The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Iterative Continuous Convolution for 3D Template Matching and Global Localization

Vitor Guizilini, Fabio Ramos
School of Information Technologies, The University of Sydney, Australia
e-mail: {vitor.guizilini@;fabio.ramos}sydney.edu.au

Abstract

This paper introduces a novel methodology for 3D template matching that is scalable to higher-dimensional spaces and larger kernel sizes. It uses the Hilbert Maps framework to model raw pointcloud information as a continuous occupancy function, and we derive a closed-form solution to the convolution operation that takes place directly in the Reproducing Kernel Hilbert Space defining these functions. The result is a third function modeling activation values, that can be queried at arbitrary resolutions with logarithmic complexity, and by iteratively searching for high similarity areas we can determine matching candidates. Experimental results show substantial speed gains over standard discrete convolution techniques, such as sliding window and fast Fourier transform, along with a significant decrease in memory requirements, without accuracy loss. This efficiency allows the proposed methodology to be used in areas where discrete convolution is currently infeasible. As a practical example we explore the key problem in robotics of global localization, in which a vehicle must be positioned on a map using only its current sensor information, and provide comparisons with other state-of-the-art techniques in terms of computational speed and accuracy.

1 Introduction

Template matching is a technique used to identify areas in a dataset that are similar to a given pattern. It has a wide range of applications in several fields of research, most predominantly image processing (Brunelli 2009), where it is used in tasks such as scene recognition, object classification, autonomous navigation and so forth. Broadly speaking, most template matching techniques either rely on data preprocessing, using predefined models (Opromolla et al. 2015) or salient features (Steder, Grisetti, and Burgard 2010) to represent information; or work directly on raw input data, using cross-correlation techniques (Briechle and Hanebeck 2001) to measure similarity.

Although generally more attractive, since it does not require preprocessing or prior assumptions about input data, cross-correlation techniques still struggle with scaling limitations, especially in higher-dimensional scenarios. Recent evolutions in sensor technology have led to a jump from two to three-dimensional datasets, that are more representative of real world structures and environments than a single 2D slice. Unfortunately, the computational complexity of cross-correlation grows exponentially in relation to the number of input dimensions, rendering 3D calculations very time-consuming. Moreover, due to the discrete nature of digital data, the exact formulation is often substituted by a discrete approximation, that even though much simpler requires substantially more computational power and memory. Several different approaches have been proposed over the years to alleviate this drawback, such as separable kernels (Hummel and Lower 1986), recursive filtering (Bernd 2005) and transformation to the frequency domain (Bracewell 2003). However, these approaches are mostly circumstantial and cannot be applied in every scenario: not all kernels are separable; recursive filtering often suffers from inaccuracy and instability; and calculations in the frequency domain have considerable more memory requirements and lower numerical precision.

Cross-correlation shares many similarities with convolution, an equally important mathematical tool with applications in areas such as filtering (Jan 2000), denoising (Vaseghi 2006), compression (Salomon 2006) and deconvolution (Castleman 1996). In order to extend its use to higher-dimensional spaces, a substantial amount of work has been done to increase efficiency in the convolution formulation: in (Riegler, Ulusoy, and Geiger 2016), an octotree is used to exploit sparsity in input data to hierarchically partition the space, creating a varying-size voxel grid that avoids wasteful computations and decreases memory requirements. This framework is then used to train a convolutional neural network for tasks such as object classification, orientation estimation and point cloud labeling. Similarly, (Li et al. 2016) represents 3D space as volumetric fields and proposes the use of field probing filters to efficiently extract features from sparse input data.

All these approaches, however, are still discrete, and very little attention has been paid over the years to the application of continuous convolution to similar tasks. This paper proposes a novel methodology that enables the use of continuous convolution between sparse 3D pointclouds[1], delivering the result as a function that can be queried at arbitrary resolutions and at logarithmic time, thus allowing for very efficient

[1]Even though this paper focuses on 3D data, the proposed methodology can be trivially extended to any input dimensionality.

searches (i.e. for local extrema). We employ the Hilbert Maps framework (Guizilini and Ramos 2016), an occupancy mapping technique that uses feature vectors to project observations into a high-dimensional Reproducing Kernel Hilbert Space, where real-world complexity can be represented in a linear fashion.

The main contribution of this paper is a technique for the continuous convolution between two functions, and how it can be applied in an occupancy mapping context for efficient cross-correlation template matching. We show that the proposed technique is substantially faster than standard convolution algorithms and has a much smaller memory footprint, since it does not require space discretization in any form. Particularly, we show that our algorithm scales to much larger kernel sizes and is more robust to the presence of sparse information, a known issue found in most real datasets. This methodology can be directly applied to a wide range of other problems that benefit from the convolution operation, such as object classification, image filtering, convolutional neural networks, and so forth. As a practical application, we focus on the problem of global localization (Xie et al. 2010), showing how the proposed methodology is able to achieve results that outperform current state-of-art similar techniques in terms of speed, accuracy and reliability.

2 Occupancy Mapping Formulation

Initially proposed in (Ramos and Ott 2015), the Hilbert Maps (HM) framework is a novel occupancy mapping technique in which real-world complexity is represented linearly by projecting observations into a feature vector that operates on a higher-dimensional space, known as the Reproducing Kernel Hilbert Space (RKHS) (Schölkopf and Smola 2001). This results in a probabilistic framework in which parameter training can be performed very efficiently using stochastic gradient descent methods, and inference can be performed at arbitrary resolutions.

We start by assuming a dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$, in which $\mathbf{x}_i \in \mathcal{R}^D$ is a point in the D-dimensional space and $y_i = \{-1, +1\}$ is its corresponding occupancy value. This dataset is used to incrementally learn a discriminative model $p(y|\mathbf{x}, \mathbf{w})$, parametrized by a weight vector $\mathbf{w}$ that predicts the occupancy values of new query points $\mathbf{x}_*$. A simple Logistic Regression (LR) classifier (Freedman 2005) is used here, due to its computational speed and direct extension to online learning. Note that, even though this classifier is fairly simple, it is accepted that linear separators are almost always adequate to separate classes in high-dimensional spaces (Komarek 2004). The probability of non-occupancy for a query point $\mathbf{x}_*$ is given by:

$$p(y_* = -1|\Phi(\mathbf{x}_*), \mathbf{w}) = \frac{1}{1 + \exp\left(\mathbf{w}^T \Phi(\mathbf{x}_*)\right)}, \quad (1)$$

where $\Phi(\mathbf{x}_*)$ is the feature vector of $\mathbf{x}_*$ that operates on the RKHS. To optimize the weight parameters $\mathbf{w}$ based on information contained in $\mathcal{D}$, we minimize the Regularized Negative Log-Likelihood (RNLL) function:

$$RNLL(\mathbf{w}) = \sum_{i=1}^{N} \left(1 + \exp\left(-y_i \mathbf{w}^T \Phi(\mathbf{x}_i)\right)\right) + R(\mathbf{w}), \quad (2)$$

with $R(\mathbf{w})$ being a regularization function, used to prevent overfitting and promote sparseness in $\mathbf{w}$. A particularly interesting property of Eq. 2 is its suitability for stochastic gradient descent optimization (Bottou 2010), in which the information contained in each training point provides one small step towards a local minimum, given by:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t \frac{\delta}{\delta \mathbf{w}} RNNL(\mathbf{w}), \quad (3)$$

where $\eta > 0$ is the learning rate, usually a constant or asymptotically decaying with the number of iterations. The only remaining question is the feature vector calculation, that projects observed points into the RKHS. It has been shown (Ramos and Ott 2015) that the dot product of these feature vectors can approximate popular kernels used in the literature, i.e. $\Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j) \approx k(\mathbf{x}_i, \mathbf{x}_j)$. Here we are inspired by the work of (Guizilini and Ramos 2016), in which $\mathcal{D}$ is clustered to produce a set $\mathcal{M} = \{\boldsymbol{\mu}_i, \Sigma_i\}_{i=1}^{M}$ of anchor points, used to correlate different areas of the input space based on a Gaussian distribution with Automatic Relevance Determination (i.e. different length-scales for each dimension):

$$k(\mathbf{x}, \mathcal{M}_i) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_i|}} \exp\left(-\frac{1}{2} \mathbf{d}^T \Sigma_i^{-\frac{1}{2}} \mathbf{d}\right), \quad (4)$$

where $\mathbf{d} = \mathbf{x} - \boldsymbol{\mu}_i$ and D is the input space dimensionality. The feature vector $\Phi(\mathbf{x}, \mathcal{M})$ produced by these anchor points is generated by the concatenation of kernel values produced by all extracted clusters (Eq. 5). To decrease computational requirements, sparsity can be enforced by calculating only the kernels related to a subset of the closest clusters and setting all others to zero,

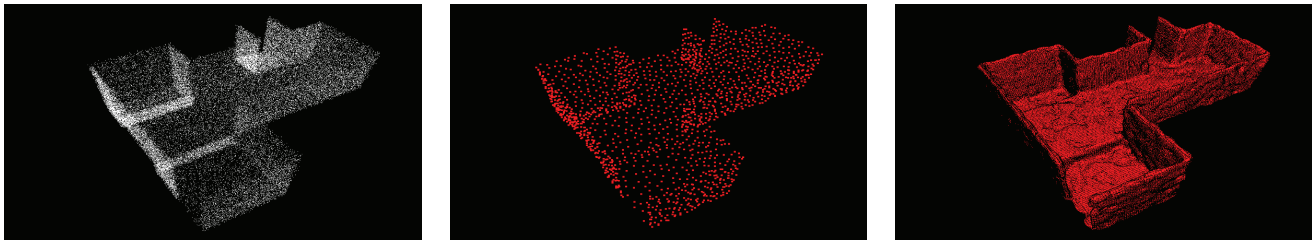
$$\Phi(\mathbf{x}, \mathcal{M}) = \left[k(\mathbf{x}, \mathcal{M}_1)\ ,\ k(\mathbf{x}, \mathcal{M}_2)\ ,\ \ldots\ ,\ k(\mathbf{x}, \mathcal{M}_M)\right]^T. \quad (5)$$

3 Continuous Convolution Model

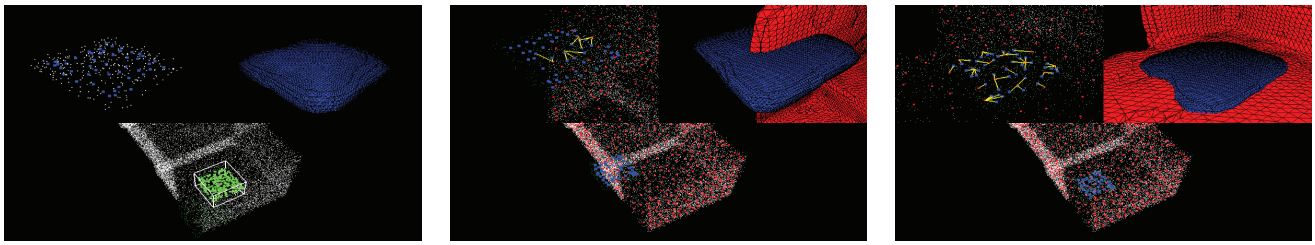
In this section a novel methodology for the continuous convolution between two occupancy state functions $\mathcal{H}^s$ and $\mathcal{H}^m$ is introduced, the first one henceforth referred to as *scene* and the second one as *mask*. The result of this operation is a third function $\mathcal{H}^a$ that does not reflect occupancy states, but rather *activation* values (i.e. how similar the scene is to the mask). Although this methodology can be applied to any two kernel functions with a closed-form convolution formula, or using Monte Carlo approximations (Gilks, Richardson, and Spiegelhalter 1995), for notation simplicity here we assume, without loss of generalization, that both kernel functions are described by Eq. 4. This choice greatly simplifies the problem because, for the convolution of two Gaussian distributions, there is a closed form solution that is also a Gaussian distribution with Automatic Relevance Determination:

$$k(\mathbf{x}, \mathcal{M}_i) * k(\mathbf{x}, \mathcal{M}_j) = k(\mathbf{x}, \mathcal{M}_i + \mathcal{M}_j), \quad (6)$$

where $\mathcal{M}_i + \mathcal{M}_j = \{\boldsymbol{\mu}_i + \boldsymbol{\mu}_j, \Sigma_i + \Sigma_j\}$. The proof of this result is omitted here for brevity (Bromiley 2003), but it is a well-known property of Gaussian distributions. Furthermore, it has been shown (Guizilini and Ramos 2016; Doherty, Wang, and Englot 2016) that the Gaussian kernel



(a) Original pointcloud (b) Extracted clusters (c) 3D Reconstruction



(d) Reconstructed convolution mask. (e) Example of convolution with low activation value (f) Example of convolution with high activation value.

Figure 1: Examples of continuous convolution between two occupancy state functions, $\mathcal{H}^s$ and $\mathcal{H}^m$. In (e) and (f), the yellow lines indicate clusters in $\mathcal{H}^s$ that were closest to clusters in $\mathcal{H}^m$ given the distance threshold (line 9 in Alg. 1). These matches are used to update the weight of $\mathcal{H}^a$ (Eq. 7).

is adequate to efficiently model the occupancy state of large-scale 3D environments at high accuracy. We will now proceed to build upon this property and propose a closed-form solution to the continuous convolution of two arbitrary occupancy state functions, assuming they are modeled using the Hilbert Maps framework with Gaussian kernels, as described by Alg. 1. Examples of convolution results with other kernel functions can be found in (Genton and Kleiber 2015; Gneiting, Kleiber, and Schlather 2010), and directly applied to the proposed methodology to produce more representative models under different circumstances.

In lines 2-3 the resulting function $\mathcal{H}^a$ is initialized as a copy of $\mathcal{H}^s$ and its weights are set to zero. Then, for each cluster $\boldsymbol{\mu}_i^s$, we take all clusters of $\mathcal{H}^m$, translate them so that they are centered in $\boldsymbol{\mu}_i^s$ (line 7) and find their closest neighbor between the clusters in $\mathcal{H}^s$ (line 8)[2]. If the closest neighbor of any particular cluster is closer than the average cluster distance r (line 9), this match is used to update $\mathbf{w}_i$ in $\mathcal{H}^a$ (line 11) according to the following equation:

$$\mathbf{w}_i^a \mathrel{+}= \begin{cases} 0 & \text{if } ||\boldsymbol{\mu}_{ij} - \boldsymbol{\mu}_*^s|| > r \\ \frac{\mathbf{w}_*^s \mathbf{w}_j^m}{\sqrt{(2\pi)^D |\Sigma_{*j}|}} & \text{otherwise,} \end{cases} \tag{7}$$

where $\Sigma_{*j} = \Sigma_*^s + \Sigma_j^m$ and $||\boldsymbol{\mu}_{ij} - \boldsymbol{\mu}_*^s||$ is the Euclidean distance between $\boldsymbol{\mu}_{ij} = \boldsymbol{\mu}_i^s + \boldsymbol{\mu}_j^m$ and its closest neighbor $\boldsymbol{\mu}_*^s$ in $\mathcal{H}^s$. Note the similarities between Eqs. 5 and 7: the weight values of both clusters are multiplied, since they are constant; and the denominator is the same, since the result is also a Gaussian distributions, with covariance matrix equal to the sum of each individual covariance matrix. The only difference is the exponential function, that measures the distance between points in Eq. 5, that is absent in Eq. 7 and is substituted by a binary value: 0 if the distance between clusters is larger than a certain threshold and 1 otherwise. This is due to the random nature of the clustering process. Given the same pointcloud, it is not possible to guarantee that the same cluster centers will be selected, which would create an artificial distance penalty in activation values. By allowing any two clusters within r to be treated as if they were in the same location (i.e. $\mathbf{d} = \mathbf{0}$), we eliminate these random variations, while ensuring that clusters with distance above this threshold are removed from the calculation and do not contribute to the final activation value. The multiplication of cluster weights also reflects the overlapping between occupied (positive weights) and unoccupied areas (negative weights), since only opposite areas will generate a penalty.

Once the convolution process is complete, the resulting function $\mathcal{H}^a$ can be queried using Eq. 4 to produce activation values for any point in the input space. Note that the number of clusters in $\mathcal{H}^a$ remains the same as in $\mathcal{H}^s$, which means that querying for activation values can be done with the same complexity as querying for occupancy values. Examples of this continuous convolution methodology can be seen in Fig. 1e-f, depicting results with low and high activation values, respectively. As expected, areas with higher overlapping will have more cluster matches that fall within the distance threshold, and therefore will generate higher activation values.

Computational Complexity

Here we analyze the computational complexity of the proposed methodology, particularly in relation to the $\mathcal{O}(N_f^3 N_g^3)$ cost of sliding window convolution and

[2]Efficient nearest neighbor search can be performed using approximate kd-trees (Arya et al. 1998).

Algorithm 1 Pseudo-code for the continuous convolution between two Hilbert Maps

1: **Input:** $\mathcal{H}^s$ and $\mathcal{H}^m$ of input dimensionality D and average cluster distance r
2: **Output:** $\mathcal{H}^a$
3: $\mathcal{H}^a = \mathcal{H}^s$ % $\mathcal{H}^a$ is initialized as $\mathcal{H}^s$
4: $\mathbf{w}^a = \mathbf{0}$ % Weights of $\mathcal{H}^a$ are set to zero
5: **for** $i = 0$ **to** M^s **do**
6: **for** $j = 0$ **to** M^m **do**
7: $\boldsymbol{\mu}_{ij} = \boldsymbol{\mu}_i^s + \boldsymbol{\mu}_j^m$
8: $\boldsymbol{\mu}_*^s, \mathbf{w}_*^s = \boldsymbol{\mu}, \mathbf{w}$ for $\boldsymbol{\mu}^s \in \mathcal{M}^s$ closest to $\boldsymbol{\mu}_{ij}$
9: **if** $||\boldsymbol{\mu}_{ij} - \boldsymbol{\mu}_*^s|| < r$ **then**
10: $\Sigma_{*j} = \Sigma_*^s + \Sigma_j^m$
11: $\mathbf{w}_i^a \mathrel{+}= \mathbf{w}_*^s \mathbf{w}_j^m / \sqrt{(2\pi)^D |\Sigma_{*j}|}$
12: **end if**
13: **end for**
14: **end for**

$\mathcal{O}\left((N_f + N_g)^3 log(N_f + N_g)^3\right)$ cost of Fourier convolution for 3D datasets (Fialka and Cadik 2006). This includes both the computational complexity of generating $\mathcal{H}^s$ and $\mathcal{H}^m$ from input data and the convolution process between them to produce $\mathcal{H}^a$.

Recently, a novel clustering initialization method called k-MC^2 was proposed (Bachem et al. 2016) in which seeding for k-means can be done with complexity $\mathcal{O}(K^2 D)$, where K is the number of clusters and D is data dimensionality. This approach completely eliminates dependency on the number N of data points, and was used here to produce the cluster set $\mathcal{M}$ from which the feature vectors in Eq. 5 are generated. To enforce sparsity, only the Q nearest clusters from each data point are selected to produce its feature vector (in all experiments, a value of $Q = 3$ was used). This is *can be* efficiently implemented by maintaining a kd-tree of cluster locations, that can be queried with a computational complexity of $\mathcal{O}(\log M)$, where M is the number of clusters used to describe the pointcloud.

Training the weights of a Hilbert Map with the above information can be performed with complexity $\mathcal{O}(NQ \log M)$ using stochastic gradient descent, in which each data point updates Q weights to better reflect its label (occupied or unoccupied). Once both $\mathcal{H}^s$ and $\mathcal{H}^m$ are obtained, their convolution (see Alg. 1) can be performed with complexity $\mathcal{O}(M_s M_m \log M_s)$, since for each point in M_s all translated points in M_m are convolved with their nearest neighbor in M_s. The resulting function $\mathcal{H}^a$ has the same number of clusters $M_a = M_s$ and can be queried with complexity $\mathcal{O}(Q \log M_s)$. To summarize, the entire proposed continuous convolution process has complexity:

$$\mathcal{O}\big(\mathcal{H}(\mathcal{D}_s, \mathcal{D}_m)\big) = \mathcal{O}\big((M_s^2 + M_m^2)D + NQ(\log M_s + \log M_m) + M_s M_m \log M_s\big), \quad (8)$$

where we can see a linear dependency in the number D of dimensions. Intuitively, the size of M_s and M_m should also increase as more dimensions are added, however in most real datasets this increase is not exponential, since there is usually a large amount of unobserved areas, that do not produce any clusters. The number of input points is considered linearly and only during the training process, which is performed once and can then be reused for any number of convolutions. When $M_m \ll M_s$ the clustering process is in fact the most computationally complex step, with $\mathcal{O}(M_s^2 D)$, whereas at the limit when $M_m = M_s$ the convolution process becomes the most computationally complex step, with $\mathcal{O}(M_s^2 \log M_s)$. Note that memory requirement is proportional to only $\mathcal{O}(M_s + M_m)$, since there is no need to produce a discretized grid of the observed space, only store cluster locations, covariances and weights.

Rotational Invariance

While the standard convolution process is translationally invariant, it still takes into account the orientation of its functions during calculations, and there are situations in which rotational invariance is desirable. For example, one could be interested in detecting particular objects, regardless of their orientation in the environment. The continuous convolution process depicted in Alg. 1 produces high activation values only in areas of $\mathcal{H}^s$ aligned with the length-scales of $\mathcal{H}^m$. In this section we propose a simple, yet effective, technique that automatically produces aligned clusters during the convolution process, so the resulting function $\mathcal{H}^a$ is also invariant to arbitrary rotations between $\mathcal{H}^s$ and $\mathcal{H}^m$.

This alignment is performed by modifying line 8 of Alg. 1, where the clusters of $\mathcal{H}^m$ are iteratively translated to be centered around each cluster in $\mathcal{H}^s$, before proceeding with nearest neighbor search (line 8) and weight update (line 11). This modification takes the form:

$$\boldsymbol{\mu}_{ij} = \boldsymbol{\mu}_i^s + R\boldsymbol{\mu}_j^m \quad , \quad \Sigma_{ij} = R\Sigma_j^m R^T, \quad (9)$$

where $R = [V_\star^m]^T [V_i^s]$ is a rotation matrix composed of V_i^s as the eigenvector matrix of Σ_i^s, sorted by ascending eigenvalue order, and $V_\star^m$ is a similar eigenvector matrix, obtained from the centermost cluster in $\mathcal{H}^m$. What Eq. 9 essentially does is rotate the cluster locations and covariances of $\mathcal{H}^m$, so they are aligned with the orientation of each cluster in $\mathcal{H}^s$ as they are iteratively convolved. To increase efficiency, these eigenvector matrices can be calculated during the clustering process and stored for later use.

Iterative Continuous Convolution

Here we show how global optimization can be performed over $\mathcal{H}^a$, to determine the input point with maximum activation value (i.e. where $\mathcal{H}^s$ is most similar to $\mathcal{H}^m$). This is achieved using an iterative process composed of three steps: 1) Sample the function in search for global extrema candidates; 2) Local convergence using gradient descent; and 3) Increase cluster resolution to refine estimates. Pseudo-code for this process can be found in Alg. 2, where we can see is various stages: (line 5) HM modeling of scene and mask pointclouds, at the current resolution; (line 6) convolution process between these two occupancy models; (line 7) determine current global maximum within cluster mean points; (line 8) local optimization using gradient information; (line 9) eliminate scene points sufficiently far from the current maximum; and (line 10) decay average cluster distance for the next iteration.

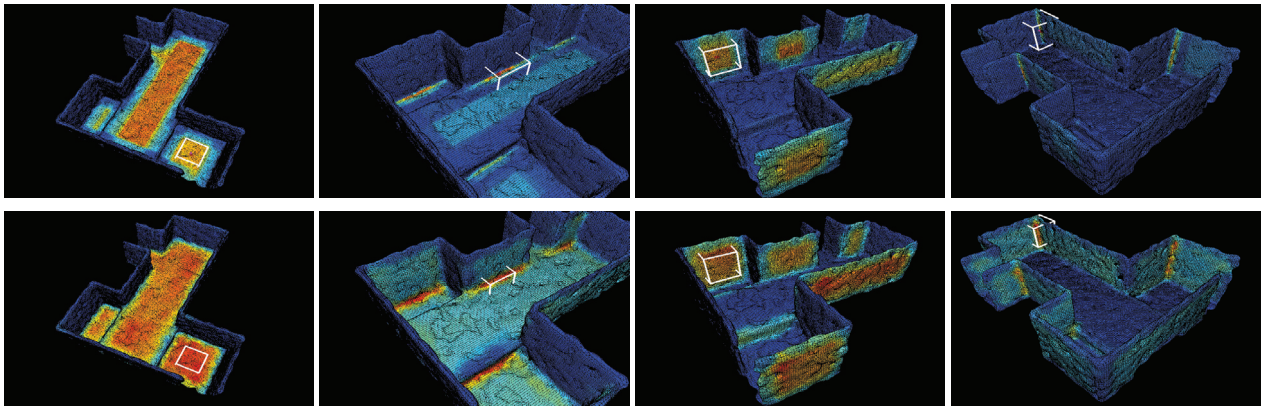


Figure 2: Examples of discrete (top row) and continuous (bottom row) convolution in a simulated 3D environment. The entire dataset is convolved with a subset of it, defined by a white cube of side $s = 0.5m$. A resolution of $r = 0.1m$ was used to produce the voxel grid for discrete convolution, and the same value was used as the average cluster distance for continuous convolution.

Algorithm 2 Iterative Continuous Convolution

Input: $\mathcal{D}^s$ and $\mathcal{D}^m$, initial average cluster distance r, number of iterations n and distance decay $\gamma < 1$
Output: Global maximum $\mathbf{x}_{opt}$
$d \leftarrow \text{argmax}_{\mathbf{x} \in \mathcal{D}^s} \|\mathbf{x}\|$ % Mask radius
for $i = 0$ **to** n **do**
  $\mathcal{H}^s, \mathcal{H}^m \leftarrow HM(\mathcal{D}^s, r), HM(\mathcal{D}^m, r)$
  $\mathcal{H}^a \leftarrow CC(\mathcal{H}^s, \mathcal{H}, r)$ % Perform convolution
  $\mathbf{x}_{opt} \leftarrow \text{argmax}_{\boldsymbol{\mu}} \mathcal{H}^a(\boldsymbol{\mu} \in \mathcal{M}^s)$ % New maximum
  $\mathbf{x}_{opt} \leftarrow grad\_desc(\ \mathbf{x}_{opt}\ ,\ \frac{\partial}{\partial \mathbf{x}} \mathcal{H}^a\ )$ % Grad. descent
  $\mathcal{D}^s \leftarrow \{\ \mathbf{x} \in \mathcal{D}^s\ \mid\ \|\mathbf{x} - \mathbf{x}_{opt}\| < d + 2 \cdot r\ \}$
  $r = \gamma r$ % Decay average cluster distance
end for

The derivative of $\mathcal{H}^a$ in relation to $\mathbf{x}$ is obtained by first calculating the derivative of its kernel, which in the case of a Gaussian distribution (Eq. 4) is given by:

$$\frac{\partial}{\partial \mathbf{x}} k(\mathbf{x}, \mathcal{M}_i) = \frac{\mathbf{d}^T \Sigma_i^{-\frac{1}{2}}}{\sqrt{(2\pi)^D |\Sigma_i|}} \exp\left(-\frac{1}{2} \mathbf{d}^T \Sigma_i^{-\frac{1}{2}} \mathbf{d}\right) = \mathbf{d}^T \Sigma^{-\frac{1}{2}} k(\mathbf{x}, \mathcal{M}_i), \quad (10)$$

where, again, $\mathbf{d} = \mathbf{x} - \boldsymbol{\mu}_i$. Subsequently, the derivative of a feature vector defined by these kernels, and used to produce $\frac{\partial}{\partial \mathbf{x}} \mathcal{H}^a$, can be written simply as $\frac{\partial}{\partial \mathbf{x}} \Phi(\mathbf{x}, \mathcal{M}) = \frac{\partial}{\partial \mathbf{x}} \left[k(\mathbf{x}, \mathcal{M}_1)\ ,\ k(\mathbf{x}, \mathcal{M}_2)\ ,\ \ldots\ ,\ k(\mathbf{x}, \mathcal{M}_M)\right]^T$, i.e. the derivative of each kernel independently (which is possible because they are factored as a sum, see Eq. 1).

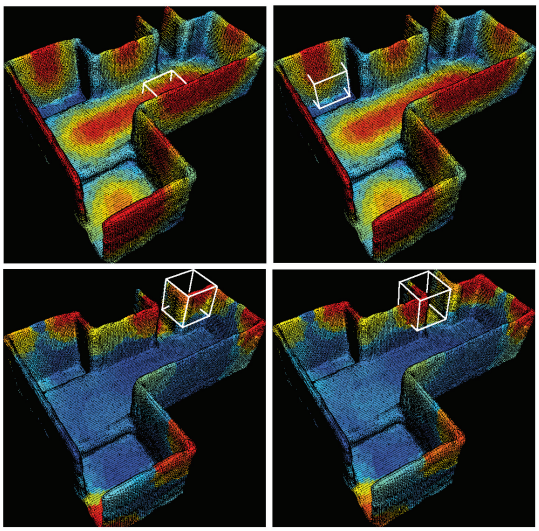
Note that a smaller subset of $\mathcal{D}^m$ is used at each iteration, since it is refining previous estimates, which promotes efficient calculations even at high resolutions. Additionally, multiple hypotheses can be maintained by refining several candidates in parallel, and the output can be used as the starting point for other local optimization techniques. Note that there is a practical limit to r, because as resolution increases each cluster receives fewer data points, which are used to calculate mean and variance parameters. So, rather than a fixed number of iterations, the refinement process stops when a significant percentage of clusters has fewer than a certain threshold of points (in experiments we used 10% of clusters with fewer than 5 points).

4 Experimental Results

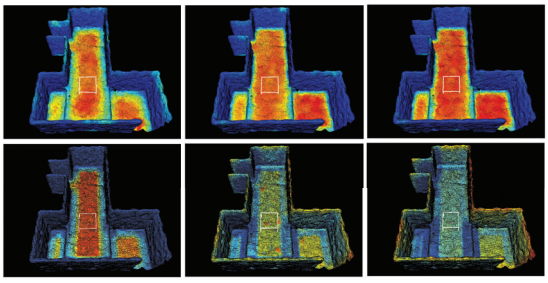
Experiments were conducted to validate the proposed approach to continuous convolution in 3D space, and how it performs in relation to standard sliding window and Fourier discrete convolutions. Initially a virtual dataset was used, composed of 44000 points covering an area of roughly 60 m^3. The entire dataset served as the scene, and a cubic subset was selected as the mask. For a fair comparison, the same resolution used to produce the discretized grid served as the average cluster distance.

Fig. 2 depicts convolution results when using discrete and continuous techniques. Note that, while discrete convolution tends to produce sharper boundaries between areas with high and low activation values, continuous convolution provides a smoother transition that better depicts "partial matches", i.e. scene areas that share some similarities with the mask. Fig. 3a shows continuous convolution results for the same dataset when using the rotational invariance technique described in Sec. 3. In the top row, similar activation values were obtained when convolving horizontal and vertical surfaces, while in the bottom row two opposite wall corners were convolved also with similar activation values.

To compare the computational speed of each technique, we performed convolutions on this dataset using different mask sizes s and resolutions r, with results presented in Fig. 4. As expected, for very small mask sizes discrete techniques are faster, however as the mask size increases, especially for higher resolutions, the proposed technique quickly surpasses even the Fourier convolution in terms of computational speed. Note that discrete techniques require the generation of grid representations of its input functions, a step absent in the continuous convolution process. As an example, for the above



(a) Continuous convolution with rotational invariance.



(b) Effects of changing resolution in continuous (top row) and discrete (bottom row) convolutions.

Figure 3: Examples of resolution and rotational invariance properties presented by the proposed convolution technique.

mentioned dataset a discretized grid generated with $r = 0.02m$ contains around 370 million points, while only 2000 clusters are necessary for the continuous representation.

A breakdown of computational times[3] during discrete and continuous convolution is depicted in Table 1, where we can see that grid generation takes up a substantial amount of time in relation to the actual convolution process, particularly for the Fourier technique. This computational time also increases cubically with resolution, while the generation of a Hilbert Map from the same dataset is not only much faster, but increases roughly linearly with resolution. Interestingly, we noticed during experiments that higher resolutions actually decrease the accuracy of discrete convolution techniques, due to the sparse and noisy nature of the input data (see Fig. 3b) that produces random gaps in the generated grid. The ability of the Hilbert Maps framework to deal with sparse data has been noted in (Guizilini and Ramos 2016), and here it contributes to the calculation of more robust activation values, smoothing out random variations in input data.

[3]All computations were performed on a *i7/2.60 x 8 GHz* notebook, with multi-threading wherever possible.

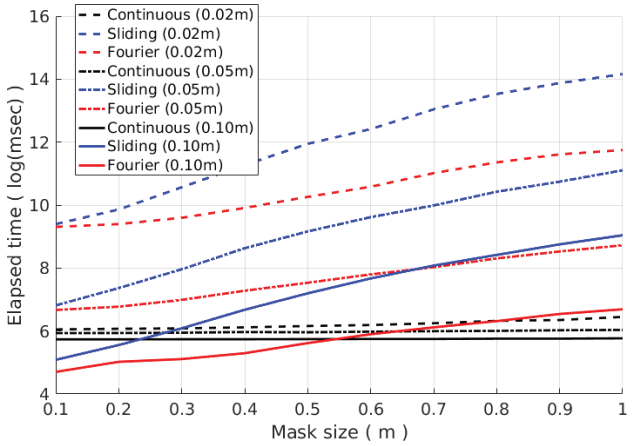


Figure 4: Speed comparison between different convolution techniques (average of 50 runs with random masks).

As an application of the proposed technique in a context of 3D template matching, in which a larger pointcloud is searched for areas that fit a smaller one, we selected the key problem in robotics of global localization (i.e. the kidnapped robot problem). Two large-scale real datasets were used, representing a structured corridor ($n = 474557$) and an outdoor area ($n = 670584$). From these datasets, various unambiguous ground-level points were selected and pointclouds were extracted from a $3m$ radius, serving as the template to be matched against the entire dataset. Two other techniques were considered for comparison: the Generalized Iterative Closest Point (GEN-ICP) (Segal, Haehnel, and Thrun 2009) algorithm, as implemented by the PCL library (Rusu and Cousins 2011); and a Particle Filter with Approximate Optimal Sampling (AOS-PF) (Blanco, Gonzalez, and Fernandez-Madrigal 2008). Nested annealing (Rajasekaran 2000) was used to improve GEN-ICP initialization, and the AOS-PF algorithm only considered ground-level points as potential particles. Note that standard discrete convolution techniques would not be applicable for such large-scale scenarios, due to memory requirements and kernel sizes. In all cases, the

	Res. (m)	Generate (ms/m^3)	Convolve ($\mu s/pt$)	Infer ($\mu s/pt$)
Slid.	0.10	1.9	-	3.5
	0.05	12.4	-	26.1
	0.02	167.9	-	416.8
Four.	0.10	1.9	-	0.4
	0.05	12.4	-	3.1
	0.02	167.9	-	5.1
Cont.	0.10	1.1	0.1	0.6
	0.05	1.7	0.4	0.8
	0.02	2.5	0.9	1.0

Table 1: Computational times using different convolution techniques. In discrete techniques, convolution and inference are performed simultaneously, while the proposed framework produces a function that can be queried at any input point.

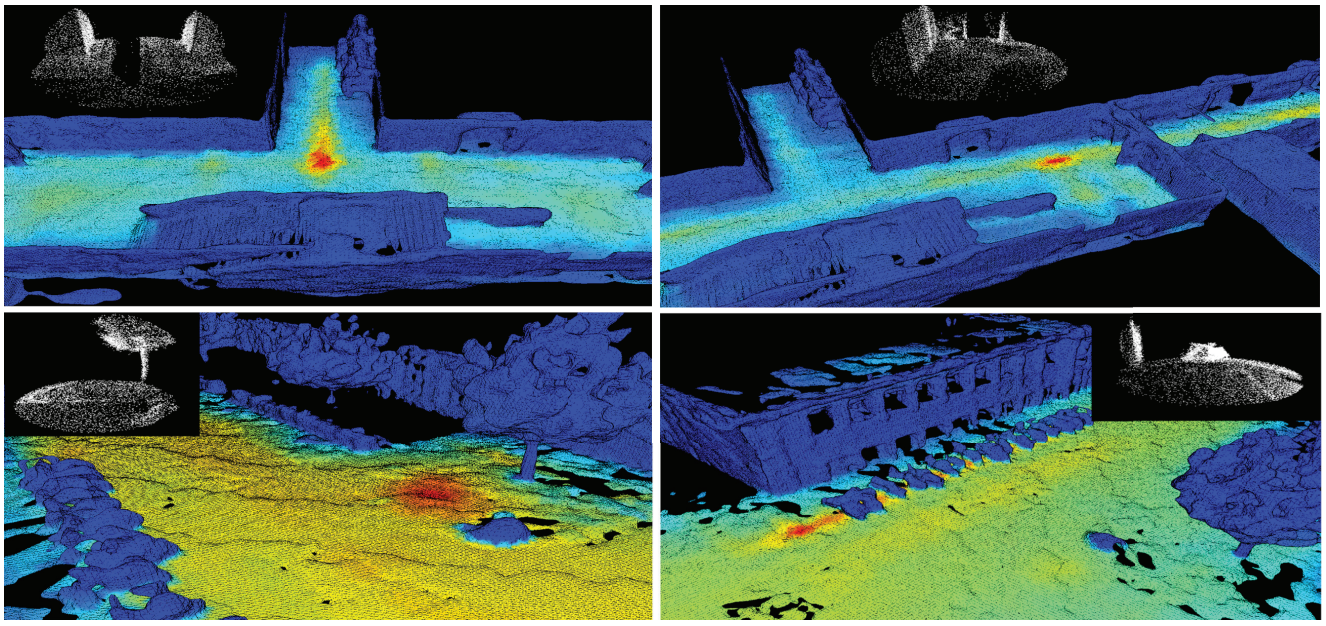


Figure 5: Template matching on real large-scale datasets, using the proposed continuous convolution technique. White dots indicate the template pointcloud, that is convolved with the dataset to produce $\mathcal{H}^a$, from which the global maximum is extracted.

template pointclouds were downsampled by a factor of 2 and arbitrarily rotated by up to 30^o, to test the ability to deal with sparse and misaligned data.

Fig. 5 shows some qualitative results obtained using the proposed continuous convolution technique (CC-HM). As expected, areas similar to the template produce higher activation values, that quickly decrease as we move away from their center. Note that the proposed technique is able to naturally deal with multiple hypotheses, in which activation values can be viewed as a probabilistic distribution for potential matches. Table 2 contains numeric comparisons between the various 3D template matching techniques considered here, in terms of accuracy and computational speed. Note that, while CC-HM produces comparable linear and angular errors, it converges much faster and more reliably than GEN-ICP and AOS-PF, that still sporadically diverge (linear error $> 1m$). We were able to further reduce these errors using the iterative convolution process described in Section 3 (ICC-HM), by incrementally decreasing r and thus increasing accuracy during optimal point calculation. Lastly, we tested the output of ICC-HM as initial estimates for GEN-ICP, producing the entry ICC-ICP, that converges in just a few iterations and achieves marginally better overall results with 100% convergence rate. This indicates that the proposed technique can be used as a first step for global optimization, since it searches the entire input space, and then further refined using conventional local techniques. Note that roughly half the time required by the proposed technique is due to dataset modeling (as shown in Table 1), that can be reused for convolutions with different templates without extra computational cost.

	Method	Error (cm)	Error (^{o})	Time (s)	Conv. ($\%$)
Corr.	GEN-ICP	11.9	3.7	55.9	60
	AOS-PF	10.4	5.2	82.0	90
	CC-HM	14.5	4.4	2.9	100
	ICC-HM	9.8	3.1	4.1	100
	ICC-ICP	9.5	2.9	7.3	100
Out.	GEN-ICP	15.5	5.1	75.6	55
	AOS-PF	12.4	5.8	102.3	75
	CC-HM	17.1	6.1	4.6	100
	ICC-HM	10.9	3.4	5.7	100
	ICC-ICP	10.3	3.2	9.3	100

Table 2: Comparison between different 3D template matching techniques. Each row represents the average of 20 runs, and the last column represents the convergence rate for these runs (only successful runs were considered).

5 Conclusion

This paper proposes a novel methodology for 3D template matching using continuous convolution, in which input data is modeled as occupancy functions using the Hilbert Maps framework, and calculations takes place directly in the higher-dimensional Reproducing Kernel Hilbert Space. We derive a closed-form solution for the convolution process that is highly scalable to larger kernel sizes, a scenario that remains challenging and is not addressed by most state-of-the-art techniques. Experimental results show a substantial increase (by orders of magnitude) in computational speed and decrease in memory requirements, relative to standard discrete convolution techniques. Experiments on large-scale global localization using real datasets were performed, with the proposed technique achieving results that outperform conventional techniques in terms of computational time, accuracy

and reliability. Future work will focus on extending the continuous convolutional framework to different scenarios, such as classification, producing much more compact filters that are able to approximate arbitrarily complex functions.

Acknowledgements

This research was supported by funding from the Faculty of Engineering Information Technologies, The University of Sydney, under the Faculty Research Cluster Program.

References

Arya, S.; Mount, D.; Netanyahu, N.; Silverman, R.; and Angela, Y. 1998. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM (JACM)* 45(6):891–923.

Bachem, O.; Lucic, M.; Hassani, S.; and Krause, A. 2016. Approximate k-means++ in sublinear time. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Bernd, J. 2005. *Digital Image Processing*. Springer-Verlag New York, Inc.

Blanco, J.-L.; Gonzalez, J.; and Fernandez-Madrigal, J.-A. 2008. An optimal filtering algorithm for non-parametric observation model in robot localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 461–466.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the International Conference on Computational Statistics (COMPSTAT)*, 177–186.

Bracewell, R. 2003. *Fourier Analysis and Imaging*. Springer-Verlag New York, Inc.

Briechle, K., and Hanebeck, U. 2001. Template matching using fast normalized cross correlation. In *Proceedings of the International Society for Optical Engineering*.

Bromiley, P. 2003. Products and convolutions of Gaussian probability density functions. Technical report, TINA Vision.

Brunelli, R. 2009. *Template Matching Techniques in Computer Vision: Theory and Practice*. John Wiley & Sons.

Castleman, K. 1996. *Digital Image Processing*. Prentice Hall Press.

Doherty, K.; Wang, J.; and Englot, B. 2016. Probabilistic map fusion for fast, incremental occupancy mapping with 3d Hilbert maps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Fialka, O., and Cadik, M. 2006. Fft and convolution performance on image filtering on gpu. In *Proceedings of the International Conference on Information Visualization (ICIV)*, 609–614.

Freedman, D. 2005. *Statistical Models: Theory and Practice*. Cambridge University Press.

Genton, M., and Kleiber, W. 2015. Cross-covariance functions for multivariate statistics. *Statistical Science* 30(2):147–163.

Gilks, W.; Richardson, S.; and Spiegelhalter, D. 1995. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1 edition.

Gneiting, T.; Kleiber, W.; and Schlather, M. 2010. Matérn cross-covariance functions for multivariate random fields. *Journal of the American Statistical Association* 105(491):1167–1177.

Guizilini, V., and Ramos, F. 2016. Large-scale 3d scene reconstruction with Hilbert maps. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*.

Hummel, R., and Lower, D. 1986. Computing large-kernel convolutions of images. Technical report, Courant Institute of Mathematical Sciences, New York University.

Jan, J. 2000. *Digital Signal Filtering, Analysis and Restoration (Telecommunication Series)*. INSPEC.

Komarek, P. 2004. Logistic regression for data mining and high-dimensional classification. Technical report, Carnegie Mellon University.

Li, Y.; Pirk, S.; Su, H.; Qi, C.; and Guibas, L. 2016. Fpnn: Field probing neural networks for 3d data. In *Advances in Neural Information Processing Systems (NIPS)*.

Opromolla, R.; Fasano, G.; Rufino, G.; and Grassi, M. 2015. A model-based 3d template matching technique for pose acquisition of an uncooperative space object. *Sensors* 15:6360–6382.

Rajasekaran, S. 2000. On simulated annealing and nested annealing. *Journal of Global Optimization (JGO)* 16(1):43–56.

Ramos, F., and Ott, L. 2015. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. In *Proceedings of Robotics: Science and Systems (RSS)*.

Riegler, G.; Ulusoy, A.; and Geiger, A. 2016. Octnet: Learning deep 3d representations at high resolutions. In *Computing Research Repository (CoRR)*.

Rusu, R., and Cousins, S. 2011. 3d is here: Point cloud library (pcl). In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Salomon, D. 2006. *Data Compression: The Complete Reference*. Springer-Verlag New York, Inc.

Schölkopf, B., and Smola, A. 2001. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press.

Segal, A.; Haehnel, D.; and Thrun, S. 2009. Generalized icp. In *Proceedings of Robotics: Science and Systems (RSS)*.

Steder, B.; Grisetti, G.; and Burgard, W. 2010. Robust place recognition for 3d range data based on point features. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Vaseghi, S. 2006. *Advanced Digital Signal Processing and Noise Reduction*. John Wiley & Sons.

Xie, J.; Nashashibi, F.; Parent, M.; and Favrot, O. 2010. A real-time robust global localization for autonomous mobile robots in large environments. In *International Conference on Automation, Robotics and Vision (ICARCV)*, 1937–1402.