2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Semantic keypoint extraction for scanned animals using multi-depth-camera systems

Raphael Falque[1], Teresa Vidal-Calleja[1], and Alen Alempijevic[1]

***Abstract*— Keypoint annotation in pointclouds is an important task for 3D reconstruction, object tracking and alignment, in particular in deformable or moving scenes. In the context of agriculture robotics, it is a critical task for livestock automation to work toward condition assessment or behaviour recognition. In this work, we propose a novel approach for semantic keypoint annotation in pointclouds, by reformulating the keypoint extraction as a regression problem of the distance between the keypoints and the rest of the pointcloud. We use the distance on the pointcloud manifold mapped into a radial basis function (RBF), which is then learned using an encoder-decoder architecture. Special consideration is given to the data augmentation specific to multi-depth-camera systems by considering noise over the extrinsic calibration and camera frame dropout. Additionally, we investigate computationally efficient non-rigid deformation methods that can be applied to animal pointclouds. Our method is tested on data collected in the field, on moving beef cattle, with a calibrated system of multiple hardware-synchronised RGB-D cameras.**

Keywords: 3D deep learning, keypoints annotation, multi-depth-camera systems, livestock

I. Introduction

Pointclouds are a common data representation generated by robotics perception systems equipped with depth sensors, such as stereo/depth cameras and LiDARs. Complex perception systems integrate several depth sensors allowing them to maximise the coverage of their surroundings [1], [2]. For frameworks using multiple depth sensors, it is common to align and merge the pointclouds using extrinsic calibration within the front-end data processing. The analysis of the merged pointcloud (e.g., semantic segmentation [3], or scene classification [4]) is then performed as part of what we call back-end process, e.g., through deep learning models for real-time processing. Most deep learning models for pointcloud analysis developed within the computer vision and computer graphics communities often overlook the noise generated during the data collection. However, this topic is at the core of robotics applications where both front-end and back-end processes have an impact on the model predictions accuracy and need to be considered. This paper studies how robustness to real-world data collection, such as calibration noise and camera frame dropping, can be built within the training of deep learning models for keypoint extraction in pointclouds.

Keypoints extraction and matching is an important task of 3D scene understanding for robotics applications such as tracking, shape retrieval, and registration. For instance, a direct application can be found for non-rigid deformation algorithms which require a good prior in order to solve the alignment of a template and a target as a gradient descent optimization. This surprisingly specific problem formulation finds applications in medical robotics for organs alignment [5], clothing manipulation [6], humans [7] and animals [8] body shape reconstruction. Furthermore, given a set of keypoints matching pairs, the rigid alignment of two shapes can be solved in a closed-form with singular value decomposition (SVD) [9] as an alternative to iterative closest point (ICP) methods [10].

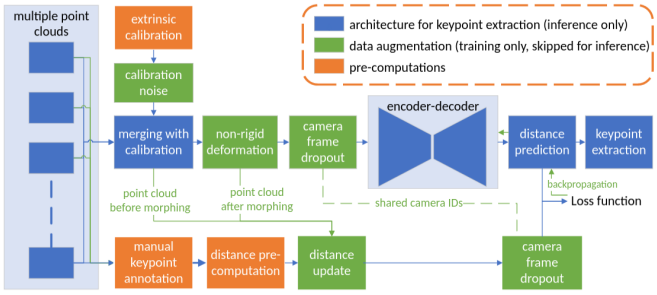

Fig. 1: Method overview: at inference time, the pointclouds are merged and passed into an encoder-decoder architecture such as Pointnet++ [16] or KPConv [17] to extract the keypoints (in blue). During training, a dataset is first manually annotated and the distance on the manifold is pre-computed (in orange). The weights of the encoder-decoder are then trained by using augmentation on the calibration, non-rigid deformation, and camera frame dropout (in green). In our experiments, the encoder-decoder inputs are $n \times 3$ points and the outputs are the $n \times 6$ distances to the 6 keypoints.

In the livestock industry, the automation of animal health assessment and monitoring is critical to ensure animal well-being and improved farming productivity. Working toward this direction, automatic pipelines are required to capture and identify key areas in animal bodies [11], [12], [13], [14], [15]. Real-time 3D shape and model reconstruction have been identified as crucial steps towards explicit segmentation of animal body regions, which are meaningful for cattle posture or behaviour recognition [13]. We consider the challenge of extracting keypoints of cattle bodies using a perception system containing synchronised RGB-D cameras capturing multiple pointclouds for each animal, shown in Figure 4.

We formulate the semantic keypoints extraction from pointclouds as a regression problem with a segmentation encoder-decoder architecture used as the main building

[1]The authors are with the Robotics Institute, University of Technology Sidney, Australia raphael.guenot-falque@uts.edu.au

block [16], [17]. In contrast with [18], where an encoder-decoder is stacked into an MLP to solve the regression problem of keypoint extraction, we do not modify the encoder-decoder architecture and formulate the regression problem by redefining how the mean squared error (MSE) loss function is computed. A bloc diagram of the method is shown in Figure 1. As an additional focus of this paper, we investigate how data augmentation can be adapted to our problem by considering the integration of augmentation specific to multi-depth-camera data and non-rigid deformation formulated in the form of shear transformation matrices.

The main contributions are: (1) we formulate the keypoint annotation problem of pointclouds as a regression of the distance on the manifold mapped into an RBF function. The regression of the distance on the manifold is learned with en encoder-decoder architecture which essentially solves the problem of keypoint matching as a semantic problem (similarly to segmentation problems). We argue that any encoder-decoder model can be used as long as the architecture is able to learn on pointcloud data. (2) We consider a complex data augmentation process that includes noise inherent to multi-depth-camera systems, through the efficient use of non-rigid deformation methods for pointclouds, and a computationally efficient approximation of the distance on the manifold.

The implementation is available at https://github.com/rFalque/semantic_keypoints_extraction.

II. Related work

Pose and body shape estimation from 2D images is a well-studied topic thanks to the large datasets available on the internet (such as COCO [19]). For humans, the pose can be estimated with a deep neural network and the shape can then be optimised over the skeleton pose [7], [20]. Similar methods have been applied to various animals [21] and refined for specific breeds such as lions [22] and zebras [23]. For 2D images, large datasets with ground truth poses are available, making the use of deep learning applications possible [24]. In cases where the ground trough is not available, 2D images and annotations can still be generated from synthetics animated animals [25], however, these datasets have a lack of variety with respect to the 3D models of the animals.

In the field of computer graphics, the problem of shape correspondences [26] studies the point-wise association between different shapes (generally defined as triangular meshes) and therefore is a generalisation of the keypoints annotation problems. Amongst the most significant work, the study of the shape spectral decomposition and the spectral correspondence has been formulated in a framework called *functional maps* [27]. This work has later been extended for partial shapes [28], and deep learning approaches have been developed to leverage the spectral domain in a learning context [29], [30].

However, for robotics applications, where LIDAR and RGB-D cameras are used for scanning the environment, working with meshes is often a cumbersome task. Standard mesh reconstruction methods [31], [32] perform poorly on irregular, noisy, and non-closed pointclouds. As a result, working directly on pointclouds and bypassing the mesh reconstruction is often preferable. While functional maps could be adapted to work directly on pointclouds, thanks to robust Laplacian operator [33], they lack robustness (e.g., functional maps can have catastrophic failures) and are over-engineered for the problem of keypoints correspondences (i.e., extracting few keypoints is a significantly simpler task than solving the full shape correspondence).

In the case of keypoint annotation of 3D pointclouds, research is sparser due to the complexity of gathering and annotating data. Traditional methods usually consider the keypoints extraction from pointcloud data by extracting a 3D skeleton [34], [35], [36], [37], from which voting systems can establish correspondences [38]. Deep learning methods have recently been developed to predict the 3D skeleton from pointclouds [37].

Most similar to our method is the recent work on unsupervised keypoint annotations from pointclouds using deep learning models. Architectures for learning on pointclouds are constantly evolving; the most widespread architectures developed over the last years are Pointnet [39], Pointnet++ [16], and Kp-CONV [17]. [18] used Pointnet stacked with additional MLP used to transform the Pointnet output into a regression problem. [40] focus on the loss functions capable of handling multiple human annotations and consider the keypoint extraction as a classification problem. [41] propose a different architecture that builds upon Pointnet and solves the problem of keypoint annotation as an unsupervised problem using symmetric linear basis shapes. Similarly, [42] considers the unsupervised problem. They use a Pointnet++ encoder and a custom-made decoder that generates the skeleton from the keypoints, which are then aligned between instances.

III. Methodology

A. Problem Definition

Given the partial 3D shape of an animal represented as a set of pointclouds captured by C depth-cameras $\{\boldsymbol{P}_1, \dots, \boldsymbol{P}_C\}$ concatenated into a single pointcloud such as $\boldsymbol{P} = \bigcup_{c=1}^{C} \boldsymbol{P}_c$ where $\boldsymbol{P} = \{\boldsymbol{p}_1, ..., \boldsymbol{p}_n\}$ and $\boldsymbol{p}_i \in \mathbb{R}^3$, we aim at estimating the m ordered keypoints $\boldsymbol{N} = \{\boldsymbol{n}_1, ..., \boldsymbol{n}_m\}$, where $\boldsymbol{n}_j \in \boldsymbol{P}$. The keypoints are ordered in the sense that they are systematically annotated in the same order, such as {right rear leg, right front leg, hip, ...}. Therefore, the keypoints have a semantic meaning. A flowchart of the methodology is shown in Figure 1.

B. Architecture and loss function

We propose to solve the keypoint extraction as a regression problem, where we learn the distance between each point from the pointcloud to each annotated point, resulting in m distance vectors of size n. Given this distance, the keypoints could then be obtained by taking the argument of the distance vectors minimum.

From a machine learning perspective, we formulate our problem as a mapping between the $n \times 3$ input matrix $\boldsymbol{P}$

input and the $n \times m$ output matrix $\hat{\boldsymbol{D}}$. This naturally fits an encoder-decoder architecture, where we aim to predict features for each point from the input. The pointcloud encoder-decoder architectures are usually designed to solve semantic segmentation problems, where the neural network predicts the probability of each point belonging to a specific class. The formulation of an encoder-decoder into a segmentation problem is only enforced by squashing the network prediction into a probability, e.g., usually using a sigmoid function, and by backpropagating the expected class probability through the loss function.

To change the classical segmentation problem into a regression problem, we start by computing the distance on the pointcloud manifold[1], $\boldsymbol{D}^g_{i,j} \in \mathbb{R}^{n \times m}$, between the i^{th} point and the j^{th} keypoint using the heat kernel method [43]. The heat kernel method can be implemented on different data structures (e.g., mesh, tetrahedral, pointclouds) as long as a Laplacian operator can be computed on this data structure. In our case, we use the *tufted Laplacian* [33], which is robust to noisy data and particularly adapted for pointclouds generated by depth cameras.

The computation of the distance on the manifold is computationally expensive and is, therefore, performed as a pre-processing step (i.e., prior to training the network to avoid this computation for all epochs). The geodesic distance is then mapped into a radial basis function (RBF) with a Gaussian kernel to have a localised heat map as follows:

$$\boldsymbol{D}_{i,j} = e^{-\epsilon\left(\boldsymbol{D}^g_{i,j}\right)^2} \tag{1}$$

where ϵ controls the sharpness of the annotation and should be chosen with respect to the scale of the pointcloud (in our case $\epsilon = 10$). Mapping the geodesic distance into an RBF serves multiple purposes. Firstly, we believe that it simplifies the learning process by reducing the areas where the distance has to be learned, i.e., only the distance in a local area needs to be learned instead of the full shape. Secondly, it squashes the learned features between 0 and 1, allowing us to use a standard mean square error (MSE) loss function as,

$$l = \frac{1}{nm} \sum_{i,j} (\hat{\boldsymbol{D}}_{i,j} - \boldsymbol{D}_{i,j})^2, \tag{2}$$

without having to modify a network designed for semantic segmentation (i.e., an architecture with an encoder-decoder). Finally, it also allows us to make some specific assumptions in the data augmentation, as discussed in Section III-D. Given Equation (1), the position of the keypoints is now obtained by finding the argument that maximises the learned distances (i.e., *argmax* of prediction).

In this work, we argue that an encoder-decoder architecture can be used interchangeably as long as it offers the capability to learn on pointclouds. Therefore, to demonstrate the flexibility of our approach, in the experiments, we test our method on both Pointnet++ [16] and Kp-CONV [17] which are commonly regarded as pillars for learning on pointclouds.

[1]In the context of a pointcloud, the notion of a manifold and geodesic distance can be built by analysing the curvature in local neighbourhoods, e.g., using the Laplace-Beltrami operator.

C. Network's inputs data augmentation

The proposed data augmentation considers two main approaches: first, we analyse the data augmentation specific to multi-depth-camera systems, secondly, we consider the data augmentation allowing us to augment the training set.

1) Multi-depth-camera system augmentation: In the literature, many of the public keypoint datasets on pointclouds are built by sampling points from meshes. For applications on pointclouds obtained from multiple depth cameras, several real-world challenges have to be considered. The camera rig is portable and needs to be mounted, calibrated, and unmounted for each data collection. Therefore, the deep learning model needs to be reliable to a slight change of calibration. This robustness is also important in the case of slight camera motion arising from the vibration of the camera system (in our case, this scenario is particularly relevant due to seldom, though very forceful, kicks from animals onto the structure used for data collection). Data augmentation is formally defined as

$$\forall c \in \{1, \ldots, C\}, \boldsymbol{P}_c = \boldsymbol{P}_c \boldsymbol{R} + \boldsymbol{t} \tag{3}$$

with $\boldsymbol{R}$ a rotation matrix in $SO(3)$ and $\boldsymbol{t}$ a translation vector that add calibration noise.

Additionally, to enforce robustness to calibration errors, the camera frames dropout has to be considered. This can occur due to network traffic between computers in the system or latency in data acquisition due to the depth camera's auto-exposure. This problem has been studied in the context of multi-view CNN by embedding a *dropview* layer within the architecture at training time [44]. This can be adapted for encoder-decoder by dropping some of the pointclouds from P such as :

$$\boldsymbol{P} = \bigcup_{c=1}^{C} \begin{cases} \boldsymbol{P}_c & \text{if } x \leq 0.95 \mid x \sim U([0,1]) \\ \emptyset & \text{otherwise} \end{cases} \tag{4}$$

where x is the probability of dropping the camera pointcloud. Keeping the pointcloud dropout within the data loader as part of the data augmentation allows our method to stay agnostic to the model architecture.

2) Geometric transformation: Similarly to [16], [45], we propose to use changes of scale which can be formulated as a geometric transformation. We also consider random flipping similarly to [46], [45]. Following [47, Sec. 4.7.6], the scale and random flipping transformation matrices are defined as:

$$\boldsymbol{T}_{\text{scale}} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \boldsymbol{T}_{\text{flip}} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

where s_x, s_y, and s_z are drawn from a normal distribution $\mathcal{N}(1, 0.1)$. f corresponds to a random reflection of the shape with respect to the YZ symmetry plane, such as:

$$f = \begin{cases} -1, & \text{if } x > 0.5 \mid x \sim U([0,1]) \\ 1, & \text{otherwise.} \end{cases} \tag{6}$$

Additional augmentation methods can be obtained using geometric transformations such as translation and rotations. The networks for pointclouds usually learn on structures defined from local neighbourhoods and are therefore invariant to global translation. Rotation transformations have been used in the literature [39], [16], [45]. However, in our case the dedicated perception system scans animals in a similar orientation as shown in Figure 4, therefore random rotations are out of the scope of our work.

3) Non-rigid deformation: Having control of the animal stance, i.e., how they stand during the data collection, is extremely challenging. As an alternative, one would rather learn to perform the keypoints annotation for all possible animal poses. This would ideally be obtained through the application of non-rigid deformation of the animal shape by simulating the motion of the animal.

In the field of non-rigid deformation, this is typically performed by rigging the shape of the animal (building a virtual skeleton) and animating its motion [48], [49]. In our case, such deformation is unobtainable as the virtual skeleton is unavailable. Further, this is to some extent what this paper is aiming to solve (with the difference that we are aiming at finding points on the pointcloud rather than nodes from the skeleton). Alternatively, 3D shapes can be deformed without a virtual skeleton as a prior by maximising the local rigidity of the shape while trying to minimise the error on "handles" in motion (handles are points selected from the shape). While near real-time implementations of such methods exist for meshes [50], [51], the equivalent for pointclouds is significantly slower [52]. Regardless of the computation time, the deformation requires the non-rigid motion to be parameterised, which is also challenging.

As an alternative to these traditional methods, we propose to use shear matrices [47, Sec. 4.7.6] to simulate the motion of the animals stepping sideways and leaning forward / backward. These shear matrices are defined as:

$$\boldsymbol{T}_{\text{sideway}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ h_s & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \boldsymbol{T}_{\text{forward}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & h_f & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

where $h_s = \tan(\theta_s)$ and $\theta_s \sim \mathcal{N}(0, \frac{\pi}{20})$ the shearing angle with respect to the x axis. Similarly, $h_f = \tan(\theta_f)$ and $\theta_f \sim \mathcal{N}(0, \frac{\pi}{20})$ the shearing angle with respect to the z axis. Formulating the motion as additional transformation matrices allows us to obtain a non-rigid deformation matrix that can be easily parameterised and applied simultaneously as to the other data augmentation discussed in Section III-C.2.

4) Application of the data augmentation: Given all the different geometric transformation matrices, we can apply this transformation to the points of the shape and the annotated keypoints as follow:

$$\boldsymbol{P}^* = \boldsymbol{T}_{\text{scale}}\boldsymbol{T}_{\text{flip}}\boldsymbol{T}_{\text{sideway}}\boldsymbol{T}_{\text{forward}}\boldsymbol{P} \tag{8}$$

$$\boldsymbol{N}^* = \boldsymbol{T}_{\text{scale}}\boldsymbol{T}_{\text{flip}}\boldsymbol{T}_{\text{sideway}}\boldsymbol{T}_{\text{forward}}\boldsymbol{N} \tag{9}$$

A visualisation of the cloud data augmentation is proposed in Figure 2.

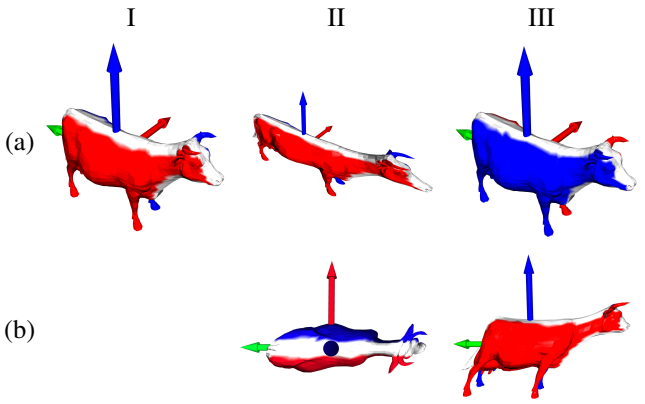

Fig. 2: Data augmentation applied to a mesh where the parameters are amplified for visualisation. The coordinate system is defined by the x, y, and z axes respectively displayed as red, green, and blue arrows. The original shape in I.(a), the forward shearing in II.(a), the side-way shearing in III.(a), the non-isotropic scaling in II.(b), and the random flipping in III.(b).

D. Network's outputs data augmentation

In the case of semantic segmentation, the data augmentation rarely impacts the network outputs and is often overlooked in the literature (e.g., for semantic segmentation, point labels are not augmented). However, in the case of the regression of the distance on the manifold, the input non-rigid deformation (i.e., $\boldsymbol{T}_{\text{scale}}$, $\boldsymbol{T}_{\text{forward}}$, and $\boldsymbol{T}_{\text{sideway}}$) should change the predicted distances and the data augmentation has to be considered during the loss function computation. Ideally, the distance on the manifold with respect to the annotated point would be recomputed. However, given the computational complexity of calculating the distance on the manifold, we propose to update the distance on the manifold with respect to the difference in the Euclidean distance between $\boldsymbol{P}$ and $\boldsymbol{P}^*$ from Equation (8). More formally, the distance on the manifold is first approximated as

$$\boldsymbol{D}^{g^*}_{i,j} = \boldsymbol{D}^{g}_{i,j} + \boldsymbol{D}^{e^*}_{i,j} - \boldsymbol{D}^{e}_{i,j}, \tag{10}$$

and the RBF distances are updated as follows,

$$\boldsymbol{D}^{*}_{i,j} = e^{-\epsilon\left(\boldsymbol{D}^{g^*}_{i,j}\right)^2} \tag{11}$$

where $\boldsymbol{D}^e = d(\boldsymbol{P}, \boldsymbol{N})$ and $\boldsymbol{D}^{e^*} = d(\boldsymbol{P}^*, \boldsymbol{N}^*)$ are the Euclidean distances before and after equation (8). $\boldsymbol{D}^*$ is the updated distance on the manifold mapped into an RBF.

The case of the flip transform, $\boldsymbol{T}_{\text{flip}}$, has to be handled differently. The transformation applies a symmetry on the YZ plane to the shape and therefore, it changes the semantic meaning of the annotation (e.g., the left legs become the right legs within the transformation). To update the outputs, a simple reshuffling of the distance vector to the annotation is necessary. Using pseudo-code with 0 indexing, the update of the outputs is reshuffled as

$$\boldsymbol{D}^* = \boldsymbol{D}^*[:, [5, 4, 2, 3, 1, 0]]. \tag{12}$$

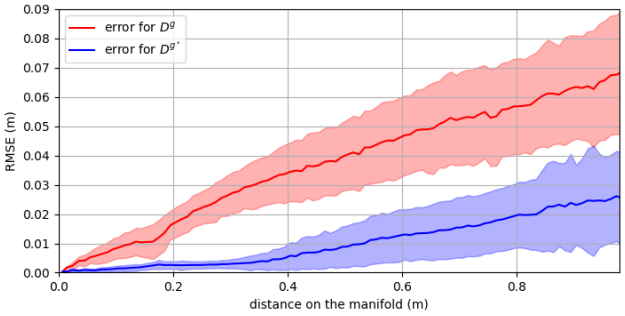

Fig. 3: Mean error and standard deviation for $\boldsymbol{D}^g$ and $\boldsymbol{D}^{g^*}$ versus the recomputed geodesic distance following (8). As shown in the plot, while the approximation considerably reduces the error, our approximation is only valid in the vicinity of the keypoints. Therefore, the use of the RBF mapping in (11) makes this approximation acceptable.

Finally, following the camera frame dropout implemented in Equation (4), $\boldsymbol{D}^*$ is updated to drop the element with indexes corresponding to the dropped cameras.

IV. Experimental Results

To demonstrate the validity of the approach, we use real data taken in a cattle research facility with an in-house built system. The data collection has been performed at the Tullimba feedlot research facility, where a race (i.e., corridor cattle walk-through) has been modified to integrate a robotics system capable of scanning cattle, as shown in Figure 4. The robotics perception system is designed with 17 RGB-D Realsense D435 cameras extrinsically calibrated with a checkerboard prior to the data collection. Cameras are synchronised using a hardware trigger allowing the capture of an animal's 3D shape in a single snapshot. The data collection was undertaken on two different occasions, with 274 animals scanned on the first date and a different cohort of 136 animals scanned on the second date. On each occasion, the system is assembled and calibrated. Therefore, we consider the data collections as two independent datasets with the larger one used as a training set and the smaller one used as a testing set. This allows us to test if the model overfits a specific dataset or if it can be used across different datasets regardless of the calibration.

For each animal, the pointclouds are stitched together using extrinsic calibration. The background is removed to isolate the animal, and outliers are filtered using radius outlier removal (ROR) and statistical outlier removal (SOR) filters [53] resulting in $\boldsymbol{P}$. All the pointclouds from both datasets have then been annotated manually with 6 keypoints (the top of each leg, the hip, and the neck), and the geodesic distance [43] between the annotated keypoints and the rest of the pointcloud has been pre-computed. As a result, every instance of our dataset can be considered as a tuple of three matrices $\{\boldsymbol{P}, \boldsymbol{N}, \boldsymbol{D}^g\}$.

For the training, we investigated both the Pointnet++ and Kp-CONV architectures while using standard tools and

Fig. 4: Data collection system used to scan cattle: the perception system is made of multiple RGB-D Realsense cameras that are positioned to allow capturing the full shape of the animals in a single snapshot (each side of the camera has two rows of four cameras and one camera is located above). Overlap between the cameras allows for extrinsic calibration and ensures that the data collection is more reliable.

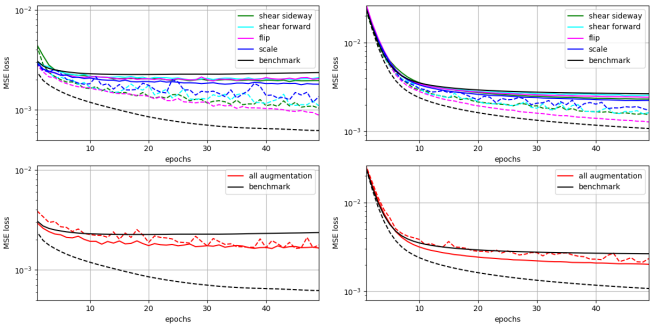

Fig. 5: Training loss (dashed) and testing loss (plain) in a log scale versus the number of epochs for the Pointnet++ [16] (left) and Kp-CONV [17] (right) networks. The top graphs are an ablation study for each data augmentation, showing they reduce the gap between training and testing loss functions while improving the loss for the testing set when compared to the benchmark with no data augmentation (in black). Once combined, the data augmentation avoids overfitting the training set which can be seen as the loss function on the testing set is below the training set. This is particularly relevant for the Pointnet++ architecture, where the network stops learning after 20 epochs.

default hyper-parameters during the training stage. While this is not the focus of the paper, we provide the hyper-parameters for the sake of algorithmic reproducibility. The optimisation of the network parameters has been done using stochastic gradient descent (SGD) with a learning rate of 0.01 using a momentum of 0.9 [2].

The error on the Euclidean distance between the annotated points and the points predicted by both networks is reported in Table I and a study on the sampling size of the training set is given in Table II. The data augmentation allows for maintaining good performances while being able to learn on a relatively small dataset. It should be noted that the remaining error does not account for the error of manual annotations.

[2]The learning rate for Kp-CONV could have been increased as it learned significantly slower but smoother than Pointnet++.

TABLE I: RMSE of the keypoint prediction (in cm) for Pointnet++ and Kp-Conv with/without data augmentation. The error decreases significantly with data augmentation. To put these results in perspective, the bounding box diagonal of a cow scan would be approximately 2m.

	w.o. augmentation (50 epochs)	w. augmentation (50 epochs)	w. augmentation (150 epochs)
Pointnet++	7.02	5.76	5.60
Kp-CONV	7.15	6.41	5.98

TABLE II: Study of the data augmentation effectiveness with respect to the training set size. The errors reported are the MSE (at the power 10^{-3}) of the updated distance on the manifold on the testing set (as defined as in Equation (2)). The results show that the reduction of samples number in the training set has a marginal impact on the network performance when the data augmentation is used for both Pointnet++ and KpCONV.

	Pointnet++		Kp-CONV	
augmentation	with	without	with	without
200 samples	0.90	1.01	1.12	1.32
100 samples	0.88	1.31	1.17	1.50
50 samples	0.94	1.41	1.13	1.76

In Figure 5, we perform an ablation study of the data augmentation and show that every non-rigid deformation data augmentation reduces the gap between the training and testing set. When we combine all the augmentation methods from Section III-C.2, we can see that both networks do not overfit the training set in the first 50 epochs (which they would most likely do after a certain quantity of epochs). This is a significant improvement when compared to the case without data augmentation (in black), particularly for Pointnet++, which starts overfitting the training set after the first epoch.

A study case of the camera dropout is shown in Figure 6. The distance prediction from Pointnet++ trained without camera frame dropout (shown in the left) is overconfident in the distance measure. After including the camera frame dropout (shown on the right), the maximum distance measure drops from 0.79 to 0.45, showing that the keypoint is far from the argmax of the prediction. Therefore, the predicted distance of the manifold can be used as a quantification of the keypoints uncertainty.

V. Conclusion

In this paper, we propose a simple yet effective method for annotating keypoints using a supervised deep learning approach. We reformulate the problem of keypoints extraction into a regression problem by inferring the distance on the manifold to keypoints. We study the relevant data augmentation methods for multi-depth-camera systems and efficient non-rigid deformation methods. Our method shows relatively good results when compared to the size of an animal's complete scan while being trained on a relatively small dataset. The experiments show that the model can be trained on a relatively small dataset while being robust to camera drops and calibration noise.

Our approach uses a single set of manual annotations for each pointcloud, the error on human annotation could be mitigated by using a consensus of expert annotations similarly to [40].

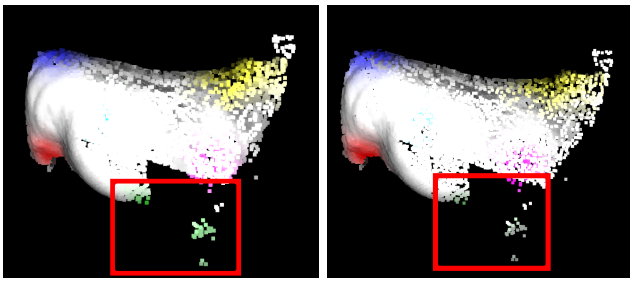

Fig. 6: Sample of pointcloud with dropped frame. The camera frame dropout in the data augmentation allows for avoiding being overconfident in the prediction of the green keypoints. The maximum value for the predicted distance on the manifold drops from 0.79 to 0.45. As a result, we can then use the predicted distance on the manifold as a measure of the keypoints uncertainty.

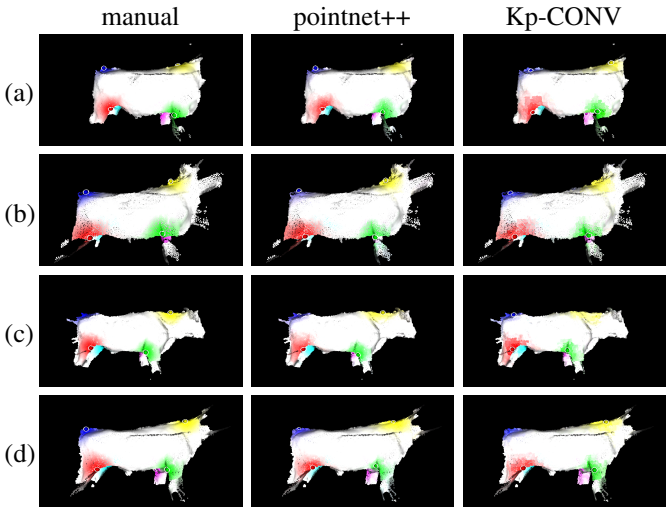

Fig. 7: Keypoints manually annotated, and predicted by Pointnet++ and Kp-CONV are highlighted in white. The cow's pointclouds are coloured with the distance on the manifold to each keypoints. In (a) and (b), the two first samples from the testing set and in (c) and (d) the worst performing predictions. The keypoints with the largest error are the neck which is harder to annotate manually consistently.

Acknowledgment

This paper is supported by funding from the Australian Government Department of Agriculture and Water Resources as part of its Rural R&D for Profit program, MLA grant number V.RDP.2005.

REFERENCES

[1] L. Heng, B. Choi, Z. Cui, M. Geppert, S. Hu, B. Kuan, P. Liu, R. Nguyen, Y. C. Yeo, A. Geiger *et al.*, "Project autovision: Localization and 3d scene perception for an autonomous vehicle with a multi-camera system," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4695–4702.

[2] M. Brandao, R. Figueiredo, K. Takagi, A. Bernardino, K. Hashimoto, and A. Takanishi, "Placing and scheduling many depth sensors for wide coverage and efficient mapping in versatile legged robots," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 431–460, 2020.

[3] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9297–9307.

[4] J. Ren, L. Pan, and Z. Liu, "Benchmarking and analyzing point cloud classification under corruptions," *arXiv preprint arXiv:2202.03377*, 2022.

[5] Y. Zhang, R. Falque, L. Zhao, S. Huang, and B. Hu, "Deep learning assisted automatic intra-operative 3d aortic deformation reconstruction," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2020, pp. 660–669.

[6] Z. Huang, X. Lin, and D. Held, "Mesh-based dynamics with occlusion reasoning for cloth manipulation," *arXiv preprint arXiv:2206.02881*, 2022.

[7] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," *ACM transactions on graphics (TOG)*, vol. 34, no. 6, pp. 1–16, 2015.

[8] N. Rüegg, S. Zuffi, K. Schindler, and M. J. Black, "Barc: Learning to regress 3d dog shape from images by exploiting breed information," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3876–3884.

[9] O. Sorkine-Hornung and M. Rabinovich, "Least-squares rigid motion using svd," *Computing*, vol. 1, no. 1, pp. 1–5, 2017.

[10] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, 2001, pp. 145–152.

[11] J. H. Gardenier, "How now lame cow: Automatic lameness assessment for dairy cattle with 3d sensors," Ph.D. dissertation, Australian Centre for Field Robotics, 2020. [Online]. Available: https://hdl.handle.net/2123/23218

[12] X. Kang, X. D. Zhang, and G. Liu, "A review: Development of computer vision-based lameness detection for dairy cows and discussion of the practical applications," *Sensors*, vol. 21, no. 3, p. 753, 2021.

[13] Y. Qiao, H. Kong, C. Clark, S. Lomax, D. Su, S. Eiffert, and S. Sukkarieh, "Intelligent perception for cattle monitoring: A review for cattle identification, body condition score evaluation, and weight estimation," *Computers and Electronics in Agriculture*, vol. 185, p. 106143, 2021.

[14] M. J. McPhee, B. J. Walmsley, B. Skinner, B. Littler, J. Siddell, L. Cafe, J. Wilkins, V. Oddy, and A. Alempijevic, "Live animal assessments of rump fat and muscle score in angus cows and steers using 3-Dimensional imaging," *Journal of Animal Science*, vol. 95, no. 4, pp. 1847–1857, 2017.

[15] A. Bercovich, Y. Edan, V. Alchanatis, U. Moallem, Y. Parmet, H. Honig, E. Maltz, A. Antler, and I. Halachmi, "Development of an automatic cow body condition scoring using body shape signature and fourier descriptors," *Journal of dairy science*, vol. 96, no. 12, pp. 8047–8059, 2013.

[16] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[17] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6411–6420.

[18] L. Ge, Z. Ren, and J. Yuan, "Point-to-point regression pointnet for 3d hand pose estimation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 475–491.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[20] A. A. Osman, T. Bolkart, and M. J. Black, "Star: Sparse trained articulated human body regressor," in *European Conference on Computer Vision*. Springer, 2020, pp. 598–613.

[21] S. Zuffi, A. Kanazawa, D. Jacobs, and M. J. Black, "3D menagerie: Modeling the 3D shape and pose of animals," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*. IEEE, Jul. 2017, pp. 5524–5532.

[22] S. Zuffi, A. Kanazawa, and M. J. Black, "Lions and tigers and bears: Capturing non-rigid, 3D, articulated shape from images," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018, pp. 3955–3963.

[23] S. Zuffi, A. Kanazawa, T. Berger-Wolf, and M. J. Black, "Three-D safari: Learning to estimate zebra pose, shape, and texture from images "in the wild"," in *International Conference on Computer Vision*. IEEE, Oct. 2019, pp. 5358–5367. [Online]. Available: https://arxiv.org/pdf/1908.07201.pdf

[24] X. L. Ng, K. E. Ong, Q. Zheng, Y. Ni, S. Y. Yeo, and J. Liu, "Animal kingdom: A large and diverse dataset for animal behavior understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 19 023–19 034.

[25] J. Mu, W. Qiu, G. D. Hager, and A. L. Yuille, "Learning from synthetic animals," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 386–12 395.

[26] O. Van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or, "A survey on shape correspondence," in *Computer graphics forum*. Wiley Online Library, 2011, pp. 1681–1707.

[27] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas, "Functional maps: a flexible representation of maps between shapes," *ACM Transactions on Graphics (ToG)*, vol. 31, no. 4, pp. 1–11, 2012.

[28] E. Rodolà, L. Cosmo, M. M. Bronstein, A. Torsello, and D. Cremers, "Partial functional correspondence," in *Computer graphics forum*. Wiley Online Library, 2017, pp. 222–236.

[29] O. Litany, T. Remez, E. Rodola, A. Bronstein, and M. Bronstein, "Deep functional maps: Structured prediction for dense shape correspondence," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5659–5667.

[30] S. Attaiki, G. Pai, and M. Ovsjanikov, "Dpfm: Deep partial functional maps," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 175–185.

[31] 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, 1999.

[32] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, pp. 1–13, 2013.

[33] N. Sharp and K. Crane, "A Laplacian for Nonmanifold Triangle Meshes," *Computer Graphics Forum (SGP)*, vol. 39, no. 5, 2020.

[34] A. Tagliasacchi, H. Zhang, and D. Cohen-Or, "Curve skeleton extraction from incomplete point cloud," in *ACM SIGGRAPH 2009 papers*, 2009, pp. 1–9.

[35] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, and Z. Su, "Point cloud skeletons via laplacian based contraction," in *2010 Shape Modeling International Conference*. IEEE, 2010, pp. 187–197.

[36] A. Tagliasacchi, I. Alhashim, M. Olson, and H. Zhang, "Mean curvature skeletons," in *Computer Graphics Forum*. Wiley Online Library, 2012, pp. 1735–1744.

[37] C. Lin, C. Li, Y. Liu, N. Chen, Y.-K. Choi, and W. Wang, "Point2skeleton: Learning skeletal representations from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4277–4286.

[38] O. K.-C. Au, C.-L. Tai, D. Cohen-Or, Y. Zheng, and H. Fu, "Electors voting for fast automatic shape correspondence." in *Comput. Graph. Forum*. Citeseer, 2010, pp. 645–654.

[39] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[40] Y. You, Y. Lou, C. Li, Z. Cheng, L. Li, L. Ma, C. Lu, and W. Wang, "Keypointnet: A large-scale 3d keypoint dataset aggregated from numerous human annotations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 647–13 656. [Online]. Available: https://arxiv.org/pdf/2002.12687.pdf

[41] C. Fernandez-Labrador, A. Chhatkuli, D. P. Paudel, J. J. Guerrero, C. Demonceaux, and L. V. Gool, "Unsupervised learning of category-

specific symmetric 3d keypoints from point sets," in *European Conference on Computer Vision*. Springer, 2020, pp. 546–563. [Online]. Available: https://arxiv.org/pdf/2003.07619.pdf

[42] R. Shi, Z. Xue, Y. You, and C. Lu, "Skeleton merger: an unsupervised aligned keypoint detector," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 43–52. [Online]. Available: https://arxiv.org/pdf/2103.10814.pdf

[43] K. Crane, C. Weischedel, and M. Wardetzky, "The heat method for distance computation," *Commun. ACM*, vol. 60, no. 11, pp. 90–99, Oct. 2017.

[44] Z. Huang, Z. Zhao, H. Zhou, X. Zhao, and Y. Gao, "Deepccfv: Camera constraint-free multi-view convolutional neural network for 3d object retrieval," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 8505–8512.

[45] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9277–9286.

[46] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 918–927.

[47] P. Schneider and D. H. Eberly, *Geometric tools for computer graphics*. Elsevier, 2002.

[48] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan, "Geometric skinning with approximate dual quaternion blending," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 4, pp. 1–23, 2008.

[49] A. Jacobson, I. Baran, J. Popovic, and O. Sorkine, "Bounded biharmonic weights for real-time deformation." *ACM Trans. Graph.*, vol. 30, no. 4, p. 78, 2011.

[50] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Symposium on Geometry processing*, vol. 4, 2007, pp. 109–116.

[51] A. Jacobson, I. Baran, L. Kavan, J. Popović, and O. Sorkine, "Fast automatic skinning transformations," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–10, 2012.

[52] R. W. Sumner, J. Schmid, and M. Pauly, "Embedded deformation for shape manipulation," in *ACM siggraph 2007 papers*, 2007, pp. 80–es.

[53] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," *KI-Künstliche Intelligenz*, vol. 24, no. 4, pp. 345–348, 2010.