




# Local Reflectional Symmetry Detection in Point Clouds Using a Simple PCA-Based Shape Descriptor

Lukáš Hruďa<sup>1</sup> <sup>a</sup>, Ivana Kolingerová<sup>1</sup> <sup>b</sup> and David Podgorelec<sup>2</sup> <sup>c</sup>

<sup>1</sup>Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Univerzitní 8, 301 00 Plzeň, Czech Republic

<sup>2</sup>Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška cesta 46, 2000 Maribor, Slovenia

Keywords: Symmetry, Local, Reflectional, PCA, Descriptor.

Abstract: Symmetry is a commonly occurring feature in real world objects and its knowledge can be useful in various applications. Different types of symmetries exist but we only consider the reflectional symmetry which is probably the most common one. Multiple methods exist that aim to find the global reflectional symmetry of a given 3D object and although this task on its own is not easy, finding symmetries of objects that are merely parts of larger scenes is much more difficult. Such symmetries are often called local symmetries and they commonly occur in real world 3D scans of whole scenes or larger areas. In this paper we propose a simple PCA-based local shape descriptor that can be easily used for potential symmetric point matching in 3D point clouds and, building on previous work, we present a new method for detecting local reflectional symmetries in 3D point clouds which combines the PCA descriptor point matching with the density peak location algorithm. We show the results of our method for several real 3D scanned scenes and demonstrate its computational efficiency and robustness to noise.


## 1 INTRODUCTION AND RELATED WORK


Many objects in real world contain some form of symmetry. By symmetry we understand a geometric transformation that leaves a given object unchanged excluding identity. Having a 3D object  $X$  a transformation  $T \neq I$  is a symmetry of  $X$  if  $T(X) = X$ , i.e. when after  $T$  is applied on  $X$  each point of the transformed  $X$  ends up in any point of the original  $X$ . There are different types of symmetry according to which types of transformations are considered. In this paper we only consider reflectional symmetry where the transform  $T$  is a reflection over a plane and can be represented by the reflection plane itself - the symmetry plane. Also, in reality, there is never a perfect symmetry which means that we are ultimately interested in approximate reflectional symmetries where the transform does not match the object onto itself perfectly but only approximately. The term *approximate symmetry*, however, does not seem to have an


exact formal definition.

The knowledge of symmetry has various applications, mostly in computer vision, such as incomplete object reconstruction (Sipiran et al., 2014), object alignment (Li et al., 2016) or symmetrical editing (Martinet et al., 2006). But before any application is possible the symmetries first need to be found.

The field of symmetry detection has advanced a lot in the past decades providing numerous methods that aim to find the symmetry plane(s) of a given input object. Such a task is difficult on its own but especially the modern methods are able to provide a robust, reliable and fast solutions that are often capable of finding visually plausible symmetry planes in objects with very weak reflectional symmetries or with missing geometry (like those in Fig. 2), they can work with various types of data and can often solve the task at hand in matters of seconds, sometimes even hundreds or tens of milliseconds. Examples of such methods are (Martinet et al., 2006; Combès et al., 2008; Sipiran et al., 2014; Li et al., 2016; Nagar and Raman, 2020; Hruďa et al., 2022; Žalik et al., 2022). Still, like in most fields, there is room for further improvements since even the most advanced methods cannot handle everything.

<sup>a</sup>  <https://orcid.org/0000-0002-9477-7464>

<sup>b</sup>  <https://orcid.org/0000-0003-4556-2771>

<sup>c</sup>  <https://orcid.org/0000-0002-0701-9201>

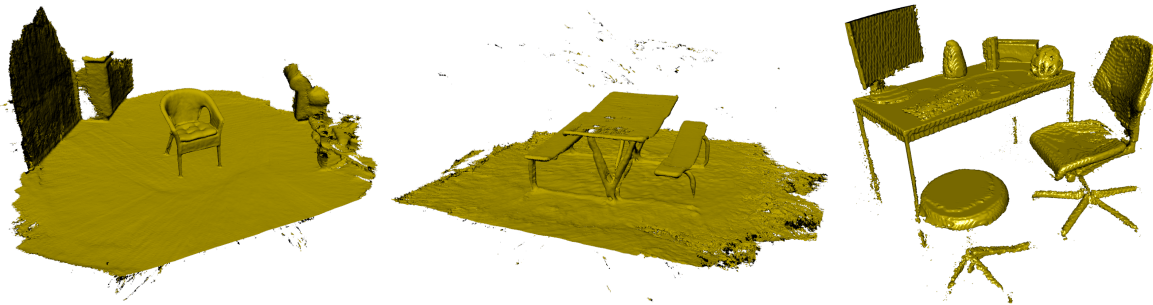


Figure 1: Examples of scenes that exhibit one or more reflectional symmetries in their subparts - local symmetries (data source: (Funk et al., 2017)).

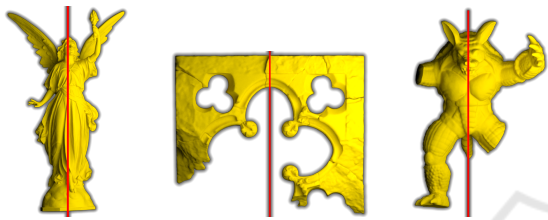


Figure 2: Examples of weakly symmetrical objects with plausible symmetry planes marked by the red line (source: (Hruda et al., 2022)).

However, in this case the task is somewhat simplified by the fact that the whole object is anticipated to be symmetric and the methods aim to find the plane that best captures its symmetry. The problem becomes remarkably more difficult when the input object or scene is not globally symmetric but merely contains some noticeably symmetric area as its subpart. This is often referred to as local symmetry and it is quite common in real 3D scans where the symmetric object is often surrounded by other geometry or sits on a larger flat surface that represents the floor/ground (see Fig. 1 for specific examples). There can also be multiple symmetric objects in the scene. The methods that aim to solve this type of problem are not as plentiful and those that exist often require some form of pre-segmentation (Ecins et al., 2017), use a local shape descriptor that requires a mesh on the input (often a manifold one) (Mitra et al., 2006; Shi et al., 2016) or seem to be very slow even for scenes with a rather low point count (Lipman et al., 2010).

In this paper we propose a very simple PCA-based local shape descriptor and we present a new method for finding local reflectional symmetries in point clouds that uses this descriptor for point matching. Our work builds on the previous work of (Mitra et al., 2006), (Hruda et al., 2019) and (Hruda et al., 2022).

## 2 METHOD DESCRIPTION

Let  $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ ,  $\mathbf{x}_i \in E^3$  be a set of 3D points (point cloud) that represents the input scene and let  $\mathbf{r}(P, \mathbf{x})$  be a transformation that represents an arbitrary point  $\mathbf{x} \in E^3$  reflected over a given plane  $P$ . The goal of our method is to extract local reflectional symmetries from  $X$ , i.e. a set of planes  $\{P_1, P_2, \dots, P_g\}$  where each plane  $P_i$  represents a reflectional symmetry described by a transformation  $\mathbf{r}(P_i, \mathbf{x})$ .

We represent a general plane  $P$  by its implicit equation  $ax + by + cz + d = 0$  with  $a, b, c, d$  being the plane coefficients. We consider all planes to always have unit normals, i.e.  $\|[a, b, c]^T\| = 1$ . We further denote  $\mathbf{r}_{vec}(P, \mathbf{v})$  a function that represents a general vector  $\mathbf{v} \in E^3$  reflected over a given plane  $P$  where  $\mathbf{v}$  is a direction without position. The function  $\mathbf{r}_{vec}$  is computed in the same way as  $\mathbf{r}$  only ignoring the plane position, i.e. as if  $d = 0$ .

### 2.1 PCA-Based Shape Descriptor

Various symmetry detection methods use local shape descriptors to find potentially symmetric pairs of points throughout the input object/scene, e.g. (Mitra et al., 2006; Shi et al., 2016; Sipiran et al., 2014). In most cases, these descriptors are curvature/Laplacian-based and are generally difficult to compute on point clouds or even non-manifold meshes because robust estimation of the principal curvatures or the Laplacian mostly requires connectivity information and manifold surface.

In contrast, we use a very simple local shape descriptor based on the Principal Component Analysis (PCA) (Wold et al., 1987) that can be computed on a general set of points (i.e. requires neither connectivity nor manifoldness) and we show that it can be used for symmetric point matching as well. The descriptor is computed as follows.

Let us take an arbitrary point  $\mathbf{x} \in E^3$  (not nec-

essarily contained in  $X$ ) and select a radius  $r$ . The set of points  $N_{\mathbf{x}} = \{\mathbf{x}_i : \mathbf{x}_i \in X, \|\mathbf{x}_i - \mathbf{x}\| \leq r\}$  is defined to be the neighborhood of the point  $\mathbf{x}$ . By computing the PCA for the points in  $N_{\mathbf{x}}$  we get the three orthogonal eigenvectors  $\mathbf{v}_{\mathbf{x},1}, \mathbf{v}_{\mathbf{x},2}, \mathbf{v}_{\mathbf{x},3}$ ,  $\|\mathbf{v}_{\mathbf{x},i}\| = 1$ , that correspond to the principle directions of  $N_{\mathbf{x}}$ , and three eigenvalues  $\lambda_{\mathbf{x},1}, \lambda_{\mathbf{x},2}, \lambda_{\mathbf{x},3}$  where each  $\lambda_{\mathbf{x},i}$  corresponds to the variance of  $N_{\mathbf{x}}$  in the direction of  $\mathbf{v}_{\mathbf{x},i}$ . The eigenvectors and eigenvalues are indexed in such a way that  $|\lambda_{\mathbf{x},1}| \geq |\lambda_{\mathbf{x},2}| \geq |\lambda_{\mathbf{x},3}|$  which means that  $\mathbf{v}_{\mathbf{x},1}$  always represents the direction with the largest variance. The PCA-based shape descriptor for point  $\mathbf{x}$  can then be simply described by the eigenvectors  $\mathbf{v}_{\mathbf{x},1}, \mathbf{v}_{\mathbf{x},2}, \mathbf{v}_{\mathbf{x},3}$  and by a vector  $\boldsymbol{\lambda}_{\mathbf{x}} = \left[ \frac{|\lambda_{\mathbf{x},1}|}{\lambda_{norm}}, \frac{|\lambda_{\mathbf{x},2}|}{\lambda_{norm}}, \frac{|\lambda_{\mathbf{x},3}|}{\lambda_{norm}} \right]^T$  where  $\lambda_{norm}$  is a normalization constant set so that it equals the variance of all points lying on a line segment of length  $2r$  which is the maximum distance of any two points in any neighborhood. This way  $\lambda_{norm}$  simulates an eigenvalue for points distributed uniformly across the entire neighborhood in the direction of a corresponding eigenvector. The variance of a discrete set of  $k$  1-dimensional points  $x_1, x_2, \dots, x_k$  is  $\frac{1}{k} \sum_{i=1}^k (\mu - x_i)^2$  where  $\mu$  is the mean of the points. For a continuous case of a line segment of length  $2r$  we can take the interval  $\langle -r, r \rangle$  and compute its variance by integration. The mean of the interval is zero so we get

$$\lambda_{norm} = \frac{1}{2r} \int_{-r}^r (0-x)^2 dx = \frac{1}{r} \int_0^r x^2 dx = \frac{1}{3} r^2.$$

Given  $r$  is chosen relative to the input scene size, the normalization makes  $\boldsymbol{\lambda}_{\mathbf{x}}$  independent of the scene scale.

The vector  $\boldsymbol{\lambda}_{\mathbf{x}}$  holds information about the shape of the neighborhood that is invariant under both rotation and translation. This means that for two different points  $\mathbf{x}, \mathbf{y}$ , if  $\boldsymbol{\lambda}_{\mathbf{x}}$  and  $\boldsymbol{\lambda}_{\mathbf{y}}$  are similar there is a good chance that the points in the two neighborhoods  $N_{\mathbf{x}}, N_{\mathbf{y}}$  represent similar shapes just differently positioned and oriented which in turn implies that there could be symmetry between them. The  $\boldsymbol{\lambda}_{\mathbf{x}}$  vector values can also be easily visualized as shown in Fig. 3 where each point  $\mathbf{x}$  of the scene is assigned a color which RGB values corresponding respectively to the coordinates of  $\boldsymbol{\lambda}_{\mathbf{x}}$  computed with  $r = \frac{l_{avg}}{5}$  where  $l_{avg}$  is the average distance of the points in the scene from the scene's centroid. It can be seen that the symmetric counterparts of the couch in the middle are very similarly colored, i.e. the  $\boldsymbol{\lambda}$  vectors in them are similar.

However, the similarity of  $\boldsymbol{\lambda}_{\mathbf{x}}$  and  $\boldsymbol{\lambda}_{\mathbf{y}}$  does not imply in any way that the potential symmetry between  $N_{\mathbf{x}}$  and  $N_{\mathbf{y}}$  is reflectional. This information can be derived from the eigenvectors  $\mathbf{v}_{\mathbf{x},i}$  and  $\mathbf{v}_{\mathbf{y},i}$ ,  $i = 1, 2, 3$ . Specifically, having a general plane  $P$ , if the neighbor-

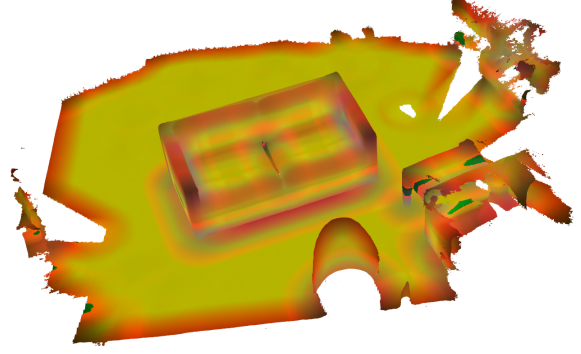


Figure 3: A scene with a reflectionally symmetric couch in the middle colored using the eigenvalues of the PCA descriptor as RGB values (data source: (Funk et al., 2017)).

hood  $N_{\mathbf{x}}$  is reflectionally symmetric with the neighborhood  $N_{\mathbf{y}}$  w.r.t. the plane  $P$  then each eigenvector  $\mathbf{v}_{\mathbf{x},i}$  will be the same as or opposite to  $\mathbf{r}_{vec}(P, \mathbf{v}_{\mathbf{y},i})$ .

These properties of the PCA descriptor can be used to find potentially symmetric pairs of points as will be described in the following text. Note that the descriptor can also be computed for points that are not directly contained in the input point cloud  $X$ , this will be shown useful later.

## 2.2 Descriptor Computation

To start matching potentially symmetric neighbourhoods using the PCA descriptor it first needs to be computed but doing so for all points in the input scene  $X$  could easily get computationally overwhelming as real 3D scanned scenes often consist of hundreds of thousands of points. Also, when computing the descriptor for a given point, the points in the neighbourhood of the selected radius first need to be extracted and doing so while avoiding brute-force computation is more than desirable.

### 2.2.1 Simplification

Instead of computing the descriptor for all points of the input scene  $X$  we create a simplified version of the scene denoted  $X' = \{\mathbf{x}'_1, \mathbf{x}'_2, \dots, \mathbf{x}'_m\}$ ,  $\mathbf{x}'_i \in E^3$ . We use the same grid-based point cloud simplification algorithm as used in (Hruda et al., 2022) which stems from putting the points of the point cloud into cells of a uniform grid and averaging points in every cell. Therefore, each cell provides a single point of the new point cloud which is generally not contained in the original point cloud. The process is repeated several times with decreasing cell size until a target point count is reached approximately. We use this algorithm with target point count 10,000 which is then the rough number of points of  $X'$ .

Since the PCA descriptor can be computed for points outside of  $X$  but still represent its local features we compute it at the  $m$  (roughly 10,000) points of  $X'$  but for its computation at a given point  $\mathbf{x}' \in X'$  we use the neighbourhood extracted from  $X$  in the corresponding region. This not only reduces computational cost but also ensures that the points used for symmetric matching (see Section 2.4) are more uniformly distributed over the scene.

### 2.2.2 Neighbourhood Extraction

To efficiently extract points of  $X$  lying in a given neighbourhood of radius  $r$  we again use a grid-based approach from (Hruda et al., 2022). At the beginning the points of  $X$  are stored in a uniform grid with cell size  $r \times r \times r$ . Then, when a neighbourhood of a given point  $\mathbf{x}'$  needs to be extracted, we search for points whose distance from  $\mathbf{x}'$  is  $\leq r$  only among points that are in the same cell as  $\mathbf{x}'$  and cells adjacent to this cell. This is, of course, significantly faster than brute-force iterating over all points of  $X$ , at least when  $r$  is reasonably small. We use  $r = \frac{l_{avg}}{5}$  as the default value where the number 5 was chosen experimentally.

Other data structures, such as K-d tree, BSP tree or octree, could be used as well. However, we believe that the grid-based approach provides similar boost in performance and is simpler to implement than the tree-based structures.

## 2.3 Finding Planar Points

The PCA descriptor at points that lie in roughly planar neighbourhoods (floor, walls, etc.) does not really provide useful information in terms of symmetry. First, because the first two eigenvectors have basically random direction at planar points. Second, because planar areas are usually quite plentiful in real world scenes and could easily cause too many false positive symmetric matches - this can be easily seen in Fig. 3 where the planar points are colored yellow. This is why we first detect the points of  $X'$  that have approximately planar neighbourhoods so that we could exclude them from the later steps.

For a given point  $\mathbf{x}' \in X'$  we decide whether it is planar in the following way. We construct a plane that passes through  $\mathbf{x}'$  and whose normal vector is given by  $\mathbf{v}_{\mathbf{x}',1} \times \mathbf{v}_{\mathbf{x}',2}$ . We then compute the average distance of points in  $N_{\mathbf{x}'}$  from this plane and denote it  $\Delta_{\mathbf{x}'}^{\perp}$  - this value will be the smaller the points in  $N_{\mathbf{x}'}$  fit the plane. We also project each point in  $N_{\mathbf{x}'}$  orthogonally on the same plane and compute the average distance of these projected points from  $\mathbf{x}'$ . We denote it  $\Delta_{\mathbf{x}'}^{\parallel}$  and it serves as a normalization value. Now the point

$\mathbf{x}'$  is marked as planar if  $\frac{\Delta_{\mathbf{x}'}^{\perp}}{\Delta_{\mathbf{x}'}^{\parallel}} < \epsilon_{\Delta}$  where  $\epsilon_{\Delta} = 0.1$  and its value was experimentally chosen.

## 2.4 Symmetric Point Matching

After the PCA descriptor has been computed in all points of the simplified point cloud  $X'$  and planar points were found we start searching for potentially symmetric neighbourhoods. First, we find pairs of non-planar points with similar  $\boldsymbol{\lambda}$  vectors and then we verify their potential reflectional symmetry using the PCA eigenvectors.

### 2.4.1 $\boldsymbol{\lambda}$ Similarity Matching

For every non-planar point  $\mathbf{x}'_i \in X'$ ,  $i = 1, \dots, m$  we find a non-planar point  $\mathbf{x}'_j \in X'$ ,  $j = 1, \dots, m$ ,  $i \neq j$  such that  $\|\boldsymbol{\lambda}_{\mathbf{x}'_i} - \boldsymbol{\lambda}_{\mathbf{x}'_j}\| \leq \epsilon_{\lambda}$ , where  $\epsilon_{\lambda} = 0.05$ , and we add the pair  $(\mathbf{x}'_i, \mathbf{x}'_j)$  to a set  $Q$  of potentially symmetric point pairs. If there are multiple points  $\mathbf{x}'_j$  satisfying these conditions we only use the one with the smallest distance  $\|\boldsymbol{\lambda}_{\mathbf{x}'_i} - \boldsymbol{\lambda}_{\mathbf{x}'_j}\|$ . To avoid brute-force computation we again use the grid-based approach described in Section 2.2.2, only this time we store the  $\boldsymbol{\lambda}$  vectors in the grid with cell size  $\epsilon_{\lambda} \times \epsilon_{\lambda} \times \epsilon_{\lambda}$ , and use it to efficiently find other vectors up to the distance  $\epsilon_{\lambda}$  from a given  $\boldsymbol{\lambda}$  vector. The value of  $\epsilon_{\lambda}$  was chosen experimentally.

In the end the set  $Q$  contains pairs  $(\mathbf{x}'_i, \mathbf{x}'_j)$  where each such pair of points could potentially provide evidence for symmetry.

### 2.4.2 Symmetry Verification

For each of the pairs acquired in the previous step we need to check whether it truly presents evidence of potential reflectional symmetry. This is done using the eigenvectors of the PCA descriptor as suggested in Section 2.1.

Since the first eigenvector bears the most relevant information about the shape of the local neighborhood (it corresponds to the largest eigenvalue), for simplicity, we do not use the second and third eigenvectors for the symmetry verification, only the first one. Therefore, we denote  $\mathbf{v}_{\mathbf{x}'} = \mathbf{v}_{\mathbf{x}',1}$  the relevant eigenvector of the PCA descriptor computed at point  $\mathbf{x}'$ . However, both the second and third eigenvectors could be included in the verification step but our experiments do not suggest that it would cause a significant improvement.

For each one of the point pairs  $(\mathbf{x}'_i, \mathbf{x}'_j) \in Q$  we construct a plane  $P$  such that  $\mathbf{r}(P, \mathbf{x}'_i) = \mathbf{x}'_j$  and perform a



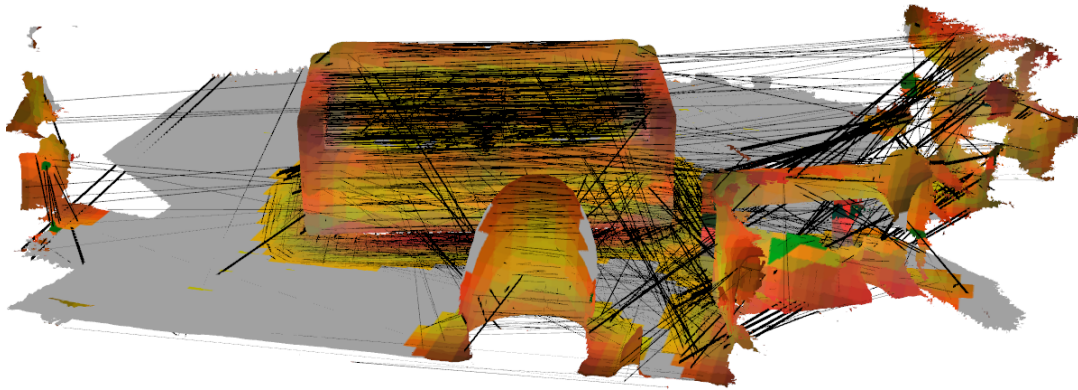


Figure 4: A scene with the PCA-based descriptor symmetric matching - the symmetrically matched pairs of points of  $X'$  are connected by the black lines, the coloring corresponds to the descriptor eigenvalues computed on  $X'$ , the grey color indicates planar areas.

simple test. The pair of points is accepted as potentially reflectionally symmetric if

$$\min(\|\mathbf{r}_{vec}(P, \mathbf{v}_{x'_i}) - \mathbf{v}_{x'_j}\|, \|\mathbf{r}_{vec}(P, \mathbf{v}_{x'_i}) + \mathbf{v}_{x'_j}\|) \leq \epsilon_v$$

where  $\epsilon_v = 0.1$  and its value was chosen based on vast experiments. When the pair of points passes the test the plane  $P$  is added to a set  $C$ . As a result we acquire a set  $C = \{P_1, P_2, \dots, P_h\}$  of candidate symmetry planes.

Fig. 4 shows the symmetric matching for the scene with the couch in the middle. The black lines connect pairs of points of  $X'$  that were matched together and passed the verification step, i.e. their symmetry plane is in  $C$ . The coloring now reflects eigenvalues of the PCA descriptor computed for  $X'$  - the color of a given point of  $X'$  was transferred to all points of the input scene  $X$  that were in the same cell of the simplification grid in the last iteration of the simplification process (see Section 2.2.1). The grey colored areas correspond to points that were marked as planar (see Section 2.3). Among others, the figure reveals numerous symmetric matches between the reflectionally symmetric counterparts of the couch indicating that there are many planes in  $C$  that capture its symmetry.

## 2.5 Mode Seeking

We follow the theory, used e.g. in (Mitra et al., 2006; Shi et al., 2016; Hruda et al., 2019), that the significant symmetries should now form modes in the space of candidate transformations - planes in our case. Finding the modes is often solved by clustering (Mitra et al., 2006; Shi et al., 2016) but we believe that clustering algorithms might become unnecessarily complex and computationally expensive. Also, a cluster of points might not necessarily correspond to

a local mode since a mode is represented by a single point but a cluster is a set of points. Therefore, with clustering, there would be some further actions required regarding selecting a proper point from each cluster to represent the mode.

Instead, we use the density peak location algorithm proposed in (Hruda et al., 2019) modified to serve our purpose. In general, the algorithm can be described as follows. Let  $\rho(P_i)$  be the density at a given candidate  $P_i \in C$ . The density at a given point in the space of planes corresponds to how many similar candidates there are to this point, i.e. the points of high density should correspond to the modes in the candidate space. The most significant mode, and therefore the strongest symmetry candidate, would be simply identified as  $\arg \max_{P_i \in C} \rho(P_i)$ .

### 2.5.1 Density Function

The density function is defined as

$$\rho(P_i) = \sum_{P_j \in C} K(D(P_i, P_j))$$

where  $D(P_i, P_j)$  is a distance measure between  $P_i$  and  $P_j$  and  $K(l)$  is a kernel which in (Hruda et al., 2019) was originally set to the standard Gaussian function but we instead use the modified Wendland's function from (Hruda et al., 2022). We denote it  $\varphi(x)$ , it has a very similar shape to the Gaussian function but is locally supported and smoothly goes to 0 at  $x = 2.6$ . We set  $K(l) = \varphi(\alpha l)$  where  $\alpha$  is a spread parameter and  $l$  represents distance. We experimentally set  $\alpha$  as  $\alpha = 25$ . The local support property of the kernel draws a direct boundary between candidates that contributed to the density of  $P_i$  and those that did not which will be used later.

Denoting  $\mathbf{p}_i = [a_i, b_i, c_i, \frac{d_i}{l_{avg}}]^T$  a representative coefficient vector of candidate plane  $P_i$ , the distance

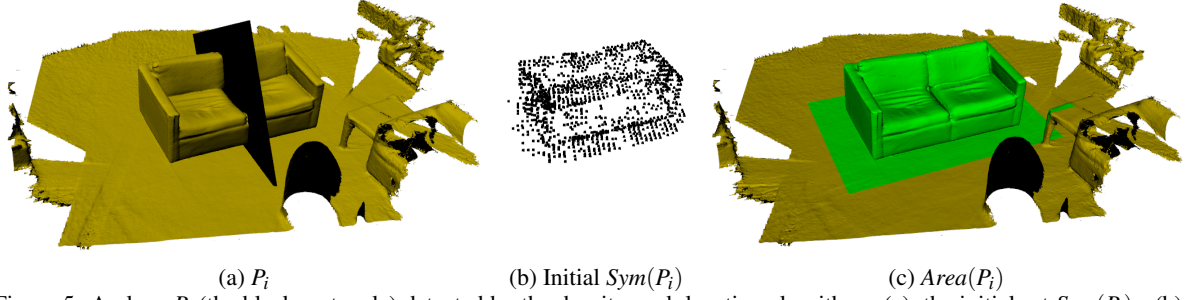


Figure 5: A plane  $P_i$  (the black rectangle) detected by the density peak location algorithm - (a), the initial set  $Sym(P_i)$  - (b), the final symmetric subpart of  $X$ ,  $Area(P_i)$  (marked by the green color) - (c).

measure  $D$  is defined as

$$D(P_i, P_j) = \min(\|\mathbf{p}_i - \mathbf{p}_j\|, \|\mathbf{p}_i + \mathbf{p}_j\|) \quad (1)$$

which, according to (Hruda et al., 2020), is usable for this purpose but only when the input object is centered at the origin which is why, at the beginning, we translate the input scene  $X$  such that its centroid lies at the origin and at the end we translate the detected symmetry planes back. The division of the  $d$  coefficient by  $l_{avg}$  ensures its normalization by the scene size which, according to (Hruda et al., 2020), is also required for the distance measure to be meaningful.

### 2.5.2 Density Peak Location

In (Hruda et al., 2019) the density peak location algorithm was only used to find the most significant mode in the transformation space and, due to usage of a non-Euclidean distance metric, the Vantage Point Tree (Yianilos, 1993) data structure was used for efficient computation of the density function. However, we can have multiple symmetries in a single scene which means we should be able to find multiple local modes, not just the largest one. Since our distance function  $D$  is Euclidean, when computing the density for a candidate  $P_i$ , we can use any standard data structure to quickly find only the candidates whose distance from  $P_i$  is not larger than the radius of the support region of the kernel  $K$ . We again use a grid as the auxiliary data structure, only this time in 4 dimensions, as previously used in (Hruda et al., 2022) for finding similar planes.

We first find the candidate  $P_i \in C$  that maximizes the density function  $\rho(P_i)$  and add it to a set  $S$  of preliminary symmetries. Then, we remove from  $C$  all candidates that had non-zero contribution to the density in  $P_i$ , i.e. were inside the support region of the kernel  $K$  at  $P_i$ . We can now repeat this process to find the second most significant mode etc. We can either do this a certain predefined number of times or, as we do, keep repeating the process as long as the density at each newly located mode is above a given threshold. We keep finding new modes and adding them to

$S$  as long as their density value is at least half of the density that the first located mode had. At the end the set  $S \subset C$  contains the preliminary reflectional symmetries.

## 2.6 Finalizing Symmetries

The planes in the set  $S$  extracted in the previous step might not be accurate and not all of them actually represent plausible symmetries. For each plane in  $S$  we now need to extract the subpart of  $X$  which is actually symmetric w.r.t. the plane and refine the symmetry for the corresponding subpart. At last, we need to discard symmetries that are too weak to be meaningful.

### 2.6.1 Extracting Symmetric Areas

For each of the planes in  $S$  we find the symmetric subpart of  $X$  in the following way. For a given plane  $P_i \in S$  we take all the planes that contributed by a non-zero value to its density in the density peak location step in which it was added to  $S$  (see Section 2.5.2). For each of these contributing planes we take the pair of points that initially created the plane (see Sections 2.4.1 and 2.4.2) and add them to a set denoted  $Sym(P_i)$  - a set of points of  $X'$  that contributed to locating  $P_i$  as a mode in the candidate space. The points in  $Sym(P_i)$  provide a rough information about the position and size of the subpart of  $X$  that is symmetric w.r.t.  $P_i$ . Fig. 5a shows an example of a plane  $P_i$  (the black rectangle) found by the density peak location process and Fig. 5b shows the points in the corresponding  $Sym(P_i)$  set.

We compute the centroid of  $Sym(P_i)$  and use PCA to compute its principal axes - PCA eigenvectors  $\mathbf{v}^x$ ,  $\mathbf{v}^y$ ,  $\mathbf{v}^z$  - and the variance in the direction of each of these axes - eigenvalues  $\lambda^x$ ,  $\lambda^y$ ,  $\lambda^z$ . We express each point  $\mathbf{x}_j \in Sym(P_i)$  as a point in the local coordinate system with the origin at the centroid of  $Sym(P_i)$  and basis directions of  $\mathbf{v}^x$ ,  $\mathbf{v}^y$ ,  $\mathbf{v}^z$ . This local expression of point  $\mathbf{x}_j$  is denoted  $\hat{\mathbf{x}}_j = [\hat{x}_j, \hat{y}_j, \hat{z}_j]^T$ . We consider a point  $\hat{\mathbf{x}}_j$  an outlier if the absolute value of one of

its coordinates is larger than double of the standard deviation of the points of  $Sym(P_i)$  (its square is larger than quadruple of the variance) in the corresponding direction, i.e. we only keep points  $\mathbf{x}_j$  for which the expression

$$\hat{x}_j^2 \leq 4\lambda^x \text{ and } \hat{y}_j^2 \leq 4\lambda^y \text{ and } \hat{z}_j^2 \leq 4\lambda^z \quad (2)$$

is true and remove all other points from  $Sym(P_i)$ . We repeat this process one more time (compute centroid and PCA of the new  $Sym(P_i)$  and remove outliers) and then we create a point set  $Area(P_i) \subset X$  that contains all points from the input point cloud  $X$  that satisfy the expression in Eq. (2) only this time with  $\mathbf{x}_j \in X$  and the centroid, principal axes and variances being computed for the final  $Sym(P_i)$  point set. The set  $Area(P_i)$  is now considered the subpart of  $X$  that is symmetric w.r.t. the preliminary symmetry plane  $P_i \in S$ . Fig. 5c depicts the extracted  $Area(P_i)$  (colored green) for the  $P_i$  plane from Fig. 5a.

### 2.6.2 Symmetry Refinement

We now refine the preliminary symmetries with respect to their corresponding subparts of the input point cloud  $X$ . For each plane  $P_i \in S$  we use the optimization step for global reflectional symmetry from the method of (Hruda et al., 2022) where  $P_i$  is the initial plane and  $Area(P_i)$  is the expected globally symmetric object. This method uses a differentiable symmetry measure that represents the symmetry of the input object w.r.t. the given plane and a gradient-based optimization to find its local maximum. We also include the simplification process described in Section 3.3 in (Hruda et al., 2022) where the input object is simplified to roughly 1000 points before the optimization is started.

We replace each plane  $P_i \in S$  by a plane that was converged to by the optimization started from  $P_i$  on  $Area(P_i)$ . As a result the accuracy of the symmetry planes in  $S$  should increase.

### 2.6.3 Excluding Bad Symmetries

The density peak location in the space of candidate planes created by the PCA-based descriptor point pairing can produce false positives and even the refinement from the previous step does not guarantee that all the planes in  $S$  are meaningful.

We use the symmetry measure from the refinement step to determine the plausibility of a given plane. The symmetry measure is computed by transforming each point of the object by the given transform (in our case reflection over a plane) and computing similarity (in terms of distance) of the transformed point to all points of the object. All the similarities

are summed together to provide the measure value (for more details we refer to (Hruda et al., 2022)). The problem is that this value is not normalized in any way and does not only depend on the strength of the symmetry but also on various properties of the input object. Therefore, we use a modified version of the symmetry measure where we divide its value by a symmetry measure value that would be achieved by perfect symmetry. We simulate the perfect symmetry measure by computing the symmetry measure of the identity transform. We call this modified version Relative Symmetry Measure (RSM).

For each plane  $P_i \in S$  we now compute the RSM of the reflection over  $P_i$  transform on  $Area(P_i)$ . All planes with RMS greater than 0.4 are accepted as the resulting symmetry planes of the input scene  $X$ .

## 3 RESULTS

Our method was implemented in C# and the results were acquired on a computer with CPU Intel® Core i7-10700F (frequency: 2.9GHz, turbo boost: 4.8GHz, 8 cores, L1 cache: 512kB, L2 cache: 2MB, L3 cache: 16MB) and 32GB memory (frequency: 3.2GHz) running a Windows 10 operating system.

We tested the method on several real 3D scenes from the dataset (Funk et al., 2017). The results can be seen in Fig. 6, the black rectangles indicate the detected planes. We note that although the objects in the figure are represented by triangle meshes, we only used the vertices as the input point cloud  $X$  for our method. Triangle meshes are much easier to visualize and most likely the objects were originally scanned as point clouds and meshing was done in post processing which means that the vertices we use actually represent the original raw point clouds. It seems that the strongest symmetries are detected correctly in all of these 12 scenes but it should be noted that some of the scenes also contain some weaker symmetries that were not identified by our method, this is especially noticeable in Scene 12. On the other hand, the method sometimes finds symmetries that are technically correct but would not be expected by a human observer. An example of this case is Scene 7 where two symmetries were found on the long edge of the couch seat on the right side. Such symmetries are not meaningful for a person but it is understandable that the method detects them since there are many points symmetric w.r.t. both these planes. In Scene 9 our method also identified a plausible symmetry plane of the partial chair/couch in the top right corner. Sometimes, mostly when there are multiple objects in different places that are symmetric w.r.t. the same plane,



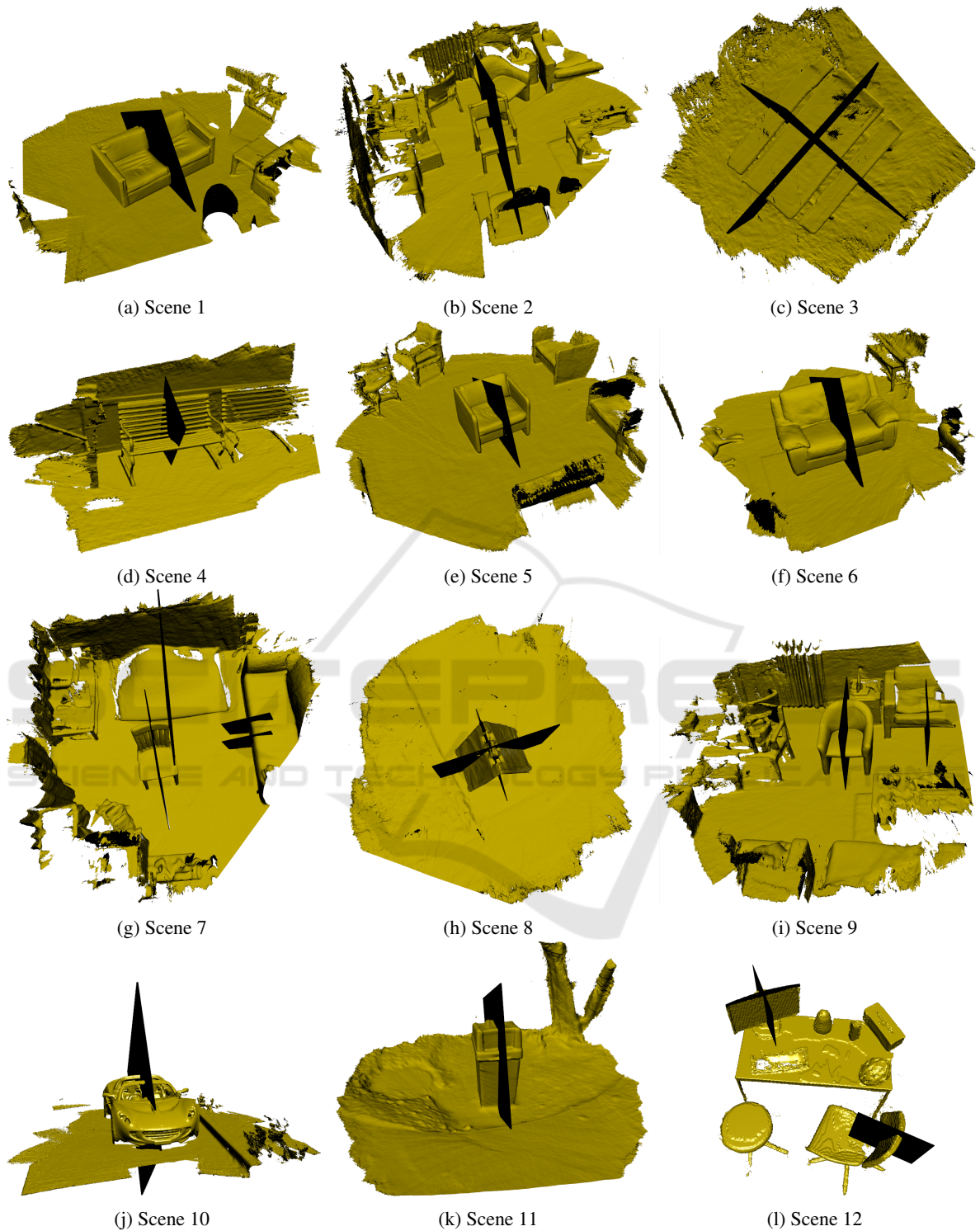


Figure 6: The results of our method on several 3D scanned scenes from the (Funk et al., 2017) dataset, the detected symmetry planes are marked by the black rectangles.

the method covers a larger area with a single symmetry. This can be seen in Scene 2 where we have a single symmetry plane for the chair in the middle

and the partial chair in front of it because both these objects were included in a single symmetric area.

For demonstration of the quality of the detected



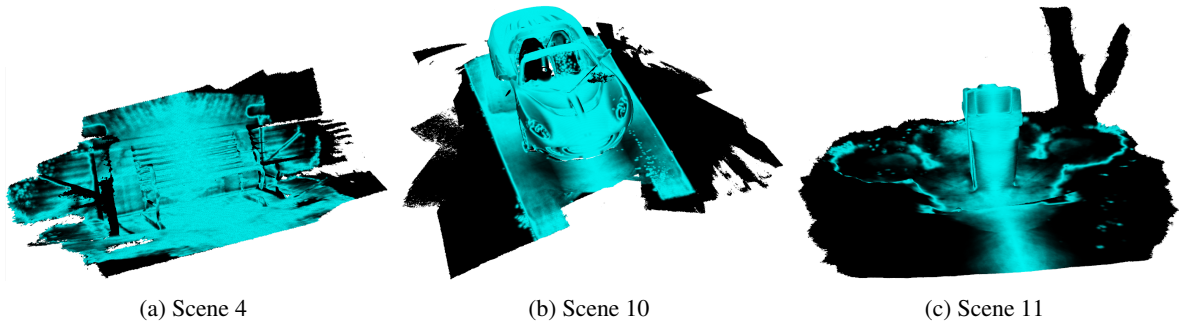


Figure 7: Symmetry coloring from (Hruda et al., 2022) for the one detected plane in three different scenes - the lighter the color the stronger the symmetry of the given point.

symmetries we used the symmetry coloring proposed in (Hruda et al., 2022) to show how strong a given symmetry is for different points of the scene. We used this for scenes 4, 10 and 11 where we always took the one detected plane (see Fig. 6) and we colored all the points in the input scene in correspondence to the symmetry of this plane. The coloring is depicted in Fig. 7. The lighter the color the stronger the symmetry of the given point. The symmetry mostly covers the object for which it was detected but, of course, partially also includes some other regions of the scene such as the ground.

Tab. 1 shows the number of points and computation time of our method for each scene. Although the point counts of the scenes are quite large (hundreds of thousands) the computation time ranges around 20s, and sometimes gets under 10s, which can be considered rather fast. The exception is Scene 10 which has almost 2 million points. We use a single-threaded implementation of the method but its most time consuming stages (descriptor computation and point pairing) could be easily parallelized which would make the overall method potentially several times faster.

Our method should work on point clouds with even larger point counts (tens of millions / billions) but the computational cost would probably become overwhelming.

### 3.1 Noisy Data

In order to test robustness to noise of our method and the PCA-based descriptor we selected two scenes and added a vector  $[rand_x, rand_y, rand_z]^T \cdot l_{avg} \cdot mag$  to each of their points where  $rand_x, rand_y, rand_z$  are uniform random values from  $-1$  to  $1$  and  $mag$  is the noise magnitude. Fig. 9 shows the detected planes for the two scenes with added noise with  $mag = 0.025$  and  $mag = 0.05$ . We colored the objects in the same way as in Fig. 4 to show what effect the noise has on the  $\lambda$  vectors of the PCA descriptor. The detected symmetry planes are very similar to those detected on the

Table 1: The computation time of our method in seconds for each one of the scenes from Fig. 6.

Scene	Point count	Time [s]
1	387540	12.3
2	598324	18
3	439089	12.5
4	287875	8.6
5	515496	18.7
6	380592	10.5
7	559905	24.4
8	321052	10.6
9	566153	16.1
10	1891185	133.7
11	288759	9.3
12	233600	7.7

noiseless versions of the scenes and the  $\lambda$  vectors of the descriptor seem to still well represent similarities of the local neighborhoods that could be symmetric.

The planar area detection step seems to start failing under the influence of the noise but it apparently does not impact the symmetry detection results very much. This might partially be because the noise also makes the PCA eigenvectors a little more random in the planar regions which probably also lowers the potential for false positive matches.

Note that, although the first step of our method is simplification which partially suppresses the noise, the simplified point set is only used for point pairing but the PCA-based descriptor is computed for the points of the original point set and is therefore fully affected by the noise.

Fig. 8 shows how the noise changes the detected plane for different scenes. For this experiment we only used scenes where the method finds exactly one symmetry plane. For each scene we kept increasing the noise magnitude and for each magnitude level we used the density peak location algorithm to find the single most pronounced plane in the candidate space regardless of its RSM value. We then used the plane distance function from Eq. (1) to measure

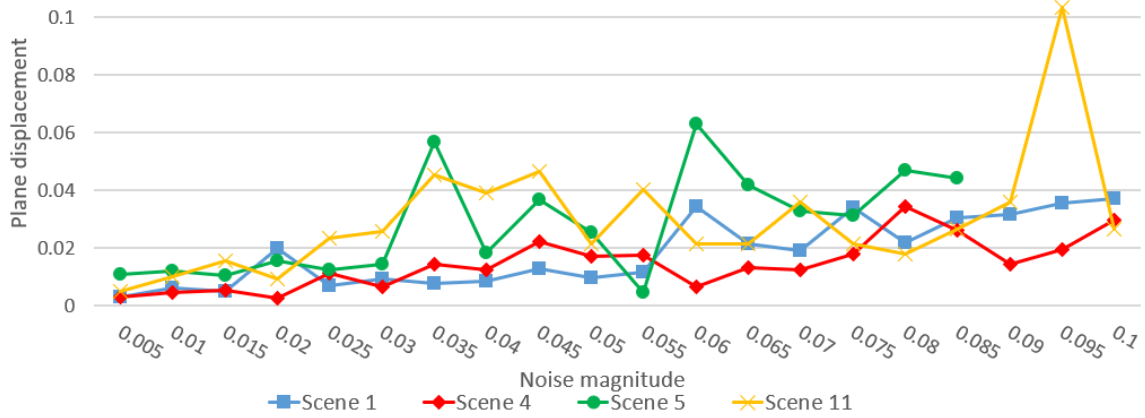


Figure 8: Displacement of the symmetry plane detected on different scenes with different magnitude of added noise from the plane detected on the noise free version of the scene expressed in terms of the distance function from Eq. (1).

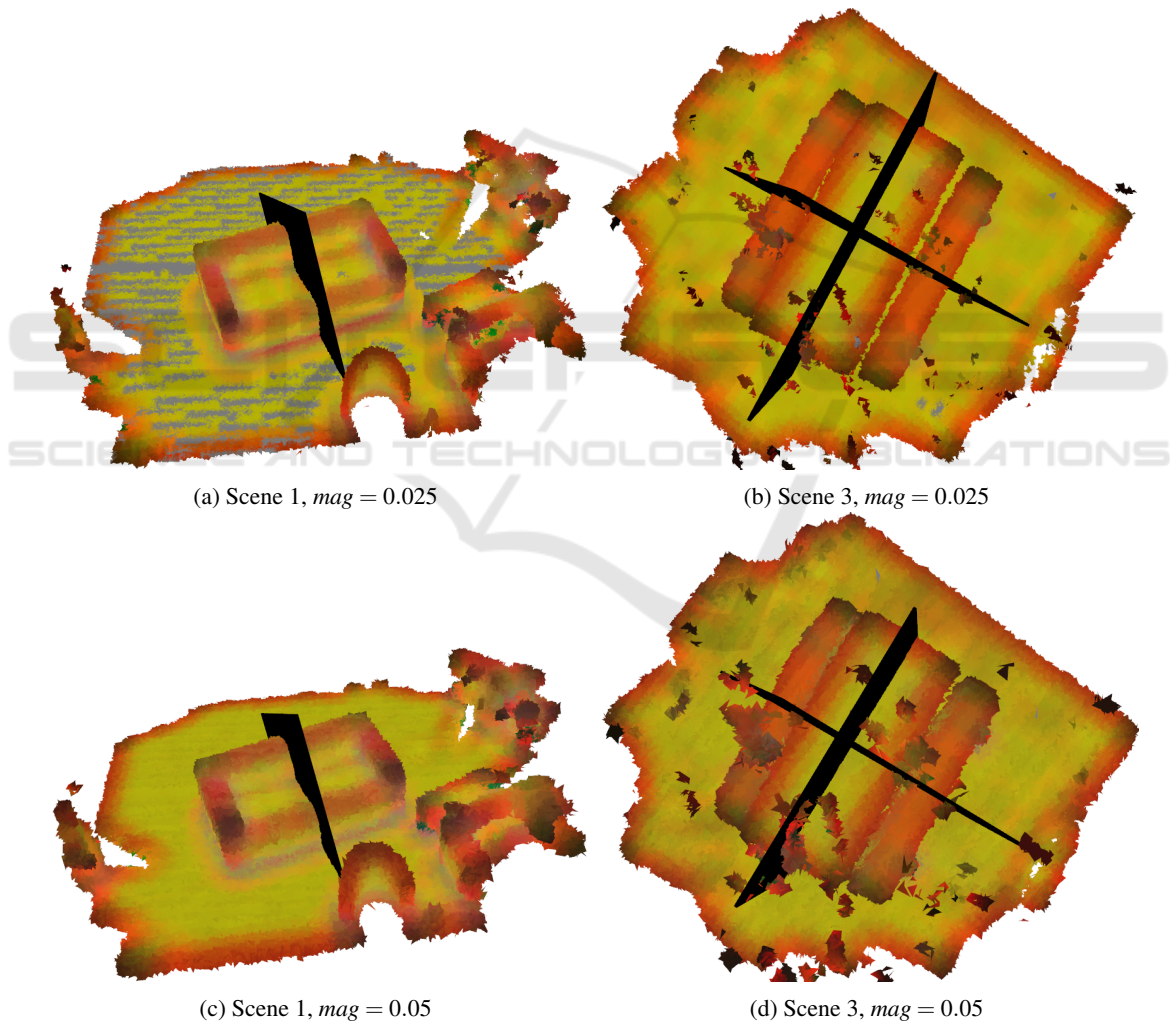


Figure 9: Results of our method for two different scenes with added noise with  $mag = 0.025$  and  $0.05$ , the colors indicate eigenvalues of the PCA descriptor, grey corresponds to detected planar areas.

the displacement of the detected plane from the plane that was found on the original noise free scene. For

Scenes 1 and 4 there seems to be roughly linear dependency between the noise magnitude and the plane

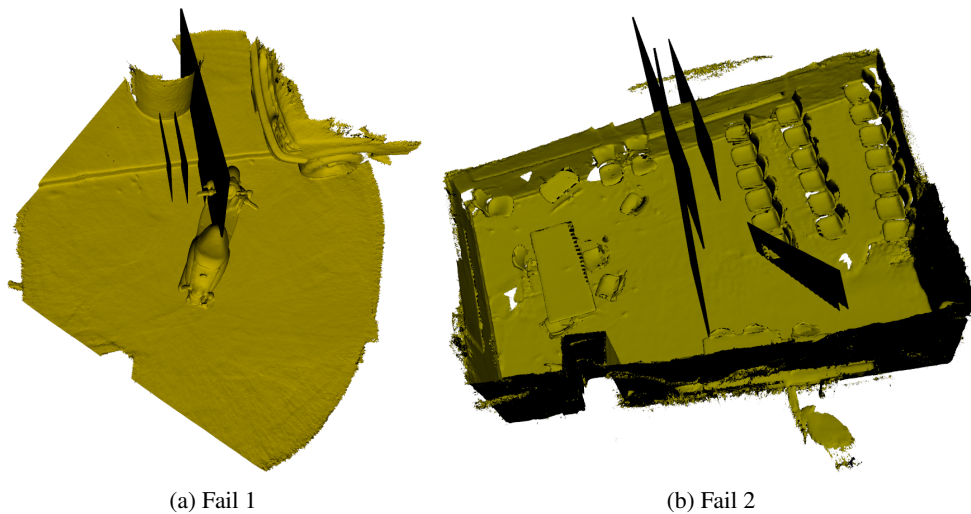


Figure 10: Examples of two input scenes where the method fails to detect plausible symmetry planes.

displacement which increases rather slowly showing that the noise does not influence the function of the PCA descriptor and the density peak location step significantly. In Scenes 5 and 11 the displacement appears to be much more random and oscillates a lot. For Scene 5 we do not include the displacement for  $mag > 0.085$  because for larger magnitudes the method stops detecting the corresponding plane completely and the displacement increases by an order of magnitude which would make the graph unreadable.

### 3.2 Globally Symmetric Objects

Since a method designed to detect local symmetries should also somewhat work for detecting global symmetries, in Fig. 11 we show the results of our method for two selected globally symmetric objects. The method correctly identified the symmetry of the Armadillo despite its missing parts. For the Buddha object the method detected its overall global symmetry and also several local symmetries of the pedestal.

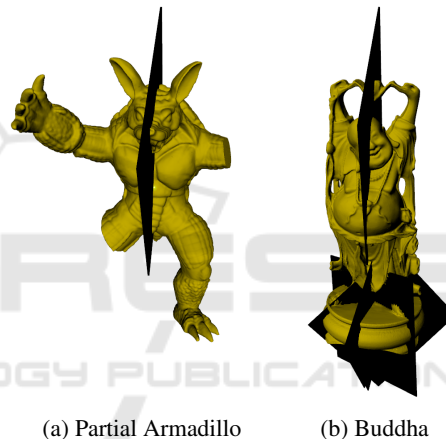


Figure 11: Results of our method for two selected globally symmetric objects.

Our method also has problems with detecting symmetries on objects that are very small in comparison to the overall size of the input scene. An example is shown in Fig. 10b where the method finds several nonsensical symmetries instead of finding symmetries of the chairs because the chairs are too small. This issue might be fixable by different parameter settings but we were unable to find a setting that would work for this particular scene.

## 4 LIMITATIONS

Our method also has some weaknesses and limitations some of which were already indicated in the previous section. The method can be easily confused by situations where long edges appear in the scene. This was already shown in Fig. 6, Scene 7 but a more explicit example of this problem can be seen in Fig. 10a where, instead of finding the dominant symmetry of the scooter in the middle, the method finds multiple visually insignificant symmetries along the long edge in the upper part of the scene.

## 5 CONCLUSION

We proposed a very simple PCA-based local shape descriptor and we have shown how it can be used for reflectional symmetric point matching in 3D point clouds. We presented a new method for finding local reflectional symmetries that is based on this idea

in combination with the density peak location algorithm and we have shown its results on several real 3D scanned scenes. We demonstrated the robustness to noise of both the PCA descriptor and the overall method as well as its computational efficiency. The limitations of the method described in the previous section are to be addressed in the future.

## ACKNOWLEDGEMENTS

This work was supported by the Czech Science Foundation, project GACR 21-08009K Generalized Symmetries and Equivalences of Geometric Data. Lukáš Hruďa was also funded by Ministry of Education, Youth and Sports of the Czech Republic – the student research project SGS-2022-015 New Methods for Medical, Spatial and Communication Data.

## REFERENCES

- Combès, B., Hennessy, R., Waddington, J., Roberts, N., and Prima, S. (2008). Automatic symmetry plane estimation of bilateral objects in point clouds. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.
- Ecins, A., Fermuller, C., and Aloimonos, Y. (2017). Detecting reflectional symmetries in 3d data through symmetrical fitting. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1779–1783.
- Funk, C., Lee, S., Oswald, M. R., Tsogkas, S., Shen, W., Cohen, A., Dickinson, S., and Liu, Y. (2017). 2017 iccv challenge: Detecting symmetry in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1692–1701.
- Hruďa, L., Dvořák, J., and Váša, L. (2019). On evaluating consensus in ransac surface registration. In *Computer Graphics Forum*, volume 38, pages 175–186. Wiley Online Library.
- Hruďa, L., Kolingerová, I., and Lávička, M. (2020). Plane space representation in context of mode-based symmetry plane detection. In *International Conference on Computational Science*, pages 509–523. Springer.
- Hruďa, L., Kolingerová, I., and Váša, L. (2022). Robust, fast and flexible symmetry plane detection based on differentiable symmetry measure. *The Visual Computer*, 38(2):555–571.
- Li, B., Johan, H., Ye, Y., and Lu, Y. (2016). Efficient 3d reflection symmetry detection: A view-based approach. *Graphical Models*, 83:2–14.
- Lipman, Y., Chen, X., Daubechies, I., and Funkhouser, T. (2010). Symmetry factored embedding and distance. In *ACM SIGGRAPH 2010 papers*, pages 1–12.
- Martinet, A., Soler, C., Holzschuch, N., and Sillion, F. X. (2006). Accurate detection of symmetries in 3d shapes. *ACM Transactions on Graphics (TOG)*, 25(2):439–464.
- Mitra, N. J., Guibas, L. J., and Pauly, M. (2006). Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (TOG)*, 25(3):560–568.
- Nagar, R. and Raman, S. (2020). 3dsymm: robust and accurate 3d reflection symmetry detection. *Pattern Recognition*, 107:107483.
- Shi, Z., Alliez, P., Desbrun, M., Bao, H., and Huang, J. (2016). Symmetry and orbit detection via lie-algebra voting. In *Computer Graphics Forum*, volume 35, pages 217–227. Wiley Online Library.
- Sipiran, I., Gregor, R., and Schreck, T. (2014). Approximate symmetry detection in partial 3d meshes. In *Computer Graphics Forum*, volume 33, pages 131–140. Wiley Online Library.
- Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.
- Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, volume 66, page 311. SIAM.
- Žalik, B., Strnad, D., Kohek, Š., Kolingerová, I., Nerat, A., Lukač, N., and Podgorelec, D. (2022). A hierarchical universal algorithm for geometric objects’ reflection symmetry detection. *Symmetry*, 14(5):1060.