Bridging the Gap Between Local and Global Approaches for 3D Object Recognition

Isma Hadji
G. N. DeSouza
Outline

Introduction

Motivation

Proposed Methods:
1. LEFT keypoint Detector
2. LGS Feature Descriptor

The LEFT-LGS combination

Conclusion and Future Work
Introduction

Object recognition algorithms, usually, rely on three main steps:

- **Keypoint Detection**
  - Distinctiveness
  - Repeatablity

- **Feature Description**
  - Discrimination
  - Robustness to:
    - Occlusion
    - Clutter

- **Object Recognition**
  - Simplicity
  - Pose Estimation Capability

- Descriptors play the most critical step in the Object Recognition Pipeline
- The techniques adopted for object recognition can be divided into two main categories: **Global versus Local**
The techniques adopted for 3D object description can be divided into two main categories:

**Global Descriptors**
- Scene Segmentation
  - e.g: EDC
- Keypoint Detection
  - e.g: LEFT
- Feature Description
  - e.g: GFPFH
- Feature Matching
  - e.g: 1-NN
- Object Recognition
  - e.g: CRH
- Pose Estimation

**Local Descriptors**
- Keypoint Detection
  - e.g: LEFT
- Feature Description
  - e.g: LGS
- Feature Matching
  - e.g: 1-NN
- Object Recognition
  - e.g: CG Algorithms
- Pose Estimation
  - e.g: RANSAC
Local Approaches

- Keypoint Detection: e.g. LEFT
- Feature Description: e.g. LGS
- Feature Matching: e.g. NN
- Object Recognition: e.g. CG
- Pose Estimation: e.g. RANSAC
Global Approaches

- **Scene Segmentation**
  - e.g: EDC

- **Keypoint Detection**
  - e.g: LEFT

- **Feature Description**
  - e.g: GFPFH

- **Feature Matching**
  - e.g: NN

- **Object Recognition**
  - e.g: CRH

- **Pose Estimation**
Local Versus Global

Local Approaches

- **PROS?**
  1. No segmentation needed.
  2. **Local descriptors** proved to be a more attractive within cluttered/occluded scenes.
  3. **Straightforward** pose estimation.

- **CONS?**
  1. less **discriminating** due to the limited scope of the local neighborhood.
  2. **Additional** step required for bad correspondence rejection.

Global Approaches

- **PROS?**
  1. **Global descriptors** are usually more discriminating as they encode the entire structure of objects.
  2. **Simple** Nearest Neighbor matching is enough to find correspondences.

- **CONS?**
  1. Requires segmentation for scene
  2. Very much affected by occlusions and clutter.
  3. Requires complicated algorithms for estimating poses.
The Local-to-Global Approach

Solution?

The Local-to-Global approach

Why Local?

Local in the sense that each feature vector describes a single keypoint, rather than the entire object ➔ Follows the local object recognition pipeline!

Why Global?

Global as it looks beyond the local neighborhood (support regions) to detect and describe object ➔ Global detection and description
There are two main motivations for finding keypoints:

1. We want to reduce the computational overhead.
2. We only want to rely on distinctive parts of an object to recognize it.

The main challenge when selecting keypoints is to maximize:

1. Distinctiveness.
2. Repeatability.
HOW?

First estimate some low level features describing each point in the cloud.

Traditionally keypoint detectors rely on local threshold-based algorithms.

In this research, we question both:

1. The local characteristic of traditional 3D keypoint detectors.
2. The threshold based approaches.
The proposed keypoint detection scheme was inspired on the way animals detect “important information”.

The least frequent a specific shape is, the more representative it is of the object!

1. Low level features describing a 3D cloud of points or mesh are estimated for each point $p_i$ in the object.
2. Most similar features are grouped together → Quantizing the feature space.
3. The histogram built in the previous step is sorted in ascending order → The features are sorted from least to most frequent.
4. Points with least frequent feature occurrence in the object are selected as keypoints!
In order to evaluate the repeatability of the keypoints detected using LEFT we performed experiments using 4 benchmark datasets.

a) The Retrieval Dataset

b) The Stereo Dataset

We also compare our results to 3 SOTA detectors: ISS, LSP and KPQ
A good keypoint detector is characterized by a high repeatability.

The steps followed for measuring repeatability are as follows:

1. Find keypoints on both scene and model.
2. Rotate and translate the model based on the ground truth transformation.
3. Find correspondences between detected keypoints in the model and in the scene.
4. Calculate the relative repeatability as the number of correspondences divided by the total number of unoccluded keypoints detected in the model.
LEFT Performance

- LEFT repeatability for variable number of keypoints.
Feature Description

- The role of the descriptors is to encode the geometry of the object.
- In the case of feature descriptors the main challenges are:
  1. Maximizing the discrimination power of the descriptors.
  2. Robustness to occlusion and clutter.
LGS Descriptor - Motivation

(a) Locally similar keypoints from two different objects

(b) SHOT feature vector

(c) LGS feature vector
LGS Descriptor - Motivation

Dragon

Armadillo

Partial view of Armadillo

GFPFH Feature Vectors
More specifically, the LGS was built around the following five main ideas:

1. Relying on surface point classification to capture the entire geometry of the object (global property).

2. Describing keypoints (local property), but using both local and global support regions grouped by the same surface class.

3. Using signatures to avoid loss of information and mitigate the effects of occlusion (local property).

4. Using distributions of L2-distances to encode the relationship between keypoint and global support regions while increasing robustness to noise by eliminating the use of sensitive features such as surface normal (global property).

5. Using confidence on the relationship above to improve the matching during object recognition (local property).
Step 1 ➔ Point Classification

To classify the points we use the radius-based surface classification proposed in the RSD descriptor.

The original RSD used both the minimum and maximum radii to classify points as belonging to one of the geometric primitive shapes.

In our implementation we chose to classify points from very sharp to very smooth.

Why?

- The ability to find a pre-defined number of classes independently of the object considered – i.e. we can vary the number of classes by simply varying the ranges of sharpness and smoothness.

- Using the minimal radii of each point on the object’s surface, the algorithm splits the radii values into N different ranges, representing the N classes of the object’s surfaces.
Step 2 ➔ Class Confidence Estimation

LGS uses continuous ranges of radii values, therefore fuzzy regions may emerge.

Fuzzy regions contain points that could belong to any of the two consecutive ranges ➔ Unstable Points

Solution?

We assign both a class and a membership to the class for each point in the cloud.

\[
c = \frac{\text{# of points in class } n \text{ in the neighborhood of } p}{\text{Total number of points in the neighborhood of } p}
\]  

A low confidence indicates that the point belongs to a fuzzy or noisy region. This determines when the algorithm gives high or low weights to the points used in the next steps.
Step 3 ➔ Signature Construction

The LGS descriptor is constructed in a signature-based fashion using:

- K-neighbors in each one of the N classes (e.g. N=3).
- Sorted in D clusters
- L2 distances and Confidences Concatenated ➔ D*N Feature Vector
Step 3 ➔ Signature Construction

Another example:
- N=5 classes.
Step 4 ➔ LGS Descriptor Matching

The confidences estimated in step 2 are used as weights during the matching stage, when the LGS computes the distance $d_{ij}$ between a pair of signatures $(i, j)$.

The distance between each entry of the pair of signatures is multiplied by the corresponding minimum confidence.

This allows LGS to reduce the effect of unstable points located on fuzzy regions.

Mathematically, the weighted distance used to compare LGS descriptors is given by:

$$d_{ij} = \sqrt{\sum_{l=1}^{(D*N)} \min(c_{li}, c_{lj}) \times (f_{li} - f_{lj})^2}$$

(2)
In order to evaluate the robustness of the LGS descriptor we performed experiments using the previous datasets.

We also compare our results to SOTA descriptors. 3 local descriptors: (SI, FPFH, SHOT) and 2 Global ones (VFH, GFPFH).

A good descriptor is characterized by a high percentage of correct matches.

1. Find and describe all keypoints in both the scene and the model.
2. For each feature vector in the scene find its nearest neighbor in the model.
3. Find the true correspondences using the provided ground-truth transformation between model and scene.
4. Compare the correspondences from step 2 and step 3 and count total number of true/false matches.
Effect of the number of classes $N$ used to construct the signature

- As expected:
  1. A small number of classes does not capture enough of the variations in the structure of these objects.
  2. Too many classes also leads to reduced discriminating power.
- Why?
  - Possible miss-classification error
Classification Accuracy

Again, as expected adding more classes caused a drop in the classification accuracy, which ultimately caused a drop in the number of good correspondences.
Effect of the number of neighbors in each class

- Interestingly enough, we can see that the LGS descriptor is not too sensitive to the use of larger neighborhoods.
- We attribute this to the advantages of the use of a signature-based feature vector.
LGS Performance

Quantitative Comparison of LGS with respect to Local SOTA 3D descriptors
LEFT-LGS Performance

Quantitative Comparison of LEFT-LGS with respect to Local SOTA 3D descriptors
A good Object Recognition algorithm should have a high TPR and a small FPR.

1. Find keypoints and their corresponding descriptors for both scene and models.
2. Find Potential matches using the a simple Nearest Neighbor matching.
3. Prune correspondences using a Correspondence Grouping algorithm!
4. Count the number of “consistent” correspondences to decide upon model presence or absence.
LEFT-LGS Compared to Local Descriptors

- LEFT-LGS combined in the object recognition pipeline compared to Local SOTA 3D descriptors

[Graphs showing performance comparisons]
We can compare the performance of LEFT-LGS to global descriptors directly in terms of correctly recognized objects.

1. Segment the scene and describe each segment.
2. For each segment descriptor find its closest model descriptor.
3. Confirm model presence or absence according to the ground truth scene annotation.
LEFT-LGS Compared to Global Descriptors

LEFT-LGS combined in the object recognition pipeline compared to Global SOTA 3D descriptors.

<table>
<thead>
<tr>
<th>Retrieval Dataset</th>
<th>noise=0.1</th>
<th>noise=0.3</th>
<th>noise=0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEFT-LGS</td>
<td>81.52%</td>
<td>79.07%</td>
<td>77.07%</td>
</tr>
<tr>
<td>VFH</td>
<td>29.89%</td>
<td>28.44%</td>
<td>28.33%</td>
</tr>
<tr>
<td>GFPFH</td>
<td>87.04%</td>
<td>86.52%</td>
<td>84.22%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Random Views Dataset</th>
<th>noise=0.1</th>
<th>noise=0.3</th>
<th>noise=0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEFT-LGS</td>
<td>55.56%</td>
<td>54.44%</td>
<td>54.44%</td>
</tr>
<tr>
<td>VFH</td>
<td>15%</td>
<td>13.33%</td>
<td>13.33%</td>
</tr>
<tr>
<td>GFPFH</td>
<td>13.06%</td>
<td>13.06%</td>
<td>11.94%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stereo Dataset</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LEFT-LGS</td>
<td>56.67%</td>
</tr>
<tr>
<td>VFH</td>
<td>8.89%</td>
</tr>
<tr>
<td>GFPFH</td>
<td>3.78%</td>
</tr>
</tbody>
</table>
Conclusion and Future Work

This work?

- Built a bridge between Local and Global approaches used the field of Object Recognition.
- Introduced the concept of detecting keypoints globally ➔ LEFT
- Proposed a hybrid descriptor using the advantages of local and global approaches while minimizing the shortcomings of each ➔ LGS

Future Work?

- Make the detection stage completely independent of the segmentation step.
- Address the problem of surface misclassification.
- Devise a method for automatic selection of the number of classes N.
Thank you!

QUESTIONS?