Efficient Visuo-Haptic Object Shape Completion for Robot Manipulation

Lukas Rustler, Jiri Matas, and Matej Hoffmann

***Abstract*— For robot manipulation, a complete and accurate object shape is desirable. Here, we present a method that combines visual and haptic reconstruction in a closed-loop pipeline. From an initial viewpoint, the object shape is reconstructed using an implicit surface deep neural network. The location with highest uncertainty is selected for haptic exploration, the object is touched, the new information from touch and a new point cloud from the camera are added, object position is re-estimated and the cycle is repeated. We extend Rustler *et al.* (2022) by using a new theoretically grounded method to determine the points with highest uncertainty, and we increase the yield of every haptic exploration by adding not only the contact points to the point cloud but also incorporating the empty space established through the robot movement to the object. Additionally, the solution is compact in that the jaws of a closed two-finger gripper are directly used for exploration. The object position is re-estimated after every robot action and multiple objects can be present simultaneously on the table. We achieve a steady improvement with every touch using three different metrics and demonstrate the utility of the better shape reconstruction in grasping experiments on the real robot. On average, grasp success rate increases from 63.3% to 70.4% after a single exploratory touch and to 82.7% after five touches. The collected data and code are publicly available (`https://osf.io/j6rkd/`, `https://github.com/ctu-vras/vishac`).**

I. Introduction

We consider the following robotic setup. A static RGB-D camera connected to a robotic arm controller is observing one or more unknown 3D objects. To be able to grasp and manipulate the object, the robotic system needs a model of the object in terms of a complete shape, *e.g.*, an accurate mesh. There are intrinsic limitations to the performance of computer vision techniques for 3D reconstruction of objects from images or point clouds if only a limited number of viewpoints is available. Solutions relying on RGB or RGB-D images, LIDAR point clouds, or voxels, cannot easily overcome self-occlusion and may have specific difficulties with transparent or specular objects. The robot arm cannot reliably grasp and manipulate the object given only partial information. However, the manipulator can be controlled to touch or poke the object in order to extend the surface for which the model is accurate.

We address the following problem. Given an initial RGB-D map, obtain an accurate representation of the complete shape of the object with the help of exploratory contact actions. The objective is either to maximize the accuracy given an upper bound on the number of touches or minimize the number of touches to reach a predefined accuracy on the complete surface. This problem in turn requires solutions of sub-problems such as prediction of the least reliable part of the surface, estimation of the free space around the objects and the detection of touch-induced object movement.

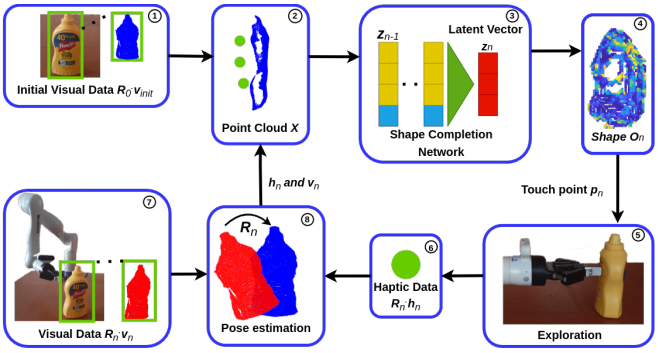

Fig. 1: Schematic operation of VISHAC. An initial RGB-D image of the scene is captured (1), a transformation $\mathbf{R}_0$ from the robot base to the object is obtained, and the object is segmented and converted into a point cloud $\mathcal{X}$ (2). Iterative reconstruction: In each step, $n = 0 : (N-1)$, the point cloud is inserted into a neural network (3) and a completed shape $\mathbf{O}_n$ is created (4). The most uncertain point $\mathbf{p}_n$ is selected for touch (5). After contact, the object may have been displaced, giving rise to a new transformation $\mathbf{R}_n$. Haptic data $\mathbf{h}_n$ (6) from contact and visual data by taking a new image from the RGB-D camera $\mathbf{v}$ (7) are collected. The transformation $\mathbf{R}_n$ is computed from pose estimation (8) and the new data, transformed into the original frame $\mathbf{R}_0$, are added to $\mathcal{X}$. See Sec. III-H for details.

The scenario models a setup, where new objects are presented to a system that minimizes the risk of an unstable grasp, and therefore "explores" the shape of the object by touching and poking before attempting a grasp and subsequent manipulation. For example, imagine a conveyor belt for sorting objects of different sizes into respective bins, e.g. in a scrapyard, where the robot must be able to pick up any object.

Contributions. We present a pipeline for visuo-haptic shape completion called VISHAC, importantly extending the work of Rustler *et al.* [1], which serves as a baseline here. The first group of improvements concerns the process of shape completion performed by Implicit Geometric Regularization for Learning Shapes (IGR), which we modified

This work was supported by the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics". L.R. was additionally supported by the Czech Technical University in Prague, grant no. SGS22/111/OHK3/2T/13.

Lukas Rustler, Jiri Matas, and Matej Hoffmann are with the Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, matej.hoffmann@fel.cvut.cz, matas@fel.cvut.cz, lukas.rustler@fel.cvut.cz.

as follows. We use a new, theoretically grounded, method to determine the points with highest uncertainty. In addition, we changed the sampling of points inside the network to respect the input point cloud more closely. Finally, the yield of every haptic exploration is increased by adding not only the contact points to the point cloud but also incorporating the empty space established through the robot movement to the object. The two last mentioned improvements together make the pipeline more robust. The second group of enhancements pertains to the practical aspects of this scenario. We do not use a dedicated "finger" for haptic exploration but directly the jaws of a closed two-finger gripper, which is a more compact solution. The object position is newly re-estimated after every robot action, so objects are allowed to move after being poked. Multiple objects can be present simultaneously on the table. The speed of the pipeline is improved through parallelization (2 times in simulation and 2.4 times in the real world). We contribute a more detailed evaluation using a set of different metrics. Finally, real-word grasping experiments demonstrate the effectiveness of our approach.

II. Related work

For the problem addressed, two main sources of information have been used: visual input (RGB or RGB-D sensors) and haptic exploration (tactile or force sensing). In recent work, the two were often combined.

A. Visual-only Shape Completion

Devices able to sense depth and thus create point clouds are widely available and can provide rich information about the scene. The early methods were based on geometric properties or templates. The geometric ones either assume that most objects humans use are symmetric, so completion can be done by mirroring partial information about its axis of symmetry [2], or detect primitive shapes [3]. Template-based methods benefit from prior knowledge in the form of a database of object shapes [4].

More recently, solutions based on machine learning gained popularity. An example is Gaussian Process Implicit Surface (GPIS) [5], which, however requires points on the whole surface and suffers from poor scaling over a dense point cloud. Thus, the input must be downsampled, resulting in loss of detail. Other methods were originally created to make surfaces from complete point clouds, but provide shape completion abilities by interpolating between shapes in a latent space [6]–[8]. In this work, we start from IGR [8] and propose improvements.

Deep Learning (DL) techniques such as Convolutional Neural Networks (CNNs) for shape completion typically represent objects as voxel grids. This allows to introduce probabilistic uncertainty in voxel grids, but the methods usually suffer from cubically growing computational requirements with the number of voxels, limiting the resolution of the output shape. Finally, the newest methods utilize graph attention networks [9] or transformers to complete a shape [10], [11].

B. Haptic-Only Shape Completion

With advances in haptic exploration, some purely haptic approaches have been proposed. They utilize some techniques mentioned above, such as implicit shape potentials [12], or Gaussian Processes (GPs) [13]–[15]. Gaussian-based methods have the advantage of having the ability to express uncertainty directly from their nature using the variance of each point. However, haptic-only completion needs a high number of touches, which is time-demanding.

C. Visuo-Haptic Shape Completion

A combination of visual and haptic data has the potential to combine the best of both worlds. GP methods were proposed [16]–[18]. However, points covering most of the surface are needed, which leads to the need for a lot of exploration. A way to overcome this issue may be to use symmetry as in [19].

CNN-based methods [20], [21] usually require fewer touches but suffer from lower resolution due to computational requirements. Smith *et al.* proposed approaches [22], [23] based on Graph Neural Network (GNN). Reconstructions by these methods have a higher resolution but are nonsmooth and, for now, evaluated only in simulation.

An important part of haptic exploration is the decision where to touch. The object can be touched randomly as done by Smith *et al.* [22], or always select a position opposite the camera (from "behind") as Watkins-Vall *et al.* [20]. However, these are not as effective as an uncertainty-driven approach. Uncertainty can come from the Gaussian distribution [16]–[19], [21]; from the Monte Carlo dropout [24]; or from the Signed Distance Function (SDF) [1], [25]. Alternatively, it can be learned where to touch as in Smith *et al.* [23].

Our work belongs to uncertainty-driven approaches and is based on implicit surface deep neural network IGR [8], exploiting the definition of SDF to estimate uncertainty in order to efficiently explore the most promising parts of the objects. We extend and directly compare ourselves with [1]. Indirectly, this encompasses also a comparison with other methods, namely [26]–[29], which were outperformed in [1].

III. Method

We propose an iterative method depicted in Fig. 1. The objective is to iteratively improve shape reconstruction of objects on the table combining images and selecting locations for tactile exploration. The algorithm is described in detail in Section III-H. The following sections detail individual modules required by the pipeline.

A. Implicit Surfaces

An implicit surface is a set of points whose signed distance to a surface is equal to zero. The function to compute this distance is called Signed Distance Function (SDF) and is defined as

$$f(\mathbf{x}) = s, \tag{1}$$

where, in our case, $\mathbf{x}$ is a point defined in 3D space and s is the signed distance. Traditionally, f would be described

analytically, but it can also be learned with a neural network. Then, the implicit surface generated with a neural network can be described as

$$\mathcal{M} = \{\mathbf{x} \in \mathbb{R}^3 \mid f(\mathbf{x}; \boldsymbol{\theta}) = 0\}, \tag{2}$$

where $f(\mathbf{x}; \boldsymbol{\theta}) : \mathbb{R}^3 \to \mathbb{R}$ is a Multi-Layer Perceptron (MLP) learned to approximate SDF, with $\boldsymbol{\theta}$ being the parameters of the network.

B. Implicit Geometric Regularization for Learning Shapes

As in [1], we selected IGR [8] to represent the function f. The method assumes at input a point cloud $\mathcal{X} = \{\mathbf{x}_{1:C}\}$, where C is the number of points in the point cloud, and optionally a set of normals for each point $N = \{\mathbf{n}_{1:C}\}$. To train on multiple objects, the network utilizes an auto-decoder architecture from Park *et al.* [6] with different latent code $\mathbf{z}_i$ for every shape $i \in I$ in the input set. In the prediction phase $|I| = 1$. Therefore we will introduce loss only for one shape $i \in I$, defined as:

$$\ell(\boldsymbol{\theta}, \mathbf{z}_i) = \ell_{\mathcal{X}}(\boldsymbol{\theta}, \mathbf{z}_i) + \lambda \mathbb{E}_{\mathbf{x}} \left[\|\nabla_{\mathbf{x}} f(\mathbf{x}; \boldsymbol{\theta}, \mathbf{z}_i)\| - 1 \right]^2 + \alpha \|\mathbf{z}_i\| \tag{3}$$

where

$$\ell_{\mathcal{X}}(\boldsymbol{\theta}, \mathbf{z}_i) = \frac{1}{C} \sum_{c=1}^{C} (|f(\mathbf{x}_c; \boldsymbol{\theta}, \mathbf{z}_i)| + \|\nabla_{\mathbf{x}} f(\mathbf{x}_c; \boldsymbol{\theta}, \mathbf{z}_i) - \mathbf{n}_c\|) \tag{4}$$

The first term in Eq. 3 encourages f to vanish on $\mathcal{X}$ and $\nabla_{\mathbf{x}} f$ to be close to the supplied normals. The second term is called the Eikonal term and regularizes the network by pushing $\nabla_{\mathbf{x}} f$ to be of unit Euclidean norm. The term is also used for uncertainty estimation—described later in Section III-E.

The result is iteratively optimized at both train and inference time (over multiple shapes in batches from the whole train set I while training and over one shape for prediction). In this work, we use a trained network from [1] and our modifications pertain to inference only. The parameters $\boldsymbol{\theta}$ are fixed during inference and only $\mathbf{z}_i$ is changed.

C. IGR Modifications – Sampling and Free Space

By default, the IGR network uses a random sampling of points in each of its iterations. It is effective when a complete point cloud is inserted. However, when a partial point cloud is used, the network tends to inflate the objects, *i.e.*, the output shape is over the boundaries of the input. We propose to use Farthest Point Sampling (FPS) as in [30]. The points sampled by this algorithm are spatially far from each other and help the network better understand the whole object in each iteration, making the final shape tighter with the input.

Another improvement of the method is the use of information about the space explored. As the robot moves through space, we know that the traversed space is free and no shape can be there. We keep in memory only the points that are less than $20\,\mathrm{cm}$ from the center of a given object. During the inference phase of the network, a signed distance is calculated for every point in the free space explored. We want all free space points to be outside of the surface, *i.e.*, to have a positive signed distance. Therefore, we add to the loss the sum of the absolute values of all distances that are lower than $1\,\mathrm{mm}$.

D. Object Representation from Visual and Haptic Data

There are several possible representations of an object O. We choose to represent an object as a point cloud (which is, in fact, an implicit surface from Eq. 2), concatenated with uncertainty for each point. Mathematically expressed as

$$O = \{\mathbf{x} \in \mathbb{R}^3, u \in \mathbb{R} \mid f(\mathbf{x}; \boldsymbol{\theta}, \mathbf{z}) = 0, u \geq 0\}, \tag{5}$$

where u stands for the uncertainty of the given point.

Our method is iterative, therefore, we compute a new shape in each iteration n. For $n = 0$ the shape O_0 is computed only from RGB-D information $\mathbf{v}_{init}$. In all other iterations we get the shape O_n with visual information $\mathbf{v}_{init}$, $\mathbf{v}_{0:(n-1)}$, and haptic information $\mathbf{h}_{0:(n-1)}$. Eq. 5 is therefore changed to

$$O_n = \{\mathbf{x} \in \mathbb{R}^3, u \in \mathbb{R} \mid f(\mathbf{x}; \boldsymbol{\theta}, \mathbf{z}_n) = 0, u \geq 0\}, \tag{6}$$

where $\mathbf{z}_n$ is the current latent vector optimized on point cloud $\mathcal{X}_n = \{\mathbf{h}_0, \ldots, \mathbf{h}_{(n-1)}, \mathbf{v}_0, \ldots, \mathbf{v}_{(n-1)}, \mathbf{v}_{init}\}$.

To obtain haptic information $\mathbf{h}_n$, we must first select the position for haptic exploration $\mathbf{p}_n$ on the object that minimizes the global uncertainty of the object, *i.e.*, select the point with the highest current uncertainty as

$$\begin{aligned} m &= \underset{u}{\operatorname{argmax}}\, O_n, \\ \mathbf{p}_n &= \mathbf{x}_m \in O_n. \end{aligned} \tag{7}$$

The desired $\mathbf{h}_n$ is then obtained from the position of the actual contact between the robot and the object.

Note that we showed the equations for only one object at a time. However, our method is capable of handling multiple objects in a complex scene. So, we will later in an algorithm refer to objects as $O_{k,n}$, where $k = 1 : K$ is the object's id in the scene, and n is the current iteration.

E. Object Shape Uncertainty

A crucial part of this work is the estimation of uncertainty. We selected part of the loss from Eq. 3, particularly the Eikonal loss

$$\ell_{Eikonal} = (\|\nabla_{\mathbf{x}} f(\mathbf{x}; \boldsymbol{\theta}, \mathbf{z})\| - 1)^2. \tag{8}$$

It was proven by Takashi [31] that a given function $f(\mathbf{x})$ that meets the condition of the Eikonal Equation $\|\nabla_{\mathbf{x}} f(\mathbf{x})\| = 1$ on a Riemann manifold M is a SDF to a hypersurface M. Furthermore, Crandal and Lions [32] presented Viscosity Solutions that prove the same applies even if $\|\nabla_{\mathbf{x}} f(\mathbf{x})\|$ does not exist on every $\mathbf{x}$. Given these results, we can compute the Eikonal loss from Eq. 8 for all points on our current shape O—where the higher the loss, the higher the uncertainty.

F. Segmentation of Multiple Objects

First, bounding boxes of all objects in the input RGB image are found—in the real world using Yolov7 [33] fine-tuned on our objects, and using color-based segmentation in the simulation. We then run the Flood Fill algorithm [34] on the depth image (aligned to RGB). The algorithm starts at a given pixel (we use the center of the bounding box) and expands over neighbors that fulfill the given criterion

Algorithm 1 Create Shape

while Pipeline is running **do**
  k = *SelectFromQueue() or WaitForRequest()*;
  $\mathbf{z}$ = *LoadLatent(k)*;
  $\mathcal{X}$ = *LoadData(k)*;
  $\mathbf{z}_{optimized}$ = *Optimize*$(\mathbf{z}, \mathcal{X})$; ▷ Loss from Eq. 3
  O = *PredictShapeAndUncertainty*$(\mathbf{z}_{optimized})$;
end while

(difference in depth in our case) until no neighbor is left. We also restrict the region of interest by the bounding box from the RGB image. Segmented depths, together with camera information, are then used to create point clouds of objects. A more detailed description can be found on GitHub `https://github.com/ctu-vras/vishac`.

G. Pose Estimation

In haptic exploration methods, a common but unrealistic assumption is that objects are fixed to the surface (also in [1]). Objects naturally move when they are touched and their pose needs to be re-estimated. Many existing pose estimation methods require prior knowledge of the objects at the instance level [35], [36] or category level [37], [38]. We seek methods that work with unknown arbitrary objects. Having segmented point clouds of each object at hand, we chose a simple and computationally cheap (no GPU) solution using Iterative Closest Point (ICP) [39]. Alternative solutions for unknown objects are [40], [41].

H. VISHAC Algorithm

We present the algorithm of our method in Alg. 2 and the same is depicted in Fig. 1. The algorithm is high-level pseudocode, with a module for shape completion Alg. 1 described in more detail.

We will first describe the module for the shape creation itself. In [1] the IGR network was used as a standalone library. To perform more efficiently and to be able to handle more objects at once, we modified it to be more compatible with the whole ecosystem (under Robot Operating System (ROS)). The module contains the input point clouds, latent vectors, and other parameters for each object in the scene, allowing simple switching between objects without excessive overhead. The next object to be completed is selected through messages sent from the main script. If a new request is received and reconstruction is running, the new objects are placed in a queue. The module runs in the background, which allowed a considerable speed-up of the whole process, as now reconstructions are processed while the robot is moving. The basic operation is shown in Alg. 1. First, a new shape is selected from a queue (if it is not empty, otherwise the module waits for a new request). Then the latent vector $\mathbf{z}$ and the input point clouds $\mathcal{X}$ for the given shape are loaded. If the shape is new, the first latent code is created randomly with a normal distribution. Otherwise, the last known vector

Algorithm 2 Multi-Object Shape Completion

Input: maximal number of haptic explorations N, maximal run time T
Output: Final shape $O_{1:K,n}$ ▷ For K shapes
1: t_{init} = *CurrentTime*();
2: Start *CompleteShape*() service in background;
3: $\mathbf{v}_{init}$ = *CaptureVisualInformation*();
4: $\mathcal{X}_{1:K}$ = *Segment*$(\mathbf{v}_{init})$;
5: $k = 1 : K$; ▷ Start with all objects
6: **for** $n = 0, \ \ldots, \ (N-1)$ **do**
7: **if** *CurrentTime*() $- t_{init} \geq T$ **then**
8: **break;**
9: **end if**
10: $\mathbf{R}_{1:K,n}$ = *ComputePose*$(1, \ldots, K)$;
11: $O_{k,n}$ = *CompleteShapeRequest*(k);
12: $\mathbf{p}_n; k$ = *SelectTouchPoint*$(O_{1:K,n}, \mathbf{R}_{1:K,n})$;
13: *MoveRobot*$(\mathbf{p}_n)$;
14: $\mathbf{R}_{k,n} \cdot \mathbf{h}_n$ = *GetContactInformation*();
15: $\mathbf{R}_{k,n} \cdot \mathbf{v}_n$ = *CaptureVisualInformation*();
16: $\mathbf{R}_{k,n}$ = *ComputePose*(k);
17: $\mathbf{v}_n; \mathbf{h}_n$ = *Transform*$(\mathbf{R}_{k,n} \cdot \mathbf{v}_n, \mathbf{R}_{k,n} \cdot \mathbf{h}_n, \mathbf{R}_{k,n})$;
18: $\mathcal{X}_k$ += *Segment*$(\mathbf{v}_n)$;
19: $\mathcal{X}_k$ += $\mathbf{h}_n$;
20: **end for**
21: **Return:** $O_{1:K,n}$

for the given object is used. The current $\mathbf{z}$ is optimized with the loss from Eq. 3. Finally, the shape O is created, together with the uncertainty computed with Eq. 8.

The main Alg. 2 starts with capturing the initial visual information (box (1) in Fig. 1, line 3 in Alg. 2). An initial transformation $\mathbf{R}_0$ of the object in the base frame of the robot is obtained. The information is then segmented and a point cloud is created for each object in the scene (box (2), line 4). The segmentation itself is described in III-F.

Every iteration starts with computation of the current pose for all objects in the scene (Section III-G). The pose for all objects is computed here—the explored object may change pose after the touch is released and surrounding objects may have been moved unintentionally, so it is more robust to compute pose for all objects. This pose is used mainly to correctly select the point to explore.

Having the segmented point clouds and poses, a request for shape is sent (box (3), line 11). In the first iteration, we request shapes for all objects to create collision shapes for the motion planning algorithm. In other iterations, we request shape only for the last touched object. The impact position $\mathbf{p}_n$ is selected (box (4), line 12) based on Eq. 7. When more than one object is available, the impact position is selected as the point with maximal uncertainty among all the objects. To prevent exploration of only one object, we allow one object to be touched three times in a row, and then it is removed from the touch-selecting algorithm for the given iteration.

After $\mathbf{p}_n$ is selected, the robot is moved to the position

and contact information is extracted (box (5-6), lines 13-14). The movement consists of two subsequent movements. Firstly, the robot is moved to a position $10\,\mathrm{cm}$ from object along the normal of $\mathbf{p}_n$. Next, the robot is moved along the normal with linear movement until contact occurs. In our case, the contact is detected with the change in joint torques. Haptic information $\mathbf{h}_n$ is created as a circle perpendicular to the impact normal, with the center in the position of the end effector.

After collision, new visual information is saved, segmented and added to the point cloud for the touched object, together with the haptic information (box (7), lines 15-19). To make sure that we segment the correct object, the RGB-D information is cropped with the bounding box found for the given object in the last iteration (the box is slightly enlarged to allow movement of the object). Finally, the pose $\mathbf{R}_n$ of the object right after touch (before the contact is released) is computed (box (8), line 16). This pose is used to transform the current data into the frame of $\mathbf{v}_{init}$—the first frame must be used so that the latent vectors for the given objects can be reused. Note that now only the pose of the explored object is computed, unlike at the beginning of each iteration (line 10). Also, in Fig. 1, only one pose estimation is shown for simplicity.

The whole pipeline runs until the selected number of touches (over all objects) is done or until the time limit is reached.

IV. Experiments and Results

The primary experiments we conducted demonstrate how the completeness and accuracy of reconstructions change with each additional touch. In addition, we evaluated the quality of the reconstructions in a real-world grasping experiment. Examples are shown in the accompanying video.

We evaluated the reconstruction obtained by VISHAC on the eight objects shown in Fig. 2 and on one more object for grasping—a transparent spray bottle for which ground truth is not available, and thus it is not possible to compute the metrics. Nevertheless, it is a typical object where haptic feedback dramatically improves the initial reconstruction from the RGB-D visual sensor. None of the objects was used in training the shape completion network.

In the real-world setup, we filled the objects with water to make them weigh about $0.5\,\mathrm{kg}$ which simplifies collision detection by the robot. Note that in the experiments evaluating the Act-VH method [1], the objects were glued to the table and a dedicated finger was used. The VISHAC method introduced a component that tracks object movement caused by touches (box (8) in Fig. 1). The sensitivity of the torque sensors is not sufficient for manipulation of very light objects. The current setup with objects partially filled with water is more challenging than gluing the objects. To avoid disadvantaging the reference method Act-VH, we report its results for objects glued to the table.

The arrangement of the experimental bench is shown in Fig. 2. The robot is a Kinova Gen3 with a Robotiq 2F-85 gripper and a Intel RealSense D435 camera.

Fig. 2: The real-world robot setup with Kinova Gen3 robot, Robotiq 2F-85 gripper, external RGB-D camera and all objects used. Closed gripper was used for haptic exploration, open for grasping.

A. Evaluation Metrics

We used three metrics to evaluate accuracy: (i) Jaccard similarity (JS), *i.e.*, the intersection over union of voxelized shapes; (ii) Chamfer distance (CD), *i.e.*, the average distance of each point in one set to the closest point in the second set and vice versa; (iii) and the deviation of the reconstructed mesh surface area from the ground truth.

We use three metrics, since the information they provide is complementary. JS does not take into account the shape of the intersection and of the union, *e.g.*, it attains the same value for a sharp hallucinated peak or a thin layer of added volume. CD is highly informative in most cases, but it is oversensitive to small scale changes, even if the reconstructed shape is close to ground truth. The deviation of the area of the mesh evaluates the accuracy of the estimated scale and allows to check for biases.

Unless otherwise stated, experiments for each scene were repeated three times. We show the performance with real time on the x-axis, with individual touches numbered.

B. Simulation Experiments

The simulation environment consists of a robot modeled in the MuJoCo [42] simulator controlled through ROS. Objects are able to move on the table. Collision with the object is computed from the manipulator joint torques—as in the real setup. Throughout this section, we compare the performance of VISHAC with two variants of Act-VH [1]: 'Act-VH' and 'Act-VH – new data'. Act-VH constitutes the original experimental results from [1], in the setup with objects fixed to the table and poking with a dedicated probe, and IGR without any major changes. Only the results for the objects used here were selected. This comparison also demonstrates improvements in the runtime of the methods. 'Act-VH – new data' is a result of running the method from [1] on the data from the new setup—contacts with closed gripper and objects moving as a result of haptic exploration. This serves to isolate the benefits of the modifications of IGR inference in VISHAC. Act-VH was run until 5 touches were completed; VISHAC and 'Act-VH – new data' until 15 touches were done.

1) Reconstruction – one object in the scene: A comparison of performance is shown in Fig. 3. Both 'Act-VH' and 'Act-VH – new data' are more "greedy" with higher reconstruction accuracy gains per touch. However, this comes at the expense of accuracy as more data come in. VISHACis more "conservative", but maintains a steady performance gain. The relative increase in performance in the time when 'Act-VH' completed 5 touches (approximately touch 10 in out method) is 7.7% in JS and 21.3% in CD. After the last touch, VISHAC is better than 'Act-VH – new data' with a relative difference of 15.5% in JS and 31.7% in CD.

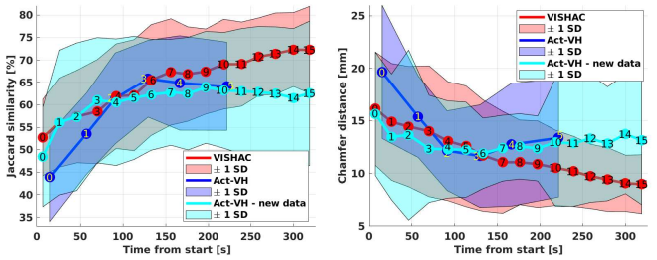

Fig. 3: Simulation – reconstruction – 1 object in scene. Average reconstruction accuracy (8 objects, 3 repetitions each). Numbers in each datapoint – number of touches. Shaded areas – standard deviation. Jaccard similarity (JS) higher values better. Chamfer distance (CD) lower values better.

The same trend can be seen from the shaded areas, showing the confidence interval of ± 1 standard deviation. The width of the areas shows the variability of the results for each object and each repetition of the experiment. For VISHAC, the variability shrinks with time, showing that the method is more precise and robust. This is not the case for both versions of Act-VH.

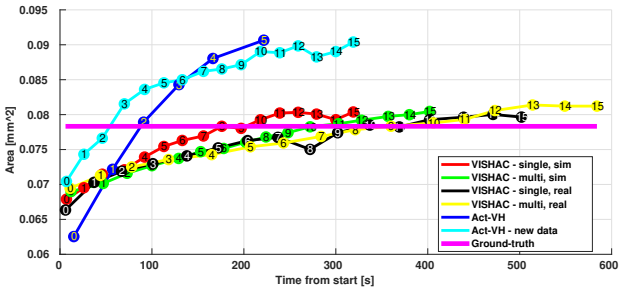

Fig. 4: Simulation and real experiments. Mean area of meshes. Numbers in each datapoint – number of touches. *Single* – scenes with only single objects; *multi* – scenes with more objects. *Act-VH* is a baseline from [1] and *Act-VH - new data* is the same method evaluated on data collected in this work.

In Fig. 4, the deviations in the mesh area are shown. We can see that even though, for example, JS performance for Act-VH in simulation was similar to VISHAC (see Fig. 3) there is a significant difference in area (blue for Act-VH, red for VISHAC). The baseline method (evaluated on both new and original data) tends to inflate the shapes, resulting in good results for JS or CD, but a high deviation in area. On the other hand, VISHAC converges to the ground truth value.

2) Reconstruction – multiple objects in the scene: In Fig. 5 results for reconstructions of multiple objects in a more complex scenes are shown. We randomly selected five configurations of objects with two or three of them present in each scene. The accuracy is almost the same as that for one-object-at-a-time experiments. However, the runtimes for 15 touches are about 80 seconds higher. This is mostly caused by more complicated touch point computation and motion planning. The deviations in the mesh area, shown in Fig. 4, further prove that the pipeline behaves similarly with one or more objects. Overall, the results show that the pipeline is able to handle multiple objects at once.

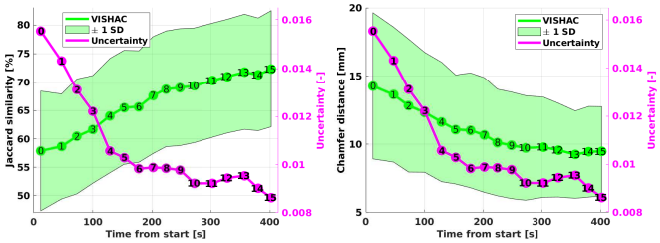

Fig. 5: Simulation – reconstruction – multiple objects in scene. Average reconstruction accuracy (5 scenes, 3 repetitions each). Numbers in each datapoint – number of touches. Shaded areas – standard deviation. Jaccard similarity (JS) higher values better. Chamfer distance (CD) lower values better.

Furthermore, we show how the uncertainty (purple line) changes over time. The uncertainty is computed as the mean of the uncertainties for each point of each object. One can see that the uncertainty decreases with increasing accuracy. Thus, it could be used to evaluate the quality of reconstruction during runtime and stop the pipeline when a predefined criterion is met.

C. Real-world Experiments

We tested the pipeline on the same set of objects as in simulation, in single- or multi-object configuration. The superior performance of VISHAC over Act-VH has already been demonstrated in simulation. Here we show the added value of VISHAC in grasping experiments.

1) Reconstruction: In Fig. 6, the results for the precision of the reconstruction are shown. The single-object experiments are shown in black. We can see that the trends of both JS and CD are the same as for the simulation, even though we can notice noise in some touches. In yellow, the results for multi-object experiments are shown. Again, the results for both types of experiments are similar, showing that the method is transferable to the real world. The overall accuracy in the real world is lower than in simulation. The

main reason is noise in the RGB-D sensor and inaccurate collision detection (noise in the joint encoders).

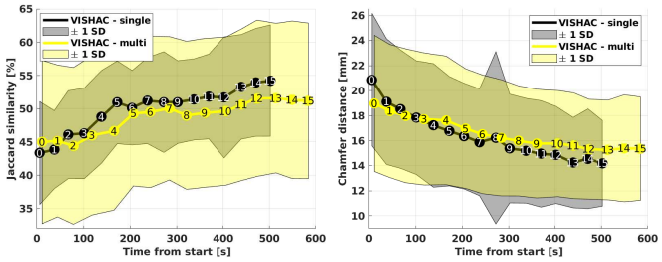

Fig. 6: Real world – Average reconstruction accuracy. 'VISHAC – single' – 8 objects, 3 repetitions each. 'VISHAC – multi' – 5 scenes, 3 repetitions each. Numbers in each datapoint – number of touches. Shaded areas – standard deviation. Jaccard similarity (JS) higher values better. Chamfer distance (CD) lower values better.

The mean mesh area is shown in Fig. 4. The area for single (black) and multiple (yellow) objects scenes approximately converges to the ground truth value.

2) Grasp Success Rate: The last experiment evaluates the grasp success rate, *i.e.*, the percentage of successful grasps. To sample grasp proposals, GraspIt! [43] was used. To check the quality of each grasp, the objects were picked and moved 10 cm in the upwards direction. If the object did not fall from the gripper, the grasp was marked as successful. We decided to inspect grasp success using reconstruction after 0 and 1 touches to show how only a single touch improves the result. In addition, reconstructions after touches 5, 10, and 15 are used. We attempted to grasp 3 times for every repetition of the pipeline on each object, resulting in 9 grasp per object per touch—that makes 81 grasps per touch and 405 grasps in total. The results are shown in Fig. 7.

There is already a difference between 0 and 1 touches. The success rate increased from 63.3% to 70.4%. Maximum success was achieved using reconstructions after 10 touches. However, the difference between 5 and 15 touches is only 2.5% (82.7% vs. 85.2%).

In general, we can say that 5 touches are enough for a sufficient grasp success rate. To compare, the maximum success rate achieved in the baseline [1] was 77.8%. It is also worth mentioning that the result was achieved after time that could be comparable to touch number 12 in our results.

V. Conclusion, Discussion, and Future Work

We proposed a new method for shape completion using a combination of visual and haptic feedback. VISHAC outperformed the baseline Act-VH [1] in terms of speed, reconstruction quality and robustness. We experimentally validated VISHAC in both simulated and real-world environments, using 8 objects and an additional one for grasping. VISHAC was evaluated in scenes with one, two, or three objects. We always touched the objects 15 times and repeated each experiment three times, resulting in almost one hundred experiments in total. In addition, a new uncertainty

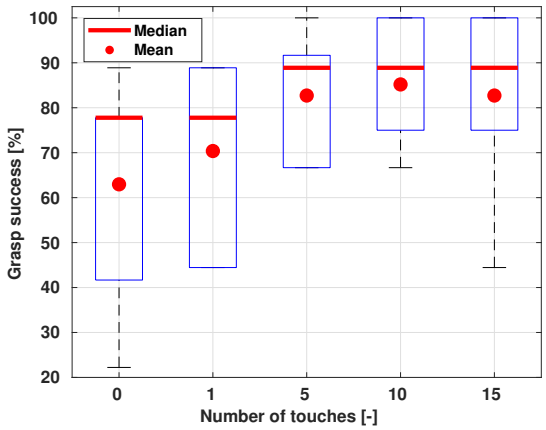

Fig. 7: Real-world grasp success rate after different number of touches. Means over 3 grasp attempts on each of 3 repetitions on each object (9 grasps per object; 81 per box) are used to create the boxplots. Box edges indicates 25th and 75th percentiles, with whiskers showing extreme points. Medians (red line) and means (red circles) are computed over all 9 objects.

computation strategy was evaluated, showing that it can be used for on-the-fly quality measurements. The reconstructions were furthermore validated with more than 400 grasps, demonstrating the usability of shape completion in a core robotic task.

There are several directions for future work. The results in the real setup are negatively affected by the noise induced by the contact events—the collision is detected with a certain delay, the object moves, and the new pose is not re-estimated perfectly. This could be mitigated in two ways. First, the most effective will be faster contact detection. In the current setup, collisions are detected from the joint torque sensors in the manipulator and its dynamic model and their remapping onto the end effector. In our setup, this leads to delayed and noisy estimation of the collision and significant movement of the object. A remedy would be a force/torque sensor at the robot wrist or tactile sensors at the end effector. Second, the object pose re-estimation after every haptic exploration could be further improved by using alternative pose estimators or adding tracking.

Furthermore, on a robot hand with sensorized fingertips, data for reconstruction could be collected more effectively by sliding the fingers over the object surface (tactile servoing). Finally, poking and touching reveal surface stiffness and other physical properties that play a role in grasping and could be exploited.

References

[1] L. Rustler, J. Lundell, J. K. Behrens, V. Kyrki, and M. Hoffmann, "Active Visuo-Haptic Object Shape Completion," *IEEE Robotics and Automation Letters*, vol. 7, pp. 5254–5261, 4 2022.

[2] J. Bohg, M. Johnson-Roberson, B. León, J. Felip, X. Gratal, N. Bergström, D. Kragic, and A. Morales, "Mind the gap-robotic grasping under incomplete observation," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 686–693.

[3] R. Schnabel, P. Degener, and R. Klein, "Completion and Reconstruction with Primitive Shapes," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 503–512.

[4] M. Pauly, N. J. Mitra, J. Giesen, M. H. Gross, and L. J. Guibas, "Example-based 3D Scan Completion," in *Symposium on Geometry Processing*, no. CONF, 2005, pp. 23–32.

[5] M. Li, K. Hang, D. Kragic, and A. Billard, "Dexterous grasping under shape uncertainty," *Robotics and Autonomous Systems*, vol. 75, pp. 352–364, 2016.

[6] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 165–174.

[7] M. Atzmon and Y. Lipman, "SAL: Sign Agnostic Learning of Shapes From Raw Data," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 2562–2571.

[8] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit Geometric Regularization for Learning Shapes," in *International Conference on Machine Learning*. PMLR, Nov. 2020, pp. 3789–3799.

[9] H. Huang, Z. Yang, and R. Platt, "Gascn: Graph Attention Shape Completion Network," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 1269–1278.

[10] A. Rosasco, S. Berti, F. Bottarel, M. Colledanchise, and L. Natale, "Towards Confidence-guided Shape Completion for Robotic Applications," 9 2022.

[11] D. Watkins-Valls, P. Allen, K. Choromanski, J. Varley, and N. Waytowich, "Multiple View Performers for Shape Completion," 9 2022.

[12] S. Ottenhaus, M. Miller, D. Schiebener, N. Vahrenkamp, and T. Asfour, "Local Implicit Surface Estimation for Haptic Exploration," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov. 2016, pp. 850–856.

[13] Z. Yi, R. Calandra, F. Veiga, H. van Hoof, T. Hermans, Y. Zhang, and J. Peters, "Active Tactile Object Exploration with Gaussian Processes," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 4925–4930.

[14] D. Driess, P. Englert, and M. Toussaint, "Active Learning with Query Paths for Tactile Object Shape Exploration," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2017, pp. 65–72.

[15] S. Dragiev, M. Toussaint, and M. Gienger, "Uncertainty aware grasping and tactile exploration," in *2013 IEEE International conference on robotics and automation*. IEEE, 2013, pp. 113–119.

[16] G. Z. Gandler, C. H. Ek, M. Björkman, R. Stolkin, and Y. Bekiroglu, "Object Shape Estimation and Modeling, Based on Sparse Gaussian Process Implicit Surfaces, Combining Visual Data and Tactile Exploration," *Robotics and Autonomous Systems*, vol. 126, p. 103433, Apr. 2020.

[17] S. Ottenhaus, D. Renninghoff, R. Grimm, F. Ferreira, and T. Asfour, "Visuo-Haptic Grasping of Unknown Objects Based on Gaussian Process Implicit Surfaces and Deep Learning," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, Oct. 2019, pp. 402–409.

[18] M. Björkman, Y. Bekiroglu, V. Högman, and D. Kragic, "Enhancing Visual Perception of Shape through Tactile Glances," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2013, pp. 3180–3186.

[19] A. A. Bonzini, L. Seminara, and L. Jamone, "Improving Haptic Exploration of Object Shape by Discovering Symmetries," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5821–5827, 2022.

[20] D. Watkins-Valls, J. Varley, and P. Allen, "Multi-Modal Geometric Learning for Grasping and Manipulation," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 7339–7345.

[21] S. Wang, J. Wu, X. Sun, W. Yuan, W. T. Freeman, J. B. Tenenbaum, and E. H. Adelson, "3D Shape Perception from Monocular Vision, Touch, and Shape Priors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 1606–1613.

[22] E. Smith, R. Calandra, A. Romero, G. Gkioxari, D. Meger, J. Malik, and M. Drozdzal, "3D Shape Reconstruction from Vision and Touch," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 14 193–14 206.

[23] E. Smith, D. Meger, L. Pineda, R. Calandra, J. Malik, A. Romero Soriano, and M. Drozdzal, "Active 3D Shape Reconstruction from Vision and Touch," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[24] P. K. Murali, C. Wang, D. Lee, R. Dahiya, and M. Kaboli, "Deep Active Cross-Modal Visuo-Tactile Transfer Learning for Robotic Object Recognition," *IEEE Robotics and Automation Letters*, vol. 7, pp. 9557–9564, 10 2022.

[25] S. Suresh, Z. Si, J. G. Mangelson, W. Yuan, and M. Kaess, "Shapemap 3-d: Efficient shape mapping through dense touch and vision," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 7073–7080, 2022.

[26] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, Oct. 1999.

[27] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.

[28] B. Guo, J. Menon, and B. Willette, "Surface Reconstruction Using Alpha Shapes," *Computer Graphics Forum*, vol. 16, no. 4, pp. 177–190, 1997.

[29] O. Williams and A. Fitzgibbon, "Gaussian process implicit surfaces," in *Gaussian Processes in Practice*, 2006.

[30] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 5105–5114.

[31] T. Sakai, "On Riemannian manifolds admitting a function whose gradient is of constant norm," *Kodai Mathematical Journal*, vol. 19, no. 1, pp. 39 – 51, 1996.

[32] M. G. Crandall and P.-L. Lions, "Viscosity solutions of hamilton-jacobi equations," *Transactions of the American Mathematical Society*, vol. 277, no. 1, pp. 1–42, 1983.

[33] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[34] A. R. Smith, "Tint fill," ser. SIGGRAPH '79. New York, NY, USA: Association for Computing Machinery, 1979, p. 276–283.

[35] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes," 06 2018.

[36] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6D Object Pose Estimation by Iterative Dense Fusion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3343–3352.

[37] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2642–2651.

[38] X. Chen, Z. Dong, J. Song, A. Geiger, and O. Hilliges, "Category Level Object Pose Estimation via Neural Analysis-by-Synthesis," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVI 16*. Springer, 2020, pp. 139–156.

[39] P. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[40] B. Wen and K. Bekris, "BundleTrack: 6D Pose Tracking for Novel Objects without Instance or Category-Level 3D Models," *IEEE International Conference on Intelligent Robots and Systems*, pp. 8067–8074, 2021.

[41] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep Iterative Matching for 6D Pose Estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 683–698.

[42] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[43] A. Miller and P. Allen, "Graspit! A versatile simulator for robotic grasping," *IEEE Robotics Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.