the closest grid center can be done in $\mathcal{O}(1)$. To get a sense of scale in this section, we assume the model is mean-subtracted and normalized so that the largest dimension is in $[-1, 1]$. We compute the uniform grid in the range $[-2, 2]$ with 81 points in each dimension. We set any elements of the precomputed features that are smaller than 10^{-6} to 0, and since most of the values are zero, we store them in a sparse matrix. We found that this approach significantly reduces the feature computation time by 6 to 20 times while maintaining the same accuracy. In our experiments, the precomputed features require less than 50MB for each shape.

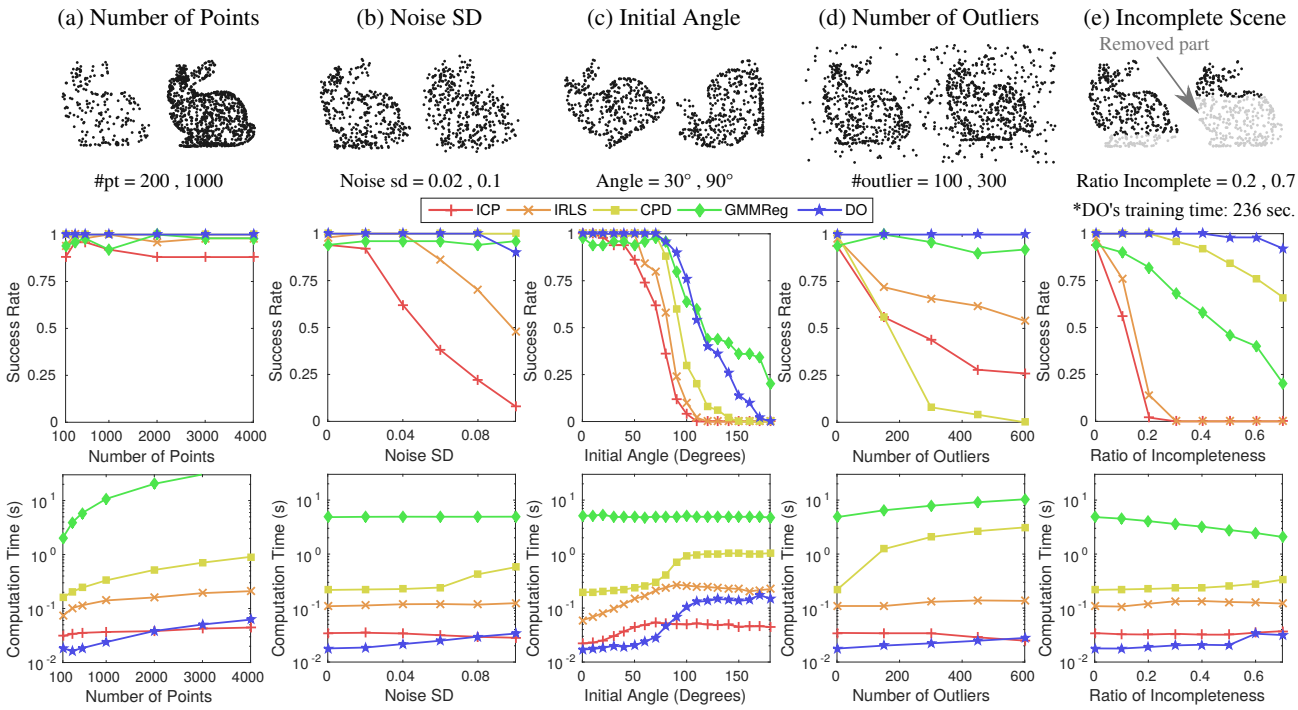


Figure 4. Results of 3D registration with synthetic data under different perturbations. (Top) Examples of scene points with different perturbations. (Middle) Success rate. (Bottom) Computation time.

5. Experiments

This section provides experimental results in 3D data with both synthetic and range-scan data. The experimental results for 2D cases are provided in the supplementary document. All experiments were performed on a single thread on an Intel i7-4790 3.60GHz computer with 16GB memory.

Baselines: We compared DO with two point-based approaches (ICP [4] and IRLS [3]) and two density-based approaches (CPD [17] and GMMReg [14]). The codes for all methods were downloaded from the authors' websites, except for ICP where we used MATLAB's implementation. For IRLS, the Huber cost function was used. The code of DO was implemented in MATLAB.

Performance metrics: We used the registration success rate and the computation time as performance metrics. We considered a registration to be successful when the mean ℓ_2 error between the registered model points and the corresponding model points at the ground truth orientation was less than 0.05 of the model's largest dimension.

Training the DO algorithms: Given a model shape $\mathbf{M}$, we first normalized it to lie in $[-1, 1]$, and generated the scene models for training by uniformly sampling with replacement 400 to 700 points from $\mathbf{M}$. Then, we applied the following three types of perturbations: *(1) Rotation and translation:* We randomly rotated the model within 85 degrees, and added a random translation in $[-0.3, 0.3]^3$. These transformations were used as the ground truth $\mathbf{x}_*$ in (4), with $\mathbf{x}_0 = \mathbf{0}_6$ as the initialization. *(2) Noise and outliers:* Gaussian noise with standard deviation 0.05 was added to the sample. We considered two types of outliers. First, sparse outliers of 0 to 300 points were added within $[-1, 1]^3$. Second, structured outliers were simulated with a Gaussian ball of 0 to 200 points with the standard deviation of 0.1 to 0.25. This created a group of dense points that mimic other objects in the scene. *(3) Incomplete shape:* We used this perturbation to simulate self occlusion and occlusion by other objects. This was done by removing points on one side of the model. Specifically, we uniformly sampled a 3D unit vector $\mathbf{u}$, then we projected all sample points to $\mathbf{u}$, and removed the points with the top 40% to 80% of the projected values. For all experiments, we generated 30000 training samples, and trained a total of $K = 30$ maps for SUM with $\lambda = 2 \times 10^{-4}$ in (4) and $\sigma^2 = 0.03$ in (7) and (8), and set the maximum number of iterations to 1000.

5.1. Synthetic data

We performed synthetic experiments using the Stanford Bunny model [19] (see Fig. 4). The complete model contains 36k points. We used MATLAB's `pcdownsample` to select 472 points as the model $\mathbf{M}$. We evaluated the performance of the algorithms by varying five types of perturbations: (1) the number of scene points ranges from 100 to 4000 [default = 200 to 600]; (2) the standard deviation of the noise ranges between 0 to 0.1 [default = 0]; (3) the initial angle from 0 to 180 degrees [default = 0 to 60]; (4) the number

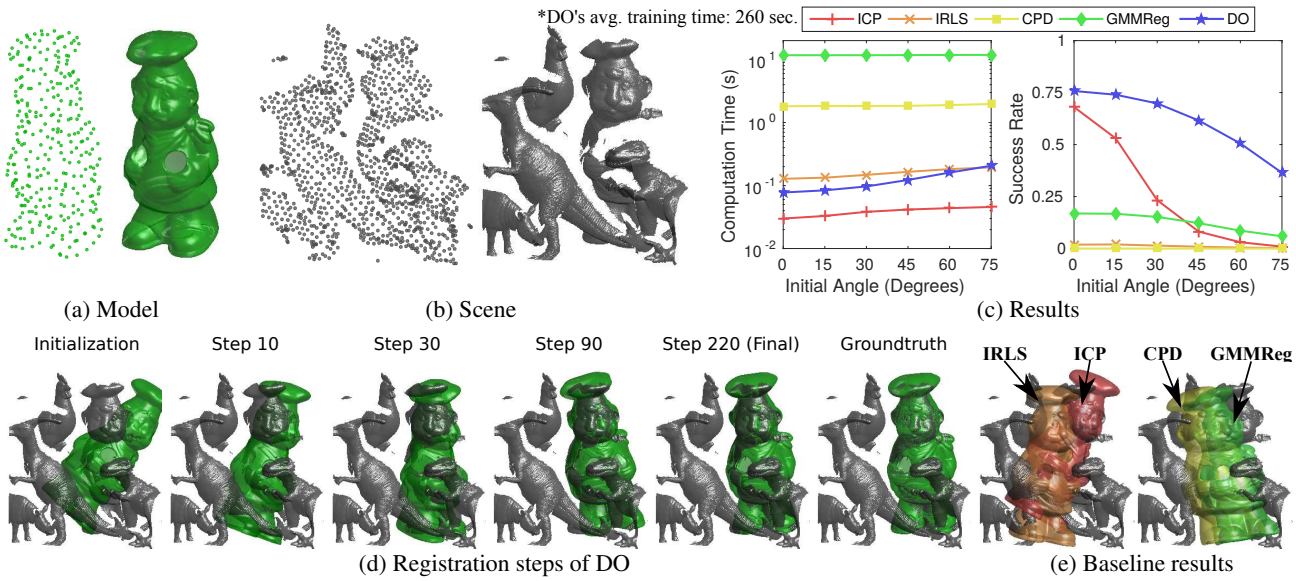


Figure 5. Results of 3D registration with range scan data. (a) shows a 3D model ('chef'), and (b) shows an example of a 3D scene. In addition to the point clouds, we include surface rendering for visualization purpose. (c) shows the results of the experiment. (d) shows an example of registration steps of DO. The model was initialized 60 degrees from the ground truth orientation with parts of the model intersecting other objects. In addition, the target object is under 70% occlusion, making this a very challenging case. However, as iteration progresses, DO is able to successfully register the model. (e) shows the results of baseline algorithms.

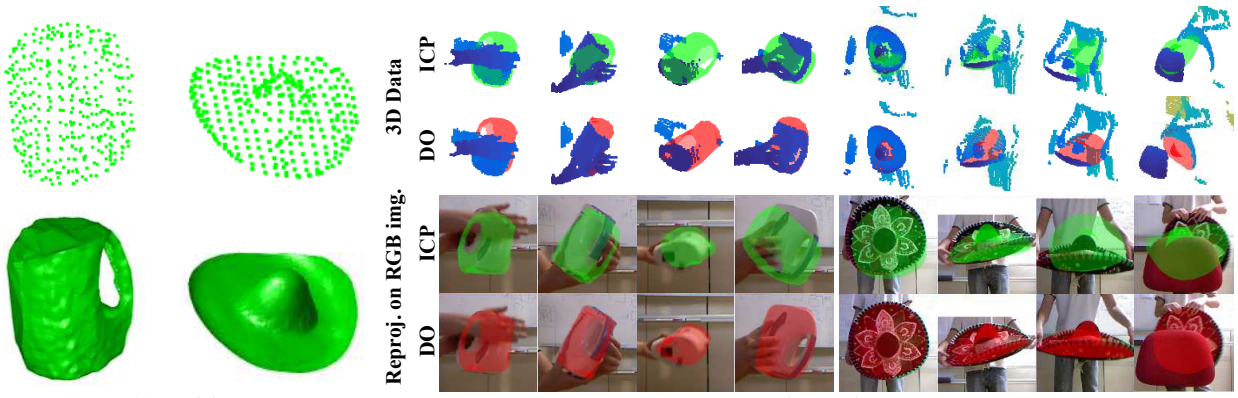
of outliers from 0 to 600 [default = 0]; and (5) the ratio of incomplete scene shape from 0 to 0.7 [default = 0]. While we perturbed one parameter, the values of the other parameters were set to the default values. Note that the scene points were sampled from the original 36k points, not from M. The range for the outliers was $[-1.5, 1.5]^3$. All generated scenes included random translation within $[-0.3, 0.3]^3$. A total of 50 rounds were run for each variable setting. Training time for DO took 236 seconds (including generating the training data and pre-computing features).

Examples of test data and the results are shown in Fig. 4. While ICP required low computation time for all cases, it had low success rates when the perturbations were high. This is because ICP tends to get trapped in the local minimum closest to its initialization. CPD performed well in all cases except when the number of outliers was high, and it required a high computation time. IRLS was faster than CPD; however, it did not perform well when the model was highly incomplete. GMMReg had the widest basin of convergence but did not perform well with incomplete shapes. It also required long computation time due to the annealing steps. For DO, its computation time was much lower than those of the baselines. Notice that DO required higher computation time for larger initial angles since more iterations were required to reach a stationary point. In terms of the success rate, we can see that DO outperformed the baselines in almost all test scenarios. This result was achievable because DO does not rely on any specific cost functions, which generally are modelled to handle a few types of perturbations. On the other hand, DO *learns* to cope with the perturbations from training data, allowing it to be significantly more robust than other approaches.

5.2. Range-scan data

In this section, we performed 3D registration experiment on the UWA dataset [16]. This dataset contains 50 cluttered scenes with 5 objects taken with the Minolta Vivid 910 scanner in various configurations. All objects are heavily occluded (60% to 90%). We used this dataset to test our algorithm under unseen test samples and structured outliers, as opposed to sparse outliers in the previous section. The dataset includes 188 ground truth poses for four objects. We performed the test using all the four objects on all 50 scenes. From the original model, $\sim$300 points were sampled using `pcdownsample` and used as the model M (Fig. 5a). We also downsampled each scene to $\sim$1000 points (Fig. 5b). We initialized the model from 0 to 75 degrees from the ground truth orientation with random translation within $[-0.4, 0.4]^3$. We ran 50 initializations for each parameter setting, resulting in a total of 50×188 rounds for each data point. Here, we set the inlier ratio of ICP to 50% as an estimate for self-occlusion. Average training time for DO was 260 seconds for each object model.

The results and examples for the registration with DO are shown in Fig. 5c and Fig. 5d, respectively. IRLS, CPR, and GMMReg has very low success in almost every scene. This was because structured outliers caused many regions to have high density, creating false optima for CPD and GMM-



(a) Models (b) Results

Figure 6. Result for object tracking in 3D point cloud. (a) shows the 3D models of the kettle and the hat. (b) shows tracking results of DO and ICP in (top) 3D point clouds with the scene points in blue, and (bottom) as reprojection on RGB image. Each column shows the same frame. (See supplementary video).

Reg which are density-based approaches, and also for IRLS which is less sensitive to local minima than ICP. When initialized close to the solution, ICP could register fast and provided some correct results because it typically terminated at the nearest–and correct–local minimum. On the other hand, DO provided a significant improvement over ICP, while maintaining low computation time. *We emphasize that DO was trained with synthetic examples of a single object and it had never seen other objects from the scenes*. This experiment shows that we can train DO with synthetic data, and apply it to register objects in real challenging scenes.

5.3. Application to 3D object tracking

In this section, we explore the use of DO for 3D object tracking in 3D point clouds. We used Microsoft Kinect to capture videos of RGB and depth images at 20fps, then reconstruct 3D scenes from the depth images. We performed the test with two reconstructed shapes as the target objects: a kettle and a hat. In addition to self-occlusion, both objects presented challenging scenarios: the kettle has an overall smooth surface with few features, while the hat is flat, making it hard to capture from some viewpoints. During recording, the objects went through different orientations, occlusions, etc. For this experiment, we subsampled the depth images to reduce the input points at every 5 pixels for the kettle's videos and 10 pixels for the hat's. To perform tracking, we manually initialized the first frame, while subsequent frames were initialized using the pose in the previous frames. No color from RGB images was used. Here, we only compared DO against ICP because IRLS gave similar results to those of ICP but could not track rotation well, while CPD and GMMReg failed to handle structured outliers in the scene (similar to Sec. 5.2). Fig. 6b shows examples of the results. It can be seen that DO can robustly track and estimate the pose of the objects accurately even under heavy occlusion and structured outliers, while ICP tended to get stuck with other objects. The average computation time for DO was 40ms per frame. This shows that DO can be used as a robust real-time object tracker in 3D point cloud. The result videos are provided as supplementary material.

Failure case: We found that DO failed to track the target object in some cases, such as: (1) when the object was occluded at an extremely high rate, and (2) when the object moved too fast. When this happened, DO would either track another object that replaced the position of the target object, or simply stay at the same position in the previous frame.

6. Conclusions

This paper proposes discriminative optimization (DO), a methodology to solve parameter estimation in computer vision by learning update directions from training examples. Major advantages of DO over traditional methods include robustness to noise and perturbations, as well as efficiency. We provided theoretical result on the convergence of the training data under mild conditions. In terms of application, we demonstrated the potential of DO in the problem of 2D and 3D point cloud registration under rigidity, and illustrated that it outperformed state-of-the-art approaches.

Future work of interest is to design a feature function that is not specific to a single model, which would allow two input shapes to register without training a new sequence of maps. Beyond 2D/3D point cloud registration, we believe DQ could be applied to a much wider range of problems in computer vision, such as non-rigid registration, camera calibration, or fitting shape models to videos.

Acknowledgment We would like to thank Susana Brandao for providing 3D model of the kettle. This research was supported in part by Fundação para a Ciência e a Tecnologia (project FCT [UID/EEA/50009/2013] and a PhD grant from the Carnegie Mellon-Portugal program), the National Science Foundation under the grants RI-1617953, and the EU-Horizon 2020 project #731667 (MULTIDRONE). The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies.

References

[1] E. Antonakos, P. Snape, G. Trigeorgis, and S. Zafeiriou. Adaptive cascaded regression. In *ICIP*, 2016. 3

[2] E. Bayro-Corrochano and J. Ortegn-Aguilar. Lie algebra approach for tracking and 3D motion estimation using monocular vision. *Image and Vision Computing*, 25(6):907921, 2007. 5

[3] P. Bergström and O. Edlund. Robust registration of point sets using iteratively reweighted least squares. *Computational Optimization and Applications*, 58(3):543–561, 2014. 2, 6

[4] P. J. Besl and H. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 2, 6

[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. 3

[6] D. Campbell and L. Petersson. An adaptive data representation for robust point-set registration and merging. In *ICCV*, 2015. 3

[7] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. In *CVPR*, 2012. 3

[8] P. Dollár, P. Welinder, and P. Perona. Cascaded pose regression. In *CVPR*, 2010. 3

[9] A. Fitzgibbon. Robust registration of 2D and 3D point sets. In *BMVC*, 2001. 2

[10] S. Gold, A. Rangarajan, C.-P. Lu, P. Suguna, and E. Mjolsness. New algorithms for 2D and 3D point matching: pose estimation and correspondence. *Pattern Recognition*, 38(8):1019–1031, 1998. 2

[11] V. Golyanik, S. Aziz Ali, and D. Stricker. Gravitational approach for point set registration. In *CVPR*, 2016. 3

[12] B. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. Springer, 2004. 5

[13] M. Harker and P. O'Leary. Least squares surface reconstruction from measured gradient fields. In *CVPR*. 2

[14] B. Jian and B. C. Vemuri. Robust point set registration using gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1633–1645, 2011. 2, 3, 6

[15] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An Invitation to 3-D Vision*. Springer-Verlag New York, 2004. 5

[16] A. Mian, M. Bennamoun, and R. Owens. On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes. *IJCV*, 89(2):348–361, 2010. 7

[17] A. Myronenko and X. Song. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, 2010. 2, 6

[18] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. 3rd Int. Conf. 3-D Digital Imaging and Modeling*, 2001. 2

[19] Stanford Computer Graphics Laboratory. The stanford 3D scanning repository. https://graphics.stanford.edu/data/3Dscanrep/, Aug 2014. Accessed: 2016-08-31. 6

[20] Y. Tsin and T. Kanade. A correlation-based approach to robust point set registration. In *ECCV*, 2004. 2

[21] O. Tuzel, F. Porikli, and P. Meer. Learning on lie groups for invariant detection and tracking. In *CVPR*, 2008. 5

[22] G. Tzimiropoulos. Project-out cascaded regression with an application to face alignment. In *CVPR*, 2015. 3

[23] X. Xiong and F. De la Torre. Supervised descent method and its application to face alignment. In *CVPR*, 2013. 3

[24] X. Xiong and F. De la Torre. Supervised descent method for solving nonlinear least squares problems in computer vision. *CoRR*, abs/1405.0601, 2014. 3, 4

[25] X. Xiong and F. De la Torre. Global supervised descent method. In *CVPR*, 2015. 3