Identifying Symmetry in Scalar Fields

A Project Report

Submitted in partial fulfilment of the

requirements for the Degree of

Master of Engineering

in

Computer Science and Engineering

by

Talha Bin Masood

Computer Science and Automation

Indian Institute of Science

BANGALORE – 560 012

June 2012

TO

My Parents

Acknowledgements

I would like to thank Dr. Vijay Natarajan for his excellent guidance and valuable suggestions during this work. I am also grateful to Dilip for having many helpful discussions with me concerning this project and giving me new ideas when I got stuck. I am thankful to all my lab mates, specially Harish and Sonali for providing some of the code.

Lastly, I would like to convey my gratitude to my parents and friends for their constant encouragement. Gaurav and Santanu deserve special mention for their moral and intellectual support.

Abstract

Study of symmetric or repeating patterns in scalar fields is important in scientific data analysis because it gives insights into the properties of the underlying phenomenon. Identifying symmetry in scalar fields has largely remained unexplored till now. A few approaches for finding global symmetry in scalar fields and finding symmetry in scalar field topology have been proposed recently. Existing methods for identifying symmetry in scalar field topology completely ignores the geometry, and hence may report non-intuitive symmetric regions. In this report we provide a more precise definition for symmetry in scalar fields that incorporates the geometry and propose an approach for identifying partial and approximate symmetry in 2D and 3D scalar fields. The proposed approach overcomes the shortcomings of identifying symmetry based solely on topology.

Contents

Chapter 1

Introduction

Symmetry is occurrence of repeating patterns. Detecting and characterizing symmetry is important in wide ranging fields like engineering, physics, biology, etc. In engineering and manufacturing, symmetric patterns are used in design because they make the whole structure more stable and efficient. In biology, almost all multicellular organisms have symmetric body structure. Symmetry is observed even at molecular and atomic level, where it has helped us in understanding the structure of molecules and atoms. Symmetry is so important in some cases that its absence points to some abnormality in the structure. Symmetry is also of importance in the fields of arts and architecture. Symmetric patterns in art and architecture have long been used to enhance aesthetic beauty.

Within computer graphics, symmetry in geometric shapes is a well studied topic. Symmetry in the context of shapes refers to properties that remain invariant under geometric transformations and is detected and characterized by studying the geometry of shapes. Symmetry finds application in the area of shape processing for tasks like object recognition and reconstruction, shape matching, segmentation, and shape editing.

We encounter scalar fields defined on a domain of interest, in many areas of engineering and scientific research, e.g. scalar field may represent scientific measurements or results of simulations. Studying the properties of scalar fields is very important for scientific data analysis. Different properties of scalar field like level sets, their topology, etc. have been well studied to gain information about the underlying phenomenon. We

believe that study of symmetry in scalar fields will provide more information about the scalar field and thus help in better understanding the phenomenon or scientific experiment represented by the scalar field. By symmetry in scalar fields, we mean invariance in the distribution of scalar field within different parts of the domain. Identification of symmetry can also help in reducing the complexity of studying a phenomenon, by focusing attention on a single region from a group of symmetric regions.

In this report, we define the notion of symmetry in scalar fields and propose an approach for efficiently finding symmetric regions in the scalar field.

Chapter 2

Previous Work

Substantial work has been done concerning symmetry in geometric shapes. However, there has been very few attempts in the direction of finding symmetry in scalar fields. In fact, there is no known formal definition of symmetry in scalar fields. Here, we provide an overview of techniques for detecting symmetry in geometric models and in scalar fields. We will focus on a voting based technique for detecting symmetry in geometric shapes [14] and a contour tree bsed technique for symmetry detection in scalar fields [19].

2.1 Symmetry in Geometric Shapes

Detection of symmetry is a hard problem because there is no prior information about the symmetries to look for. Early work in this field addressed the problem of finding perfect symmetries [3,13]. However, it is rare to have perfect symmetry. Real data is usually noisy, so robust techniques are required to detect partial and approximate symmetries. Several solutions have been proposed for this problem [12,14,16]. Mitra et. al. proposed a two step solution for solving the problem of identifying partial and approximate symmetry [14]. In the first step, evidence for symmetries is accumulated by considering local signatures of shape and voting for a transformation in transformation space. In the second step, clusters obtained after clustering in transformation space are verified and symmetric patches are extracted.

Symmetry in geometric shapes has been applied in the field of shape matching and object recognition. Approximate symmetry has also been used for mesh enhancement. Distorted meshes can be corrected by converting the approximate symmetric patches to exactly symmetric patches [15,18]. Symmetry can also be applied to aid surface reconstruction from point data acquired by 3D scanners. The point data is often noisy and incomplete. Here, symmetry information can be exploited to construct better meshes. Symmetry can be used for segmentation and compression of the meshes too [17].

2.2 Symmetry in Scalar Fields

There has been very little work in the field of symmetry detection in scalar fields. Hong et al. use a parallel algorithm to detect reflective symmetry in scalar fields of volumetric data [9]. But the clear shortcoming is that it works only for global reflective symmetry. It does not address the problem of partial symmetry in scalar fields.

The latest work addressing the problem of partial symmetry detection in scalar fields is [19]. Here, a unique approach for detection of symmetry has been proposed. Instead of directly processing the scalar field to detect symmetry, similar subtrees in the contour tree of the scalar field are identified. The argument is that regions having similar contour trees are symmetric. But, regions having similar contour trees can have very different function distributions from a geometric perspective. For example, in Figure 2.1, the regions shown would be identified as symmetric using the contour tree approach. However, geometrically they are very dissimilar. The geometry of contours differ and the gradients are also quite different. The other major issues with the contour tree approach, are 1) tolerance to noise, and 2) restricting the regions to those defined by subtrees of contour tree. Two symmetric regions can have very different contour trees in the presence of noise. To overcome this problem, a contour tree stabilization procedure is prescribed. But, this may not handle large spikes in function values due to noise. Also, the detected regions are restricted to those defined by subtree of contour tree. So, symmetry among regions that are subset of contour tree regions or span across these regions will not be detected.

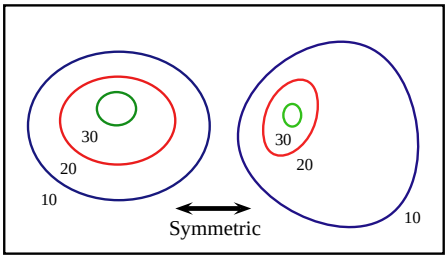


Figure 2.1: Non-intuitive symmetry detected using contour tree approach

The advantage of this approach is that it is very efficient. This is because contour tree of scalar field is usually much smaller than the scalar field itself. Therefore, symmetric regions can be efficiently detected for much larger datasets. Another advantage is that the solution works for any simply connected n-dimensional scalar field.

Another recent work which addressed the problem of partial Symmetry detection was by Kerber et.al. [11]. They proposed an approach of extracting line features and constructing the skeleton of 3D scalar fields in their earlier work [10]. The volume skeleton is used for symmetry detection in the scalar field. This approach is efficient but symmetry in skeleton may not correspond to actual symmetry in the scalar field and vice versa. Also, they show results for scalar fields of mechanical parts which have sharp creases and faces, and hence results in nice skeletal representation. However, extracting line forms for other datasets may not be easy, and thus the symmetry detection algorithm may not perform as well for data from other application domains.

Chapter 3

Background

3.1 Scalar Fields

An n-dimensional manifold is a space which locally resembles the n-dimensional Euclidean space, $\mathbb{R}^n$. A scalar field, s, is a scalar function defined on a manifold, $\mathbb{M}$.

$$s : \mathbb{M} \rightarrow \mathbb{R}$$

In this report, we consider scalar fields defined on either $\mathbb{R}^2$ or $\mathbb{R}^3$. Specifically, we consider scalar fields defined on a simply connected subset of $\mathbb{R}^n$. In practice, the scalar field is available as a discrete sample at vertices of a simplicial mesh that represents the domain. 2D scalar fields are defined on a triangle mesh and 3D scalar fields are defined on a tetrahedral mesh. Each vertex in a mesh is assigned a scalar value. Scalar values at rest of the points in the mesh are obtained by linear interpolation. For the rest of the report we will assume that the scalar fields are actually given as a mapping from a simplicial mesh to $\mathbb{R}$, and linear interpolation is used for calculating the scalar function at an arbitrary point in the domain.

3.2 Level Sets

Level set, L, for a given value, v, is the preimage of scalar field, $s : \mathbb{D}_s \to \mathbb{R}$ for that value.

$$L(v) = s^{-1}(v)$$

Level sets of 3D scalar fields are called isosurfaces, while the level sets of 2D scalar fields are called isocontours.

3.3 Gradient

The gradient of a scalar field, s, is a vector field that points in the direction of the greatest rate of increase of the scalar field, and whose magnitude is the greatest rate of change.

$$\text{grad}(s) = \nabla(s) = \left(\frac{\partial s}{\partial x_1}, \frac{\partial s}{\partial x_2}, \ldots, \frac{\partial s}{\partial x_n} \right)$$

Chapter 4

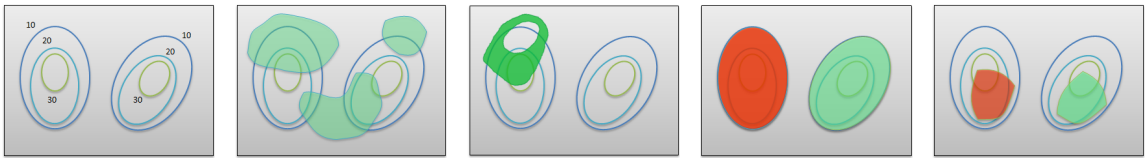
Symmetry in Scalar Fields

Loosely speaking, symmetry refers to invariance of the function distribution under a transformation. Here we will try to formalize the notion of symmetry in scalar fields. We want the formalization to take into account the geometry of the domain also.

4.1 Regions in Scalar Fields

For a scalar field $s : D_s \to \mathbb{R}$, any subset $r \subseteq D_s$ is a region of s.

A region r is called a connected region of s if r is path-connected. i.e. for all $(p, q) \in r^2$, there is a path $P \subseteq r$ that connects p and q.



(a) (b) (c) (d) (e)

Figure 4.1: (a) Level sets extracted for a scalar field. Three level sets of this scalar field are shown in different colors. (b) An arbitrary region in this scalar field. (c) A connected region. (d) Maximal symmetric region pair. (e) A symmetric pair that is not maximal.

4.2 Transformation

A transformation, T transforms the scalar field by transforming the domain or the range of s or both. Any transformation can be represented by transformation matrix, M_T.

4.2.1 Transformations on Domain of s

A transformation, $T_{\mathbb{D}}$, on domain is defined as a mapping from affine space to another. Following transformations would be considered for symmetry:

- **Translation:** The translation transformation maps a point $p \in \mathbb{D}$ to some point $p + \vec{t}$ where $\vec{t} \in \mathbb{R}^n$.
- **Uniform Scaling:** The scaling transformation maps a point $p \in \mathbb{D}$ to some point $s \times \vec{p}$ where $s \in \mathbb{R}$.
- **Rotation:** The rotation transformation maps a point $p \in \mathbb{D}$ to a point p_r where $p_r = M_r p$. Here, M_r is the $n \times n$ rotation matrix.
- **General Domain Transformation:** This is a combination of the above transformations.

4.2.2 Transformations on Range of s

We can also apply transformation on the range of s. Following range transformation would be considered:

- **Translation:** The translation transformation maps any value $v \in \mathbb{R}$ to some value $v + t$ where $t \in \mathbb{R}$.
- **Scaling:** The scaling transformation maps any value $v \in \mathbb{R}$ to some value $s \times t$ where $s \in \mathbb{R}$.
- **General Range Transformation:** A general transformation T_R combines translation and scaling transformations.

4.2.3 Transformation Sets

We will use the following notations for transformation sets:

$\mathbb{T}_{dtr}$: All domain transformations which are combination of domain translation and rotation.

$\mathbb{T}_D$: All domain transformations (combination of translation, rotation and uniform scaling).

$\mathbb{T}_R$: All range transformations (combination of translation and scaling).

$\mathbb{T}$: Set of all transformations possible on the scalar field. Each $T \in \mathbb{T}$ is a pair (T_D, T_R) where $T_D \in \mathbb{T}_D$ and $T_R \in \mathbb{T}_R$

4.3 Symmetric Regions

4.3.1 Exact Symmetry

Two regions r_1 and r_2 are exactly symmetric ($r_1 \simeq r_2$) under T,

$$r_1 \simeq r_2 \iff \forall p \in r_1, \exists p_T \in r_2 \text{ such that } p_T = T_D(p) \text{ and } T_R(s(p)) = s(p_T)$$

Clearly the relation $\simeq$ is an equivalence relation.

4.3.2 Approximate Symmetry

Roughly speaking two regions are approximately symmetric if the scalar field in the two regions do not differ too much and the scalar values are equal at a significantly large fraction of points.

Following is one attempt at formalizing the notion of approximate symmetry between regions:

λ-approximate : Given $\lambda \in \mathbb{R}$, two regions r_1 and r_2 are λ-approximate symmetric

$(r_1 \sim r_2)$,

$$r_1 \sim r_2 \iff \forall p \in r_1, \exists p_T \in r_2 \text{ such that}$$
$$p_T = T_D(p) \text{ and } |s(p_T) - T_R(s(p))| \leq \lambda$$

Here λ specifies the maximum permissible difference in function values of symmetric regions.

4.4 Symmetric Region Pair

Let $s : D_s \to \mathbb{R}$ be the scalar field. Let r_1 and r_2 be subsets of D_s. The ordered pair $(r_1, r_2)_T$ is called *symmetric region pair* of scalar field s, under transformation T, if

1. $r_1 \simeq r_2$ i.e. r_1 and r_2 are symmetric under T.
2. r_1 is connected. It immediately follows that r_2 will also be connected.
3. r_1 is maximal symmetric region of s under transformation T, i.e.

$$\nexists p \in Neighbourhood(r_1) \text{ such that } p_T = T_D(p) \in D_s \text{and } T_R(s(p)) = s(p_T)$$

The notion of connectedness and maximality of symmetric region pairs is explained visually in Figure 4.1.

4.5 Significance of Symmetric Region pair

The significance captures the importance of the symmetric region pair, $SP = (r_1, r_2)$. The significance, σ_{SP}, can be defined in many ways but we will consider the intuitive notion of significance, i.e. larger the regions, higher the significance of symmetric pair. Therefore

$$\sigma_{SP} \propto (Volume(r_1) + Volume(r_2))$$

If the domain of the scalar field has finite volume then, σ_{SP} is defined as

$$\sigma_{SP} = \frac{(Volume(r_1) + Volume(r_2))}{2 \times Volume(D_s)}$$

For scalar fields defined on simplicial meshes, the number of points in a region is often a good approximation of the volume of the region.

$$\sigma_{SP} = \frac{(NumPoints(r_1) + NumPoints(r_2))}{2 \times NumPoints(D_s)}$$

4.6 Problem Statement

Given the above definitions of symmetry and symmetric regions, we formally state the problem.

Problem Given a scalar field $s : D_s \to \mathbb{R}$, identify all symmetric region pairs with significance at least σ.

Symmetry can be required to be exact or λ-approximate as discussed earlier.

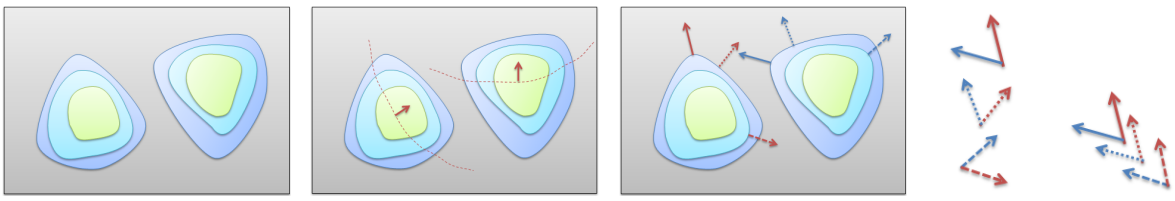
Chapter 5

Symmetry Identification Pipeline

5.1 Motivation for the solution

We notice that quantities like scalar values, gradient magnitude and curvature of the contour remain unchanged after geometric transformations of a region. The direction of gradient and tangent to the contour change, but they do so in a predictable manner. All the gradients and tangents are transformed by the specified transformation. Essentially, we know how regions behave under transformations.

In order to detect symmetric regions, we first search for point pairs in the scalar



(a) (b) (c) (d) (e)

Figure 5.1: (a) A scalar field. Three level sets of this scalar field are shown in different colors. (b) Symmetric regions present in this scalar field. The region in the directions pointed by arrows are geometrically symmetric. (c) A few normals of the ideal pairs in the symmetric regions. (d) Normals paired up. (e) Angles between the paired normals are equal. So, these ideal pairs will vote for the same transformation.

field that have same the function value, gradient magnitude and contour curvature. As already mentioned these quantities remain invariant under geometric transformations. Each such pair thus provides evidence for presence of a symmetric region. For each pair, we determine the transformation from the gradient direction and position of the points. For large symmetric regions, several pairs will vote for the same transformation. Thus, the voted transformations will appear as dense clusters in transformation space. So, the problem of finding symmetric regions in a scalar field is reduced to finding point pairs voting for a transformation, and lastly clustering in transformation space to obtain symmetric regions. The reader is referred to Figure 5.1 for a motivating example.

We now describe our proposed pipeline for identifying symmetric regions in the given scalar field. The scalar field is given as a scalar function defined on a discrete simplicial mesh.

5.2 Overview

This pipeline is inspired by Mitra et. al's pipeline for symmetry detection in geometric shapes [14]. We follow a similar two stage approach for identifying symmetry in scalar fields. The stages are:

1. Evidence accumulation stage: Accumulate evidence for various symmetries. This stage can also be called *Discovery* stage because here we discover the symmetries.

2. Validation stage: Verify the validity of symmetries obtained in previous stage. This can also be called extraction or region growing stage because the actual symmetric regions are extracted in this stage.

The pipeline is illustrated in Figure 5.2.

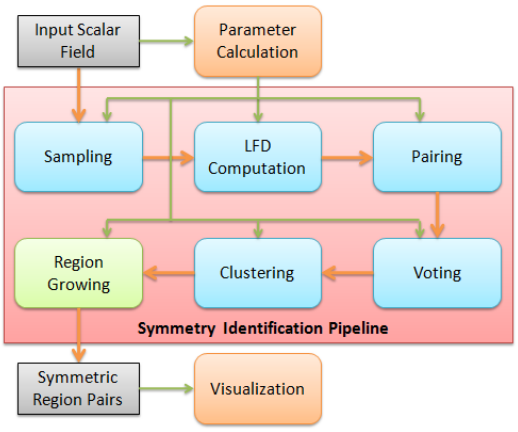


Figure 5.2: Symmetry Identification Pipeline

5.3 Evidence Accumulation Stage

5.3.1 Sampling

We sample the input domain because the number of points in the mesh can be very large. It should be noted that as we increase the sampling rate, upto a point evidence for new symmetries can be found. Further increasing the sampling rate will not result in new symmetries. The increased sampling will merely provide more evidence for previously found symmetries. So, sampling strategy along with the optimum sampling rate plays an important role in increasing the efficiency of this pipeline.

Strategies for sampling: We can follow various sampling strategies. Some are listed below:

- Random Sampling: Each point in the mesh has equal probability of being selected i.e. the sampling rate.
- Uniform Sampling: Sample such that sample points are uniformly distributed over the domain.
- Gradient Guided: Sample such that the probability of point p being sampled is proportional to the magnitude of the gradient of scalar field, s, at p. This helps in avoiding sampling from flat regions of the scalar field.

- Contour tree guided: In this approach, we use the contour tree [7,8] of the scalar field to guide the sampling. Each branch of the contour tree corresponds to some region, r, in the scalar field. The height, h, of the branch is the difference between the maximum and minimum value of the scalar function for the branch. We sample such that the probability of a point, $p \in r$ being sampled is proportional to $h \times$ Volume(r).

In our experiments, we use a combination of Uniform (or Random) and Gradient guided sampling strategies, where we sample the mesh only at the regions where gradient is greater than specified threshold. This ensures that flat regions are ignored. The output of the sampling stage is a set of sampled points, P_{smpld}.

5.3.2 Local Function Descriptor Computation

For each sample point, we compute the *Local Function Descriptor*.

Local Function Descriptor (LFD): This captures the properties of local function distribution around a point. The descriptor depends on the type of transformations allowed and the dimension of the domain. The LFD is used by the pairing and voting stages of the pipeline. We allow combinations of domain translation and domain rotation only for detecting symmetry. i.e. $T \in \mathbb{T}_{dtr}$.

Local Function Descriptor has two components: invariant component and alignment component:

1. Invariant component: This consists of all the properties that remain unchanged after transformation. As $T \in \mathbb{T}_{dtr}$, we can have following invariant properties:
   (a) Function value at the point, $s(p)$.
   (b) Magnitude of the gradient at the point, $|\nabla s(p)|$.
   (c) Curvature of the contour passing through the point. For 2D scalar field, the contour is a curve, so, curvature is a single real value, κ. For 3D scalar field, the contour is a surface, so, we consider minimum and maximum curvatures $(\kappa_{min}, \kappa_{max})$.

2. Alignment component: This consists of vectors used for alignment.

   (a) Gradient vector at the point p, $\nabla s(p)$. This is sufficient for alignment in 2D scalar fields.

   (b) Principal curvature directions of the local isosurface. In 3D we require additional vectors for alignment. We choose principal curvature directions (PC_{min}, PC_{max}) that are orthogonal to the gradient vector.

So, the LFD for 2D scalar field is:

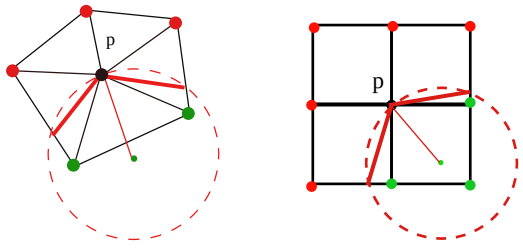
$$LFD_{2D}(p) = (s(p), |\nabla s(p)|, \kappa; \nabla s(p))$$

Similarly, the LFD for 3D scalar field is:

$$LFD_{3D}(p) = (s(p), |\nabla s(p)|, \kappa_{min}, \kappa_{max}; \nabla s(p), PC_{min}, PC_{max})$$

LFD computation involves computation of curvature of contour and the gradient vector. Gradient vector is approximated by computing the difference in function values of the neighbors of the point.

For 2D scalar fields, the contour passing through the point, p, is extracted locally within the 1-ring neighbors of p. The curvature is approximated as the inverse of the



(a) Triangle Meshes (b) Regular Grids

Figure 5.3: The contour passing through the point p is shown in bold red. The curvature is computed by taking the inverse of the radius of circle fitted to the contour, shown in red dashed line

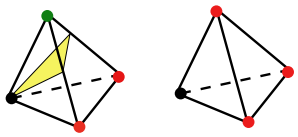


Figure 5.4: For extracting isosurface passing through a point(colored black) in a tetrahedral mesh, there are 2 cases as shown above

radius of the circle passing through the points of the contour. This is illustrated in Figure 5.3. A better approximation of curvature can be acheived by considering 2-ring (k-ring) neighbors of the point p.

For 3D scalar fields, local isosurface extraction for tetrahedral meshes is simple. There are essentially two possible configurations as shown in Figure 5.4. For regular meshes, local isosurface extraction is non-trivial (Please refer to Appendix B for details). Once, the local isosurface is extracted, the principal curvatures and directions are computed using the technique presented in [1,4].

5.3.3 Pairing

In the pairing stage, we find the point pairs that can provide evidence for symmetry in the scalar field.

In this stage, the first step prunes the sample set, P_{smpld}. The points that don't satisfy some properties are pruned from P_{smpld} to obtain a pruned sample set, P'_{smpld}. The properties which are satisfied by $p \in P'_{smpld}$ are as follows:

- $|\nabla s(p)| > 0$. This requirement helps avoid flat regions, where the vector does not help compute the alignment.
- For 3D case, $\kappa_{min} \neq \kappa_{max}$. Points where $\kappa_{min} = \kappa_{max}$ are called umbilic points. At such points curvature is equal in all directions, and there is no consistent (PC_{min}, PC_{max}) pair.

Algorithm 1 Pairing algorithm

Input: P_{smpld} – set of sampled points

Input: tol – tolerance vector specifying tolerance for invariant properties.

Input: gT – threshold for gradient magnitude.

Output: All pairs that can vote for a transformation.

procedure GETVOTERS(P_{smpld}, tol, gT)
  $V := \emptyset$
  $P'_{smpld} :=$ PRUNE(P_{smpld}, gT)
  $kDTree :=$ BUILDKDTREE(P'_{smpld})
  for all $p_i \in P'_{smpld}$ **do**
    $rangeQuery :=$ BUILDQUERY(p_i, tol)
    $result :=$ RANGEQUERY($kDTree, rangeQuery$)
    for all $p_j \in result$ **do**
      Add (p_i, p_j) to V
    end for
  end for
  return V
end procedure
procedure PRUNE($P, threshold$)
  $P' := \emptyset$
  for all $p \in P$ **do**
    $grad := LFD(p).|\nabla s|$
    $\kappa_{max} := LFD(p).\kappa_{max}$, $\kappa_{min} := LFD(p).\kappa_{min}$
    if $grad > threshold$ AND $\kappa_{max} \neq \kappa_{min}$ **then**
      Add p to P'
    end if
  end for
  return P'
end procedure

In the next step, all $\binom{n}{2}$ pairs are compared. Here $n = |P'_{smpld}|$. If the invariant component of the pairs are equal or within a user defined threshold, then they are added to the voter pairs set, V. Thus, the output of the pairing stage is a set of point pairs that are evidence of symmetry.

The pairing process is speeded up by building a kd-tree on the invariant component of LFD of points in P'_{smpld}. Here, for each point, we can determine the pairs by performing a range query on the kd-tree. The range query is built for each point using the tolerance allowed for each invariant component of LFD. Algorithm 1 computes the point pairs using kd-tree for the 3D case.

5.3.4 Voting

In this stage each pair, $(p_i, p_j) \in V$ votes for a transformation.

Representation of transformation In 2D, transformations in $\mathbb{T}_{dtr}$ can be represented by a 3-dimensional vector $T_{ij} = (t_x, t_y, r)$, where (t_x, t_y) is the 2D translation vector and r is the rotation angle. In 3D, the transformation is represented as a 6 dimensional vector $T_{ij} = (t_x, t_y, t_z, r_x, r_y, r_z)$ where (t_x, t_y, t_z) is the 3D translation vector and r_x, r_y and r_z are the Euler angles for rotation.

Determination of transformation vector Each pair, $(p_i, p_j) \in V$ votes for a transformation. The transformation vector is computed with the help of alignment component of LFD's of p_i and p_j.

For 2D case, r is determined by the angle between the gradient vectors at p_i and p_j. Let the rotation matrix corresponding to the rotation r be R. Then the translation vector (t_x, t_y) is computed as $p_j - Rp_i$.

In 3D, each point p has local coordinate frame defined by $(\nabla s(p), PC_{min}, PC_{max})$. To determine the transformation vector, first the local coordinate frames of the points p_i and p_j are aligned. Let R be the rotation matrix corresponding to the alignment. We can determine the Euler angles (r_x, r_y, r_z) from this rotation matrix R. Again, the

Algorithm 2 Voting algorithm

Input: V – set of pairs selected for voting

Output: Set of votes in transformation space.

procedure GetVotes(V)
  $\mathbf{T} := \emptyset$
  for all $(p_i, p_j) \in V$ **do**
    $t :=$ Vote(p_i, p_j)
    Add t to $\mathbf{T}$
  end for
  return T
end procedure
procedure Vote(p_i, p_j)
  $T(t_x, t_y, t_z, r_x, r_y, r_z) := (0, 0, 0, 0, 0, 0)$
  $F_i := LFD_{p_i}(\nabla s, PC_{min}, PC_{max})$
  $F_j := LFD_{p_j}(\nabla s, PC_{min}, PC_{max})$
  $R_{ij} := F_j F_i^{-1}$ ▷ Compute the rotation matrix
  $T(r_x, r_y, r_z) :=$ EulerAngles(R_{ij})
  $t_{ij} := p_j - R_{ij} p_i$ ▷ Compute the translation vector
  $T(t_x, t_y, t_z) := t_{ij}$
  return T
end procedure

translation vector (t_x, t_y, t_z) is computed as $p_j - Rp_i$.

The output of the voting stage is a set of transformations, **T**. The voting algorithm for the 3D case is described in Algorithm 2.

5.3.5 Clustering

To determine significant symmetries in the scalar field, clustering is applied on the transformations **T** obtained in the previous stage.

The points **T** lie in a six-dimensional transformation space (Transformation space is three dimensional for 2D scalar fields). Significant symmetries manifest as clusters in the transformation space.

We have various options for performing clustering on transformation space:

- Single Linkage Clustering: This is a very simple and fast clustering algorithm. But the disadvantage is that it can give linear clusters, which is not what we want.
- DBSCAN [6]: This gives better clusters than Single Linkage clustering but it can also give linear clusters. One major advantage of this algorithm is that it identifies the points that cannot be assigned to good clusters as noise.
- Mean shift clustering [5]: This algorithm gives the best clusters for our purpose, but it is slower as compared to the algorithms discussed earlier. This algorithm proceeds by moving each point in the direction of density gradient and terminating when the basin of attraction is reached. The basin of attraction is a better alternative than using centroid for region growing stage.

A cluster's significance is determined by the number of points assigned to that cluster. A cluster in transformation space is large only if many point pairs vote for the same transformation, thus indicating a large symmetric region. We rank the clusters in decreasing order of their significance, and compute the centroid of the cluster as the representative transformation from the cluster. So, the final output of the clustering stage is a ranked list of transformations.

5.4 Validation Stage

In the validation stage we validate whether the transformations obtained from the accumulation stage are valid significant symmetries. It may happen that pairs from different non-significant regions vote for the same transformation, thus resulting in a large cluster. We prune such transformations during the validation step, and also re-rank the transformations.

Algorithm 3 Region Growing

Input: $clusters$ – clusters in transformation space

Input: σ_{min} – minimum significance

Output: Set of Symmetric Region Pairs.

procedure GROWREGIONS($clusters, \sigma_{min}$)
  $SRPs := \emptyset$
  for all $cluster \in clusters$ **do**
    while $cluster \neq \emptyset$ **do**
      $(p_1, p_2) :=$ GETRANDOMPAIR($cluster$)
      $cluster := cluster \setminus \{(p_1, p_2)\}$
      $R_1 := \{p_1\}, R_2 := \{p_2\}$
      $T := transformation(p_1, p_2)$
      while

$$\exists p_i \in Neighbourhood(R_1) \text{ and } p_j \in Neighbourhood(R_2)$$
$$\text{such that } Tp_i = p_j \text{ and } s(p_i) = s(p_j)$$

      do
        $cluster := cluster \setminus \{(p_i, p_j)\}$
        Add p_i to R_1 and p_j to R_2
        $T :=$ BESTTRANS(R_1, R_2)
      end while
      $SRP := (R_1, R_2)$
      if $\sigma(SRP) \geq \sigma_{min}$ **then**
        Add SRP to $SRPs$
      end if
    end while
  end for
  return $SRPs$
end procedure

During validation stage, region growing is also done, to determine the actual region(s) that voted for a particular transformation. So, significant symmetric regions are grown in this final validation stage. The output of the validation stage is a set of Symmetric Region Pairs (r_i, r_j) along with the associated transformation, T_{ij} under which they are symmetric. Algorithm 3 outlines the region growing procedure.

In the algorithm we are using BestTrans to incrementally update the transformation for the region being grown. This updation of transform is done by using the approach proposed in [2] . We can ignore this step to increase efficiency at the cost of quality.

5.5 Summary

Algorithm 4 summarizes the symmetry identification pipeline. The user provides as input the scalar field s, sampling rate δ, and significance σ. The output of the pipeline is set of all the detected Symmetric Region Pairs with significance at least σ.

Algorithm 4 Symmetry Identification algorithm

Input: s – discrete scalar field defined on simplicial mesh

Input: σ – significance

Input: δ – sampling rate

Output: All detected MSCRPs.

procedure IDENTIFYSYMMETRY(s, σ, δ)

P_{smpld} := SAMPLE(s, δ)

COMPUTELFDS(P_{smpld})

V := GETVOTERS($P_{smpld}, tolerance, gradThres$)

$\mathbf{T}$:= GETVOTES(V)

$clusters$:= CLUSTER($\mathbf{T}$)

$regions$:= GROWREGIONS($clusters, \sigma$) ▷ Validation

return $regions$

end procedure

Chapter 6

Results

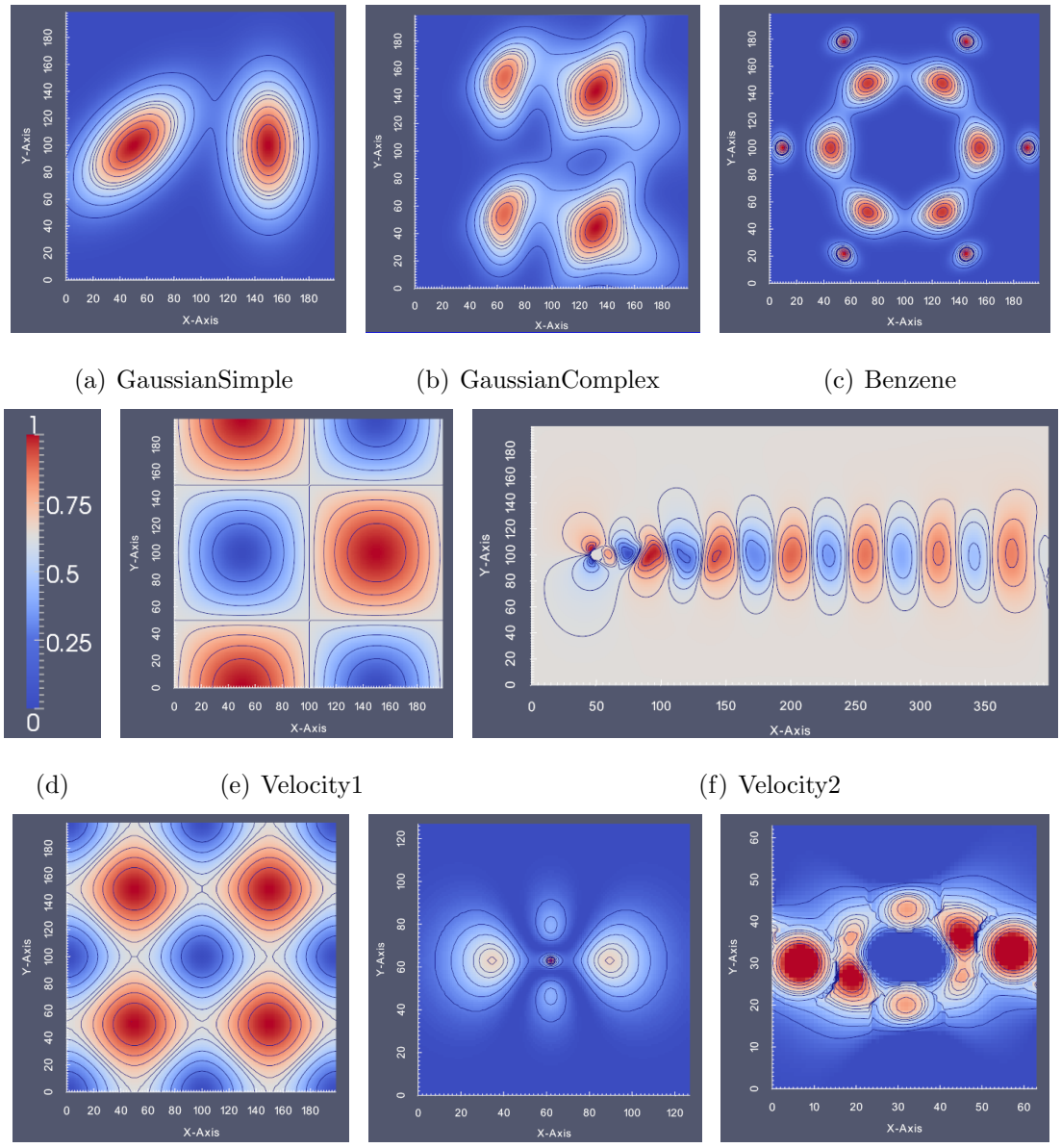
For experimentation three types of datasets were used viz. Synthetic, Simulation and Slices from 3D scalar fields. Figure 6.1 shows these datasets. The range of all these scalar fields was normalized to fall between 0 and 1. This normalization does not affect the symmetries and symmetric regions in anyway. In Figure 6.1, a diverging color map is used which assigns blue hues to low scalar values and red hues to high scalar values.

6.1 Synthetic Data

Synthetic 2D scalar fields were generated by adding 2D Gaussian functions at different positions. These added Gaussians could have different standard deviations and orientations. Using this approach arbitrarily complex synthetic datasets can be generated with known symmetries.

6.1.1 GaussianSimple

This 200×200 dataset is shown in Figure 6.1(a). This dataset is generated by first adding a 2D Gaussian with different standard deviations in X and Y directions to a Zero scalar field. Then the same Gaussian is rotated by 45°and added at a different location to the scalar field. We thus obtain a dataset containing a single Symmetric Region pair i.e. the two Gaussians. However it must be mentioned that any 2D Gaussian is highly

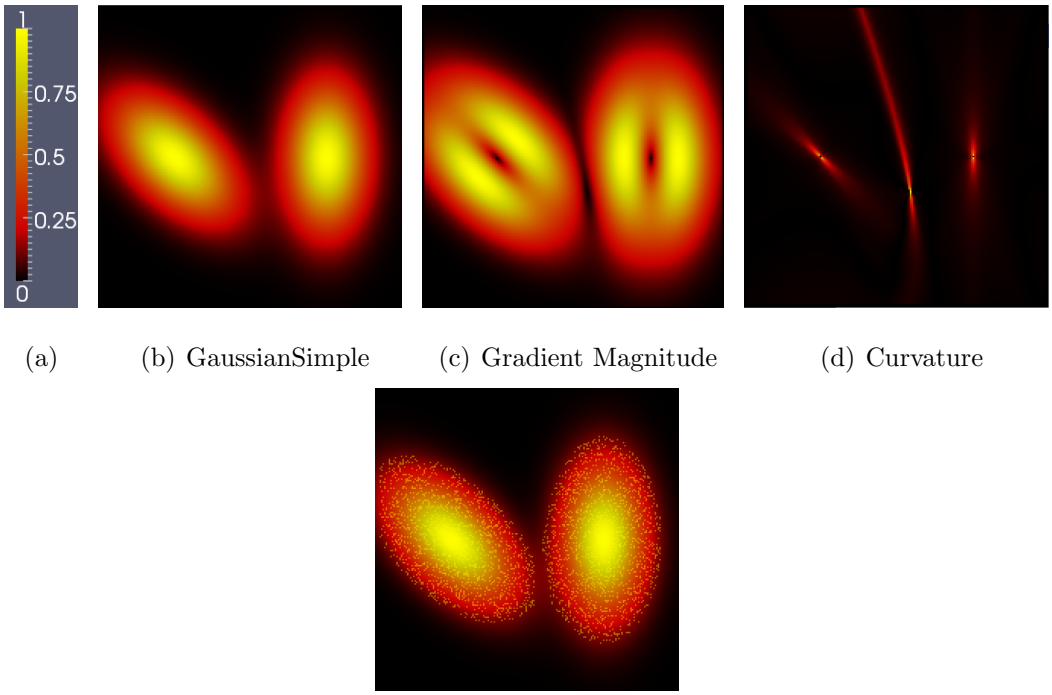


(a) GaussianSimple (b) GaussianComplex (c) Benzene

(d) (e) Velocity1 (f) Velocity2

(g) Pressure (h) Hydrogen (i) Neghip

Figure 6.1: The set of figures above shows the 8 datasets used in Experimentation. (d) The diverging color map used. (a)–(c) are synthetic datasets generated by adding 2D Gaussian functions with different standard deviations in different orientations and positions. (e) – (g) are fluid simulation datasets. They represent scalar quantities like velocity magnitude and pressure in different simulations. (h) and (i) are slices of real 3D volume datasets. (h) is a slice from Hydrogen dataset. (i) is a slice from Neghip dataset.



(a) (b) GaussianSimple (c) Gradient Magnitude (d) Curvature

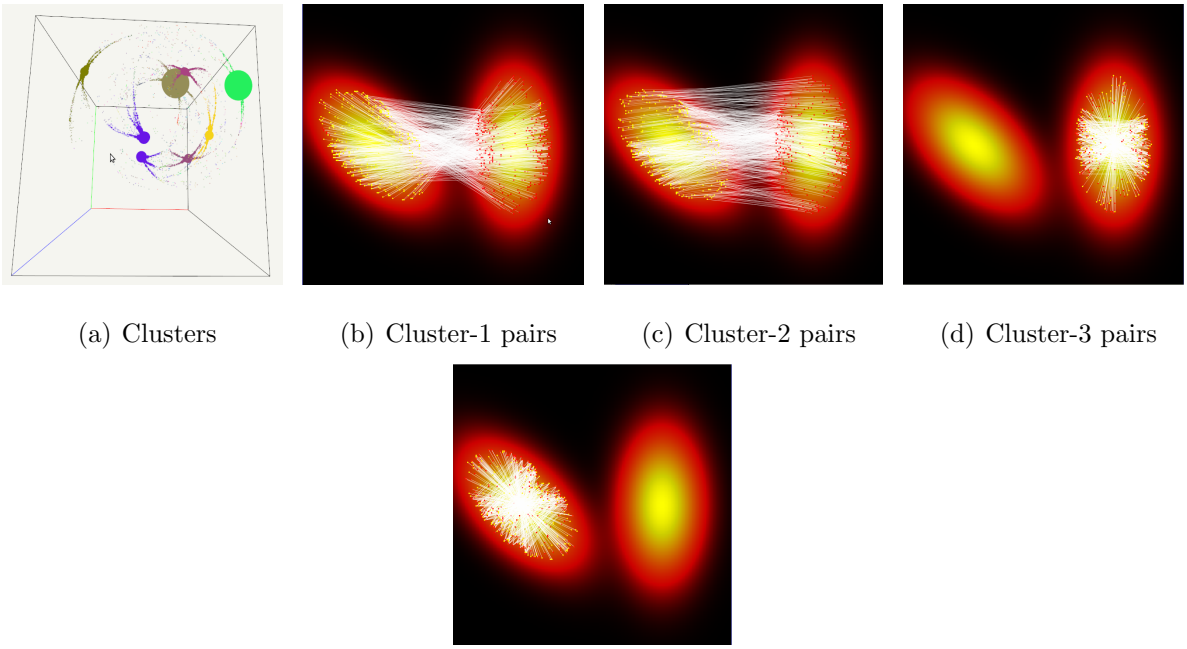
(e) Samples

Figure 6.2: (a) Color map used. The same color map is used for later figures too. (b) The GaussianSimple data set. (c) Gradient magnitude of the GaussianSimple data set. (d) Contour curvature scalar field. (e) Sample points: sampling is controlled so that very low gradient points are not sampled. After sampling stage, pairing of the sample points is performed. The pairing stage uses the scalar fields shown in (b) – (d) to pair up the sample points. Points with similar values for these parameters (LFD) are paired up and vote for a transformation.

symmetric, it has 2 reflective symmetries and a 180°rotational symmetry with itself. For now, we are ignoring the reflective symmetries.

Figure 6.2 shows the sampling stage of the pipeline. Here we use the gradient magnitude field to avoid flat regions by sampling only in regions having gradient magnitude above a specified threshold. Next the sampled points are paired up if their LFDs are similar i.e. the difference in the shown scalar fields (6.2(b) – 6.2(d)) is below a specified threshold.

After pairing, the voting for transformations takes place. Then clustering is applied on these transformations to detect symmetries. Figure 6.3(a) shows the transformation



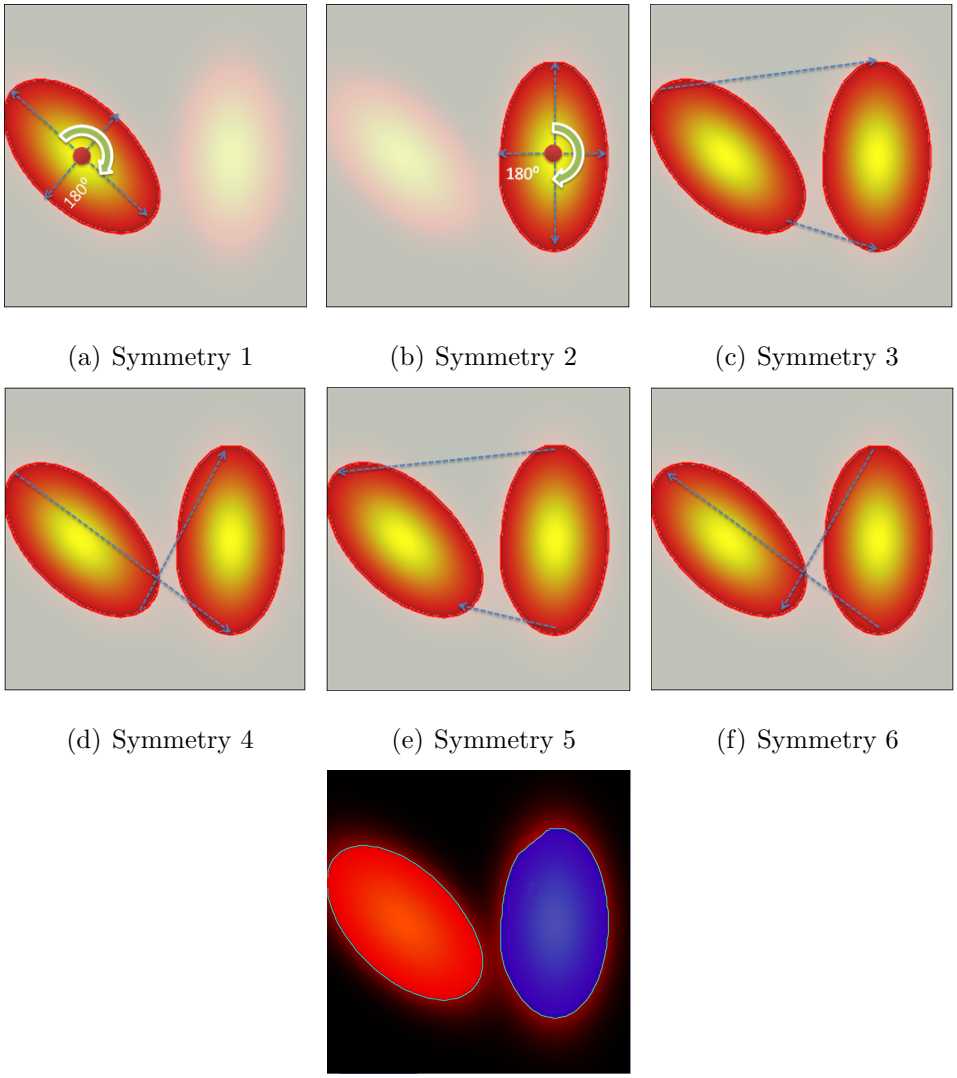
(a) Clusters (b) Cluster-1 pairs (c) Cluster-2 pairs (d) Cluster-3 pairs

(e) Cluster-4 pairs

Figure 6.3: (a) Clusters in the transformation space obtained for GaussianSimple Dataset. Different clusters are represented by different colors. (b) – (e) show the pairs corresponding to some of the clusters.

space and the clusters obtained. Figure 6.3(d) and 6.3(e) shows the pairs which vote for 180°Self symmetry transformation for the two Gaussians. Figure 6.3(b) shows the pairs which vote for 225°rotational symmetry across the two Gaussians, while Figure 6.3(c) shows the pairs voting for 45°rotational symmetry.

After clustering, region growing is performed to obtain Symmetric regions. The results after region growing stage is shown in Figure 6.4. Figures 6.4(a) – 6.4(f) show the 6 symmetries obtained for this dataset. Two of these are 180°self symmetries. The 45°and 225°symmetries across the two Gaussians, along with their inverses, are the remaining four symmetries. Figure 6.4(g) shows the Symmetric region pair obtained. It can be visually verified that this pair is actually symmetric and corresponds to the two Gaussians in the dataset.



(a) Symmetry 1 (b) Symmetry 2 (c) Symmetry 3

(d) Symmetry 4 (e) Symmetry 5 (f) Symmetry 6

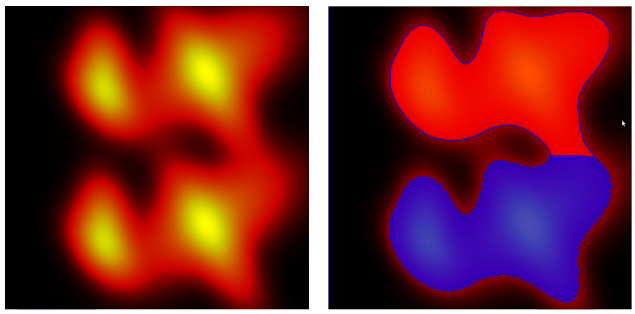
(g) Symmetric Region Pair

Figure 6.4: (a) – (f) show the symmetries detected in the GaussianSimple dataset. In (a) and (b), the 180°self-symmetry is shown. (c) 45°rotational symmetry among the two regions shown. The dotted arrows show where the points will be transformed after transformation. (d) 225°rotational symmetry detected between the same regions, this happens because each Gaussian has 180°rotational self symmetry. (e) and (f) are inverse transformations of those depicted in (c) and (d) respectively. So, in total we detect 4 transformations under which the two Gaussians are symmetric. (g) The Symmetric Region Pair detected in GaussianSimple dataset.

6.1.2 GaussianComplex

This scalar field is shown in 6.1(b). This is also a 200×200 dataset. As mentioned in previous section, the 2D Gaussian function exhibits high self-symmetry, which results in many transformations for the same region. To avoid that problem in this dataset, four different Gaussians were overlapped to avoid self symmetries in the region. This gives a more complex region. Now, this region is translated and added to the scalar field, resulting in a more complex symmetric region pair.

After executing the Symmetry detection pipeline, the symmetric region pair detected is shown in Figure 6.5(b).



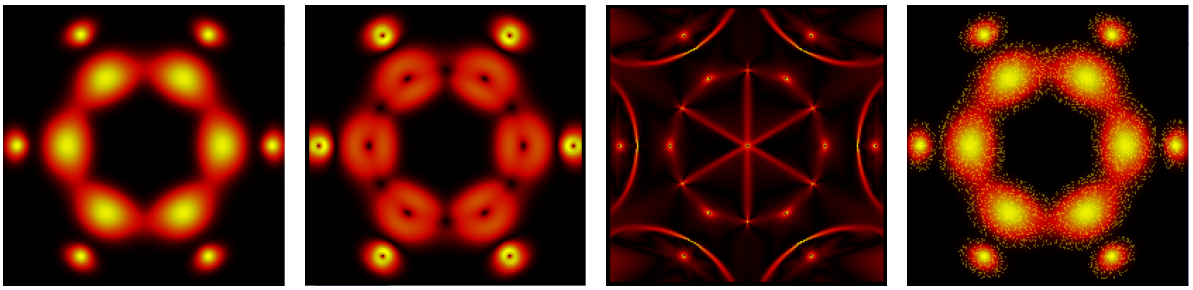
(a) GaussianComplex (b) Symmetry

Figure 6.5: (a) The GaussianComplex data set. (b) The detected Symmetric Region.

6.1.3 Benzene

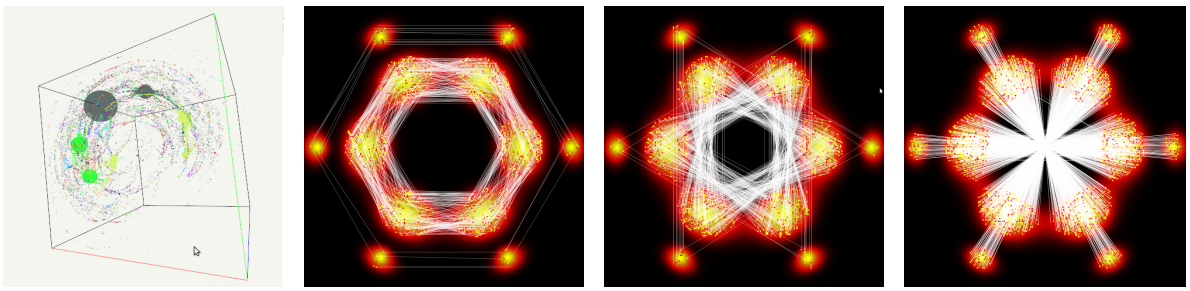
This is another synthetically generated 200×200 dataset. It is shown in 6.1(c). The idea was to generate a complex dataset with many symmetries. So, a dataset which resembles a Benzene molecule was created. Benzene has six Carbon atoms and six Hydrogen atoms. The dataset consists of six sub-regions consisting of one Carbon and Hydrogen atom each. These six sub-regions are symmetric and radially distributed in sectors of 60°. This distribution results in many symmetries, the whole scalar field becomes symmetric to itself under 60°rotation group.

Figure 6.6 shows the sampling stage. Figure 6.7(a) shows the clusters obtained in the transformation space for this dataset. Figure 6.7(b) – 6.7(d) show the pairs voting



(a) Benzene Dataset (b) Gradient Magnitude (c) Curvature (d) Samples

Figure 6.6: (a) Scalar field from the Benzene dataset. (b) Gradient magnitude of the Benzene data set. (c) Contour curvature scalar field. (d) Sample points: sampling is controlled so that very low gradient points are not sampled.



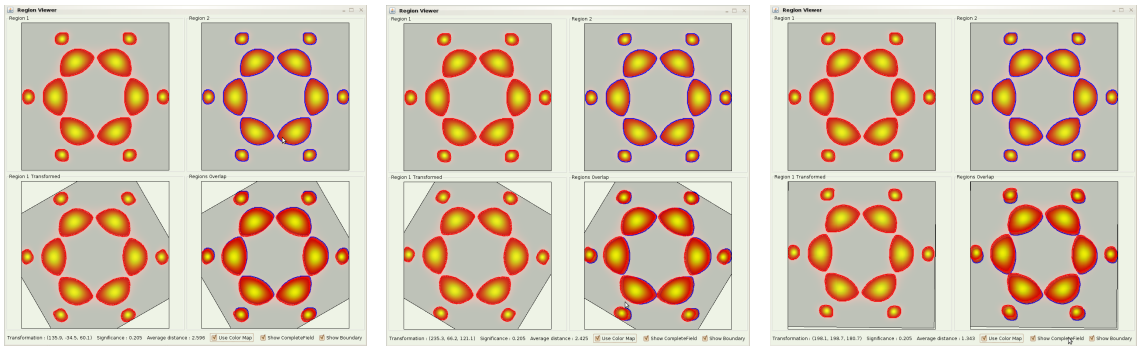
(a) Clusters (b) Cluster-1 pairs (c) Cluster-2 pairs (d) Cluster-3 pairs

Figure 6.7: (a) Clusters in the transformation space obtained for Benzene Dataset. (b) – (d) show the pairs corresponding to some of the clusters. It should be noted that there were many other clusters obtained. We show only three of the clusters.

for three of those clusters. They are pairings for 60°, 120°and 180°rotations respectively. Figure 6.8(a) – 6.8(c) show some of the Symmetric regions detected in the dataset.

6.2 Simulation Data

We next report experimental results on datasets from fluid simulation representing quantities like velocity and pressure. Even though these datasets are not noisy, they contain approximate and not necessarily exact symmetries.



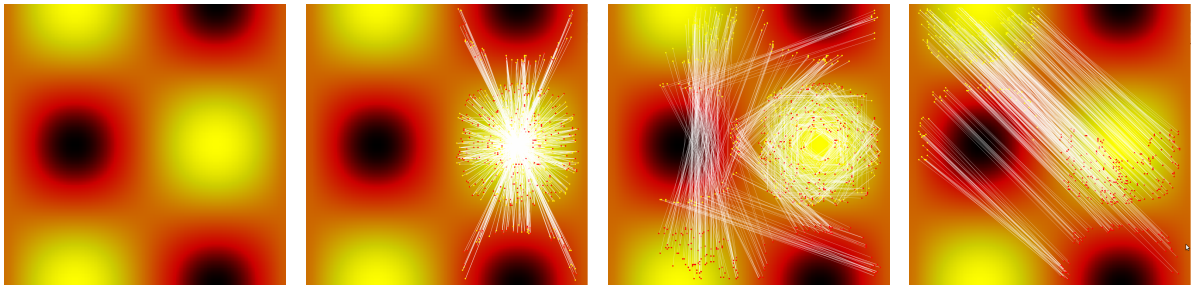
(a) Symmetry 1 (b) Symmetry 2 (c) Symmetry 3

Figure 6.8: Result obtained after region growing stage. Regions are by definition maximal and connected but here we show all the regions for a particular cluster as a single region. (a) 60°symmtery. Region-1 is shown in top-left part of the figure, top-right shows the Region-2. The bottom-left portion of the figure shows the transformed version of Region-1. The bottom-right shows how the Region-1 and Region-2 overlap after transformation. (b) and (c) show 120°and 180°rotational symmteries present in Benzene dataset.

6.2.1 Velocity1

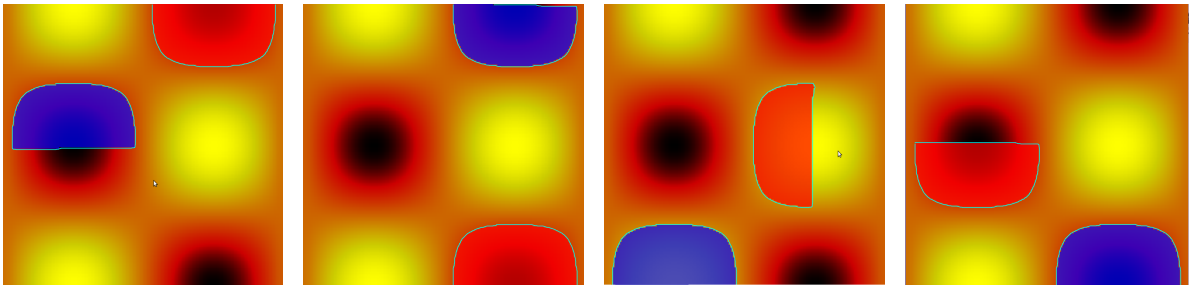
The dataset is shown in Figure 6.1(e). This dataset has many exact symmetries. Some of the clusters obtained are shown in Figures 6.9(b) – 6.9(d). They show a representative set of the many translational and rotational symmetries.

Figures 6.10 shows some of the Symmetric region pairs. As there are many symmetries in this dataset, we use a Symmetry Graph to visualize these symmetries. In a Symmetry graph, nodes represent the regions, the edges represent the symmetries. Two nodes are connected if they are symmetric under some geometric transformation. Because of the transitive nature of symmetries, any connected component of the symmetry graph will give the set of regions which are symmetric to each other. Figure 6.11(a) shows the symmetry graph for Velocity1 dataset. It must be mentioned here that it is not the complete Symmetry graph for this dataset. Some symmetries are not considered to avoid clutter. Figure 6.11(b) and 6.11(c) show the two connected components of the Symmetry graph. It can be visually verified that the regions belonging to these connected components are actually symmetric to each other.



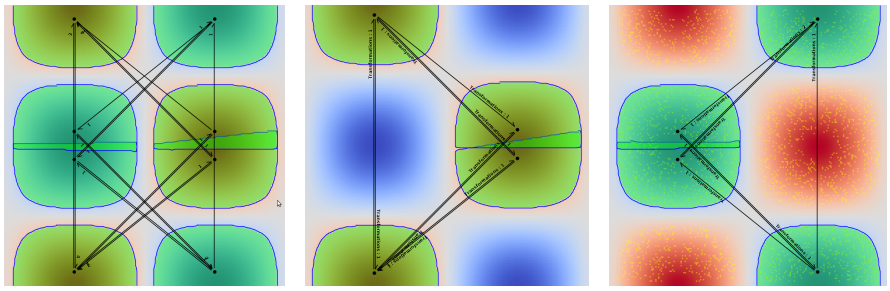
(a) Velocity1 (b) Cluster-1 pairs (c) Cluster-2 pairs (d) Cluster-3 pairs

Figure 6.9: (b) – (c) pairs corresponding to some of the clusters obtained in Velocity1 dataset



(a) Symmetric Region 1 (b) Symmetric Region 2 (c) Symmetric Region 3 (d) Symmetric Region 4

Figure 6.10: Some of the detected Symmetric Region Pairs after region growing stage.



(a) Symmetry Graph (b) (c)

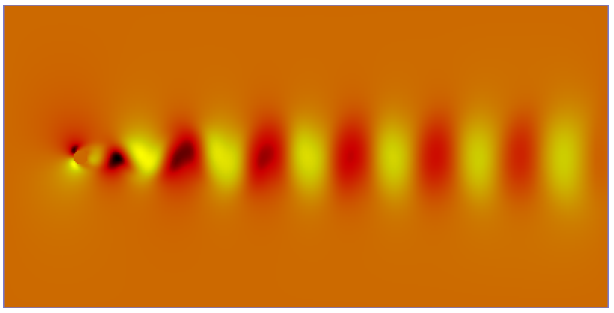
Figure 6.11: The Velocity1 dataset has many symmetries. The Symmetry Graph is used to visualize them effectively. (a) The Symmetry Graph. The regions are nodes of the graph, a node is connected to another node if a transformation is detected by the pipeline i.e. two nodes are connected if they are symmetric. The edges are labeled by number of transformations detected for that region pair. (b) and (c) Two Connected components in the Symmetry graph. The regions in each connected component are symmetric to each other.

6.2.2 Velocity2

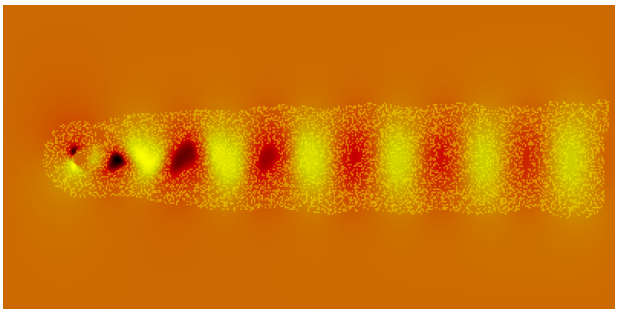
This is a 400 × 200 dataset with no clear exact symmetries. However, we can see that some of the regions show approximate symmetries. The symmetry detection pipeline successfully identified many of these approximate symmetries.

Figures 6.13(a) and 6.13(b) show some representative clusters obtained for these datasets. Most of the clusters were translational symmetries as shown in 6.13(a), while some of them were 180°rotational symmetries as shown in 6.13(b). Figures 6.13(c) and 6.13(d) show the result after the region growing stage. These figures show all the symmetric region pairs detected from a single cluster. Figure 6.13(c) shows translational symmetry while 6.13(d) shows 180°rotational symmetry.

Some individual symmetric region pairs are shown in Figures 6.14(a) – 6.14(d). The regions constituting these pairs are indicated by red and blue patches on the scalar field.

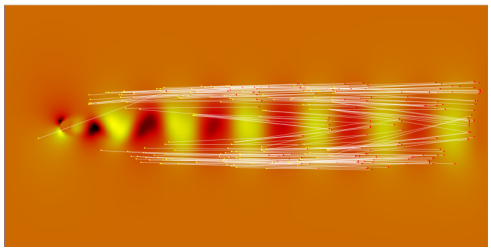


(a) Velocity2 Dataset

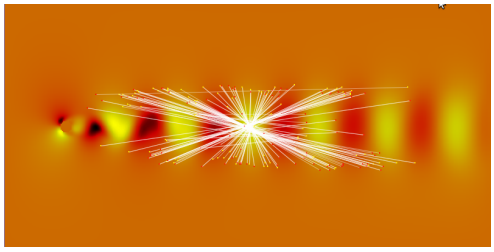


(b) Samples

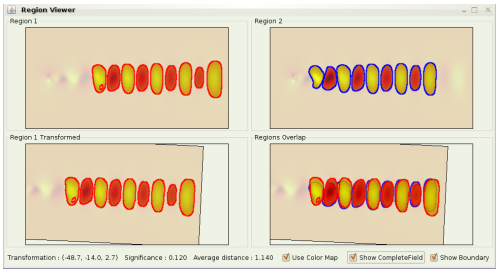
Figure 6.12: (a) The Velocity2 dataset. (b) Samples from the input. The large flat region is ignored by avoiding points having low gradient.



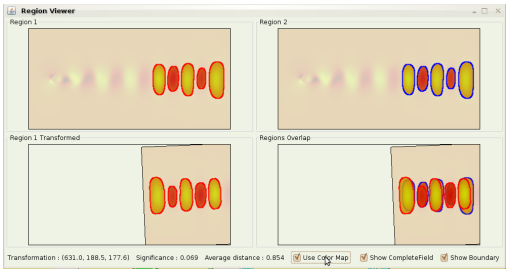
(a) Cluster-1 pairs



(b) Cluster-2 pairs

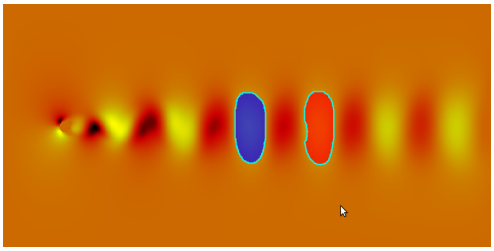


(c) Symmetry 1

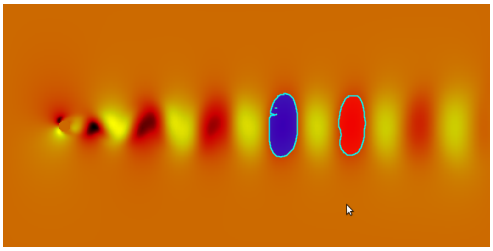


(d) Symmetry 2

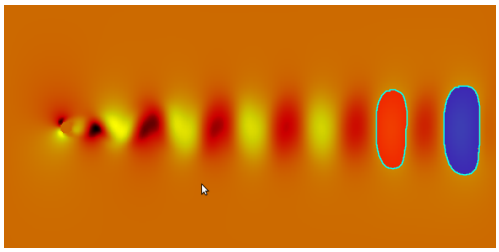
Figure 6.13: (a) and (b) Some representative detected clusters. Major symmetries detected in this dataset are translational and 180°rotational symmetries.(c) One translational Symmetry. (d) A 180°rotational symmetry.



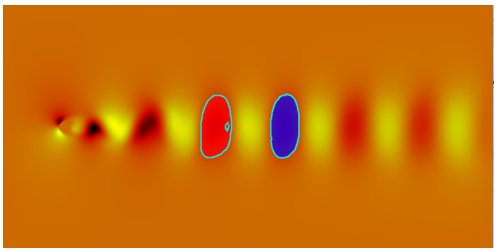
(a) Symmetric Region 1



(b) Symmetric Region 2



(c) Symmetric Region 3



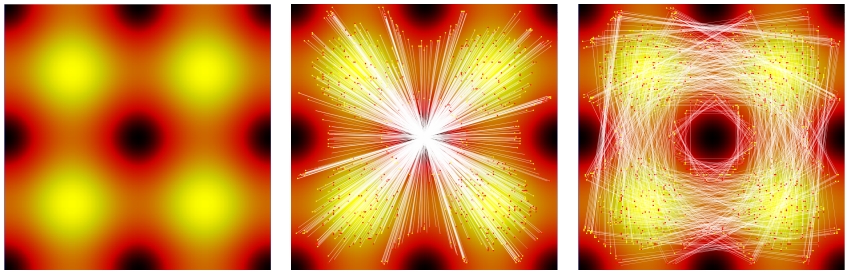
(d) Symmetric Region 4

Figure 6.14: (a) – (d) Symmetric Regions. It should be noted that these regions show approximate symmetry rather than exact symmetry.

6.2.3 Pressure

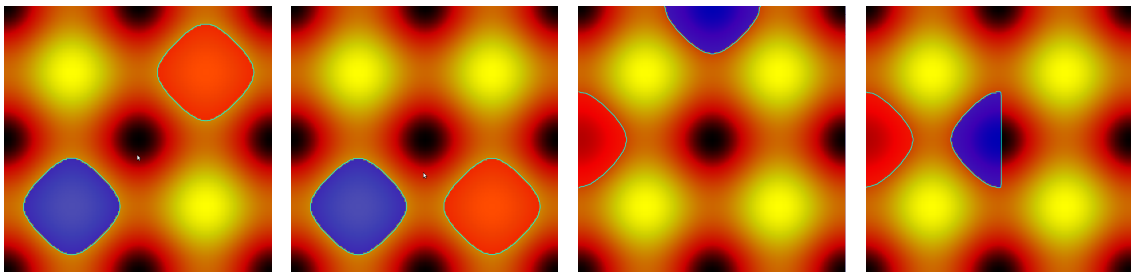
This dataset represents a measurement of pressure for a fluid simulation. This depicts a large number of exact symmetries.

Two of the cluster pairs are shown in Figures 6.15(b) and 6.15(c). Figures 6.16(a) – 6.16(f) show a representative set of symmetric region pairs detected in this dataset. To visualize the huge number of Symmetries, we again use Symmetry Graph. The Symmetry Graph along with three of its connected components is shown in Figures 6.17(a) – 6.17(d).

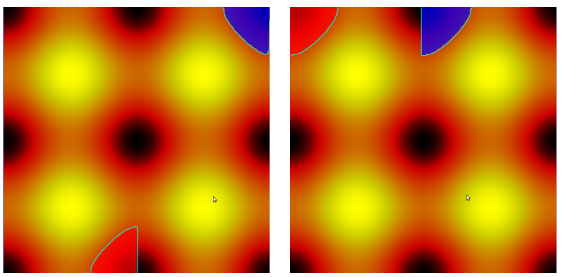


(a) Pressure Dataset (b) Cluster-1 pairs (c) Cluster-2 pairs

Figure 6.15: (b), (c) Pairs corresponding to two of the clusters obtained in Pressure dataset

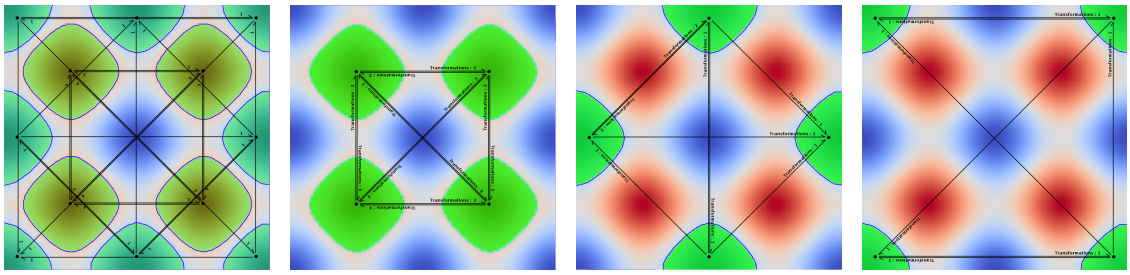


(a) Symmetric Region 1 (b) Symmetric Region 2 (c) Symmetric Region 3 (d) Symmetric Region 4



(e) Symmetric Region 5 (f) Symmetric Region 6

Figure 6.16: Symmetric region Pairs detected in the Pressure dataset.



(a) Symmetry Graph (b) (c) (d)

Figure 6.17: Symmetry Graph is used to visualize the symmetries detected in the Pressure dataset. (a) The Symmetry Graph. (b) – (d) Three connected components in the symmetry graph.

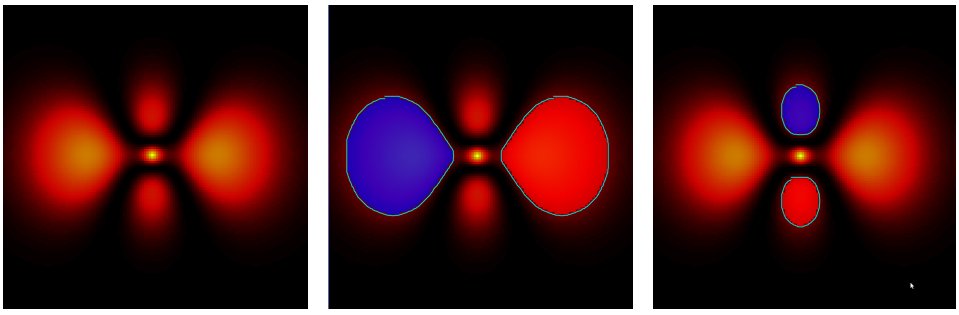
6.3 Slices from 3D scalar fields

We next report symmetries detected from slices of 3D scalar fields.

6.3.1 Hydrogen

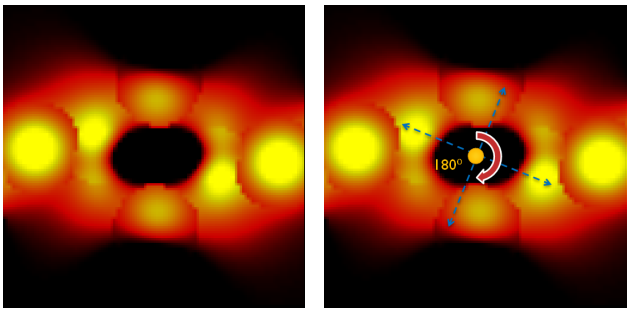
This dataset is an XZ slice with $Y = 64$ taken from $128 \times 128 \times 128$ hydrogen[1] dataset. This slice has two Symmetric region pairs and has 180°rotational symmetry. These two symmetric regions were successfully detected by the pipeline as shown in Figures 6.18(b) and 6.18(c).

[1] www.volvis.org



(a) Hydrogen (b) Symmetric Region 1 (c) Symmetric Region 2

Figure 6.18: The two Symmetric Regions Pairs detected in the Hydrogen dataset.



(a) Neghip (b) Detected Symmetry

Figure 6.19: The 180°rotational symmetry is detected by this technique.

6.3.2 Neghip

This dataset is an XZ slice with $Y = 20$ taken from $64 \times 64 \times 64$ neghip[2] datset. This slice has 180°rotational symmetry which was successfully detected by the pipeline as shown in Figure 6.19(b).

6.4 Performance

Table 6.1 lists the runtimes of the various stages of the pipeline for all the datasets. The experiments were performed on a workstation with a dual core Intel Xeon 5130 2.00GHz processor and 4GB main memory. The implemetation of the symmetry detection algorithm is done in Java (JRE-6). We believe this basic unoptimized Java implementation is much slower than what we can achieve by parallel implementations using C++.

From the results shown in Table 6.1, it is clear that clustering stage is the most time consuming stage. However time spent in clustering depends on the number of candidate pairs generated by pairing stage. Region growing also requires significant time. The runtimes for region growing shown in the table are for naive region growing without incremental transform refining. The robust region growing algorithm was used for some datasets. These runtimes are mentioned in brackets. Time requirements for sampling, pairing and voting are negligible.

[2]www.volvis.org

Table 6.1: Results

Dataset	Results					Time taken (sec)			
	Samples	Pairs	Significant Clusters			Sampling	Pairing & Voting	Clustering	Region Growing
			Number	Smallest	Biggest				
GaussianSimple	2800	16538	6	1435	3930	0.064	0.758	14.741	2.021 (42.89)
GaussianComplex	2800	2250	3	231	364	0.096	0.271	0.281	0.573 (17.92)
Benzene	4000	12122	6	595	2169	0.234	0.486	8.213	1.307 (57.91)
Velocity1	2800	14370	21	146	1380	0.108	0.501	9.896	1.671
Velocity2	7646	4742	15	20	581	0.312	0.393	0.859	0.141
Pressure	2800	27254	36	154	1243	0.264	1.162	43.218	1.312
Hydrogen	877	3738	2	230	596	0.167	0.199	1.508	0.514
Neghip	704	484	1	–	442	0.104	0.229	0.373	0.254

6.5 3D scalar fields

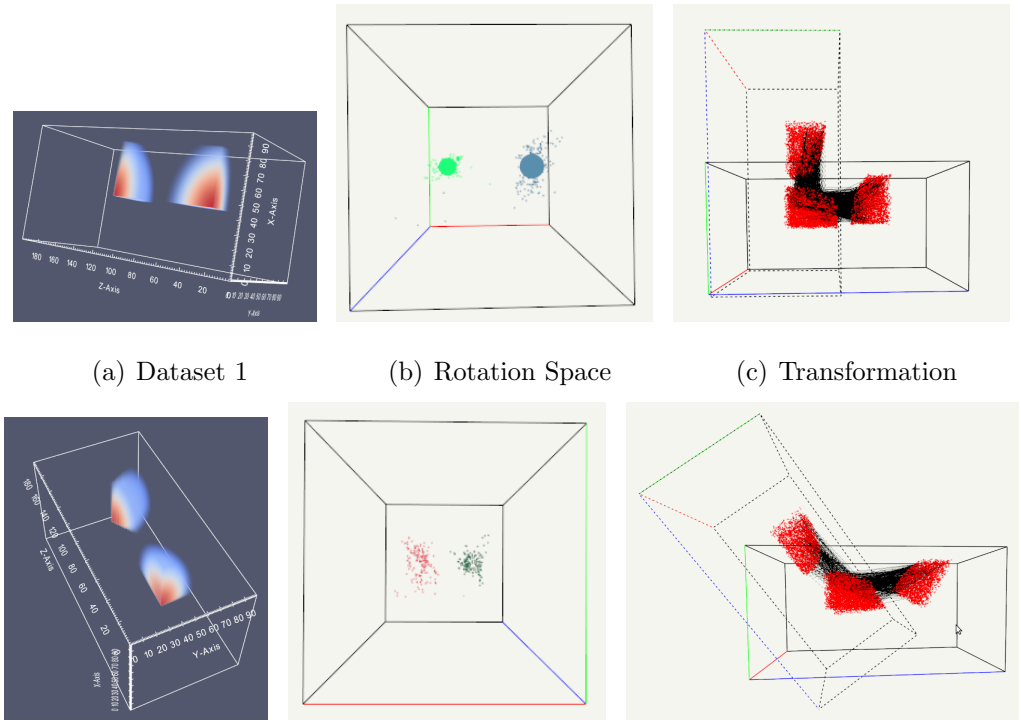
Till now we have discussed results for 2D scalar fields. The pipeline was implemented for 3D scalar fields too, but the results obtained were not very promising. Even for synthetic datasets with simple symmetries, the clusters obtained in the transformation space were not dense enough. Figure 6.20 show some experimental results for 3D scalar fields.

The dataset shown in Figure 6.20(a) is generated by combining a 3D Gaussian function and its copy after rotating by 90°about X-axis. The Gaussians are restricted to one octant to avoid cluttering of transformation space. Figure 6.20(b) shows the clusters in the rotation transformation. As expected two very dense clusters at 90°and -90°are obtained. Figure (c) shows one of the transformations applied to the volume. The two regions (which are represented by the dense sample points in the figure) are clearly aligned after transformation.

The Figures 6.20(d) – 6.20(f) show similar results when the symmetry detection pipeline is executed for the dataset generated by 60°rotation instead of 90°. Now the clusters in the transformation space are not dense enough, they are spread out as seen in Figure (e). The reason for this cluster spreading is that after 60°rotation the points are not grid-aligned. The absence of dense clusters even for synthetic datasets indicated that the experiments with real 3D data may not be successful.

Some of the reasons and possible solutions for the problem above:

- The increase in dimension of transformation space from 3 in case of 2D scalar fields to 6.
- The curvature computation is done based on immediate neighbors only. Considering larger neighborhood would help in more robust curvature computation.
- High resolution datasets with dense sampling can still yield dense clusters.
- Even with sparse clusters in transformation space, we can devise a better clustering algorithm which finds these sparse clusters. Weights assigned to transformations followed by a density based algorithm may provide better results. One strategy is



(a) Dataset 1 (b) Rotation Space (c) Transformation

(d) Dataset 2 (e) Rotation Space (f) Transformation

Figure 6.20: Figures above show some experimental results for 3D scalar fields. The dataset shown in (a) is generated by combining a 3D Gaussian function and its copy after rotating by 90°about X-axis. The Gaussians are restricted to just one octant so that cluttering of Transformation space can be avoided. (b) shows the clusters in the Rotation transformation. As expected two very dense clusters at 90°and -90°are obtained. (c) shows one of the transformations applied to the volume. The two regions (which are represented by the dense sample points in the figure) are clearly aligned after transformation. The figures (d) – (f) show similar results when the Symmetry detection algorithm is executed for the dataset generated by 60°rotation instead of 90°. Now the clusters in the transformation space are not dense enough, they are spread out as seen in (e).

to use weights inversely proportional to the distance between invariant component of LFDs of the voting points.

- Instead of points voting for transformation, regions can vote for transformation. This may result in more robust transformations.

Chapter 7

Conclusions and Future Work

In this report we have defined the notion of symmetry in scalar fields and proposed a computationally efficient approach for detecting partial and approximate symmetry. This approach overcomes the problems associated with detecting symmetry based on similarity of subtrees of contour trees.

Good results were obtained for 2D scalar fields. However, the algorithm for 3D scalar fields does not produce the desired results. The evidence accumulation stage of the pipeline needs to be redesigned so that it gives dense detectable clusters. We have identified a few causes for the occurence of spread out clusters and proposed some possible solutions which may eliminate these problems.

Currently the pipeline is slow for larger datsets. In future, parallelizing various stages of the symmetry identification pipeline can be done to increase efficiency. Another limitation is that currently only domain translation and rotation transformations are considered. Supporting all the domain and range transformations ($\mathbb{T}$) has unique challenges and it is a much harder problem. The technique of voting and clustering may not work in such a scenario, and a different approach may be required. Another interesting problem would be detecting symmetry in vector fields.

Appendix A

Source code Design

The implementation of the symmetry identification pipeline is done in Java. Object oriented and modular design methodology was followed. The overview of the design is shown in Figure A.1. Only the major classes and modules involved are shown in the figure.

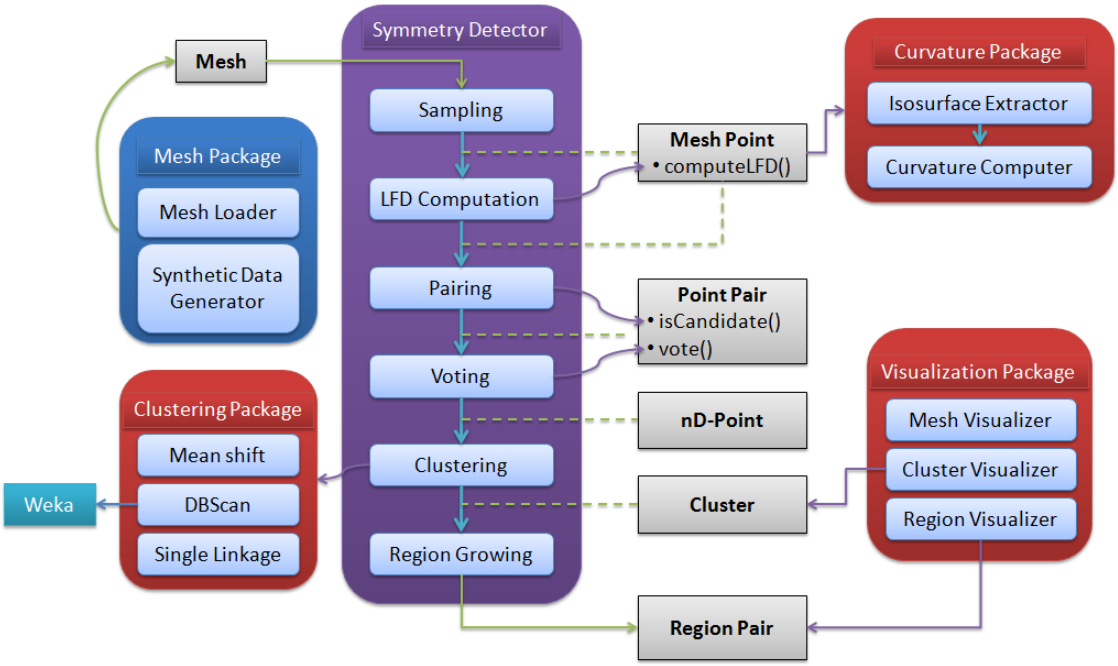


Figure A.1: Overview of the design

Modules Central to the implementation is Symmetry Detector module which is abstraction of symmetry identification pipeline itself. Each stage of the pipeline is implemented as a function in this module. In some cases this module delegates some of the functions to other helper modules.

Some of the helper modules include clustering and curvature computation modules. Clustering module has various clustering algorithm implementations. It also uses external libraries e.g. Weka[1] for some implementations. Curvature computation module has two major sub-routines. First is for extraction of local isosurface at the specified point in the mesh. The other is for computing principal curvatures using the algorithm specified in [1]. These sub-routines are called in sequence to compute curvatures at sample points.

The other modules which although are not used by Symmetry Detector module, but are still important include Mesh Loader and Visualization modules. Mesh Loader is responsible for loading scalar fields from the disk or generating synthetic scalar fields. The Visualization module implements various routines for visualizing the final output as well as intermediate outputs of Symmetry identification pipeline.

Classes Now, let us discuss major classes involved briefly. Mesh class provides the blue-print for scalar fields, each loaded scalar field is represented as Mesh object. Mesh Point class is used to represent sample point on the mesh. It encapsulates the position and LFD of the sample point.

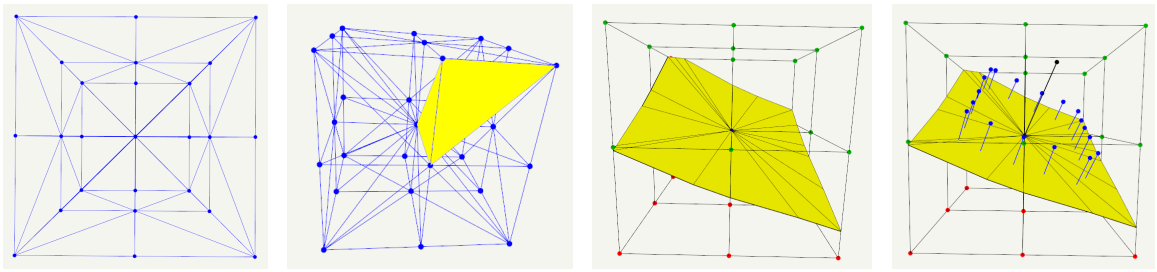
Point Pair class represents a pair of Mesh points. It has sub-routines for determining whether the point pair is candidate for voting and the transformation voting procedure itself. The other important classes include nD-Point class which represents a transformation vote in nD-space, Cluster class for representing clusters in transformation space, and Region Pair class for encapsulating the region growing code and symmetric region pairs extracted from the scalar field.

[1] www.weka.org

Appendix B

Local Isosurface Extraction for 3D grids

Here we discuss the local isosurface extraction procedure for 3D grid points. We are given a grid point p and we want the isosurface passing through p. There are 8 neighboring voxels of p. We decompose each of these voxels into 6 tetrahedra. This decomposition is done such that all the 6 tetrahedra have p as one of the vertices. This results in 48 tetrahedra centered at p. Now, trivial isosurface extraction is done on these tetrahedra using linear interpolation to get the mesh of the isosurface passing through p.



(a) (b) (c) (d)

Figure B.1: (a) The decomposition of neighborhood of p into 48 tetrahedra. (b) One of the 48 tetrahedra is highlighted in yellow. (c) The local isosurface extracted for one of the sample points. (d) The same local isosurface with normals shown pointing towards the gradient direction. All triangles in the extracted mesh are correctly oriented.

Bibliography

[1] P. Alliez, D. Cohen-steiner, O. Devillers, B. Levy, and M. Desbrun, "Anisotropic polygonal remeshing," *ACM Transactions on Graphics*, vol. 3, pp. 485–493, 2003.

[2] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, September 1987.

[3] M. J. Atallah, "On symmetry detection," *IEEE Trans. Computers*, vol. 34, no. 7, pp. 663–666, 1985.

[4] D. Cohen-steiner and J.M-.Morvan, "Restricted delaunay triangulations and normal cycle," in *Proc. of Symposium on Computational Geometry*, 2002, pp. 312–321.

[5] D. Comaniciu and P. Meer, "Mean shift: A robust approach towards feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, May 2002.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining*, 1996.

[7] H.Carr and J.Snoeyink, "Path seeds and flexible isosurfaces using topology for exploratory visualization," in *In Proc. Symposium on Data Visualization*, 2003, pp. 49–58.

[8] H.Carr, J.Snoeyink, and M. de Panne, "Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree," *Computational Geometry*, vol. 43, no. 1, pp. 42–58, 2010.

[9] Y. Hong and H.-W. Shen, "Parallel reflective transformation for volume data," *Computer and Graphics*, vol. 32, no. 1, pp. 41–54, 2008.

[10] J. Kerber, M. Bokeloh, M.Wand, J. Krüger, and H.-P. Seidel, "Feature preserving sketching of volume data," in *Vision, Modeling, and Visualization*, R. Koch, A. Kolb, and C. Rezk-Salama, Eds., 2010.

[11] J. Kerber, M. Wand, J. Krüger, and H.-P. Seidel, "Partial symmetry detection in volume data," in *Vision, Modeling, and Visualization*, P. Eisert, K. Polthier, and J. Hornegger, Eds., 2011.

[12] G. Loy and J.-O. Eklundh, "Detecting symmetry and symmetric constellation of features," in *Proc. European Conference on Computer Vision*, 2006, pp. 508–521.

[13] P. Minovic, S. Ishikawa, and K. Kato, "Symmtery identification of a 3d object represented by octree," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 507–514, 1993.

[14] N. Mitra, L. J. Guibas, and M. Pauly, "Partial and approximate symmetry detection for 3d geometry," *ACM Transactions on Graphics*, vol. 25, pp. 560–568, 2006.

[15] ——, "Symmetrization," *ACM Transactions on Graphics*, vol. 26, no. 3, p. 63, 2007.

[16] M. Pauly, N. Mitra, J. Wallner, H. Pottman, and L. J. Guibas, "Discovering structural regularity in 3d geometry," *ACM Transactions on Graphics*, vol. 27, no. 3, 2008.

[17] P.D.Simari, E. Kalogerakis, and K.Singh, "Folding meshes: hierarchical mesh segmentation based on planar symmetry," in *In Proc. Symposium on Geometry Processing*, 2006, pp. 111–119.

[18] J. Podolak, A. Golovinskiy, and S. Rusinkiewicz, "Symmetry-enhanced remeshing of surfaces," in *In Proc. Symposium on Geometry Processing*, 2007, pp. 235–242.

[19] D. M. Thomas and V. Natarajan, "Symmetry in scalar field topology," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2035–2044, 2011.