Neurocomputing 150 (2015) 126–135

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Efficient pose estimation of rotationally symmetric objects

Rui Pimentel de Figueiredo *, Plinio Moreno, Alexandre Bernardino

Instituto Superior Técnico, Universidade de Lisboa, Portugal

ARTICLE INFO

Article history:
Received 1 February 2014
Received in revised form
22 July 2014
Accepted 29 July 2014
Available online 15 October 2014

Keywords:
3D pose estimation
3D object recognition
Symmetry
Voting
Real-time

ABSTRACT

In this paper we propose algorithms for 3D object recognition from 3D point clouds of rotationally symmetric objects. We base our work in a recent method that represents objects using a hash table of shape features, which allows to match efficiently features that vote for object pose hypotheses. In the case of symmetric objects, the rotation angle about the axis of symmetry does not provide any information, so the hash table contains redundant information. We propose a way to remove redundant features by adding a weight factor for each set of symmetric features. The removal procedure leads to significant computational savings both in storage and time while keeping the recognition performance. We analyze the theoretical storage gains and compare them against the practical ones. We also compare the execution time gains in feature matching and pose clustering. The experiments show storage gains up to 100× and execution time savings up to 3500× with respect to state-of-the-art methods.

1. Introduction

3-D object recognition plays a role of major importance in the robotics field. Many applications, such as object grasping and manipulation, critically depend on visual perception algorithms. These must be robust to cluttered environments and to sensor noise, as well as fast enough for real-time operation, in order for the robot to correctly interact with the surrounding environment. During the last few decades several methods have been proposed to solve the object recognition problem, but it is still a very challenging task and many research efforts continue to be made. Due to recent technological advances in the field of 3-D sensing, range sensors provide 3-D points with reasonable quality and high sampling rates, sufficient for efficient shape-based object recognition. In recent work, Drost et al. [1] proposed an approach which extracts description from a given object model, using point pair features, encoding the geometric relation between oriented point pairs. The matching process is done locally using an efficient voting scheme (see Fig. 3) similar to the Generalized Hough Transform (GHT) [2]. Their method is robust to sensor noise and outperforms other feature-based state-of-the-art methods like Spin Images [3] and Tensors [4], both in terms of computational speed and in terms of robustness to occlusion and clutter.

In this paper we introduce an important extension to [1] for dealing efficiently with rotationally symmetric objects, which are common in many daily tasks (e.g. kitchenware objects like cups, glasses, cans, plates), showing improvements both in memory storage and in processing speed with respect to [1].

We also extend the previous approach with a set of rules for re-sampling the point clouds that prevent aliasing effects related to feature space discretization, and an extensive analytical analysis of the proposed contributions.

Next section addresses the related work, followed by the description of Drost et al. object recognition and pose estimation algorithm. Then, in Section 4 we propose a methodology to efficiently deal with rotational symmetries. In Section 5 we propose an automatic sampling selection criteria to avoid loss of information caused by sampling aliasing in the feature space. Lastly, in Section 6 we show results that validate our approach.

2. Related work

Symmetry in objects has been extensively studied due to its application to areas such as object recognition [5], shape matching [6], object model compression [7], geometric hashing [8] and pose detection [9]. The common idea across all these areas is that symmetry allows to define invariant models/features to geometric transformations, which are gathered in the symmetry group [10].

Regarding shape matching, symmetry descriptors (i.e. invariant features) are able to match parts of 3D objects that hold the symmetry constraints (i.e. partial symmetry). Kazhdan et al. [6] introduced a collection of spherical functions that detect symmetry and retrieve objects. The main constraint of this work is the availability of full models in order to retrieve robustly the objects.

Regarding model compression, detection of symmetry in 3D models provides the parts of the models that can be compressed. Martinet et al. [7] demonstrate experimentally the storage gains of the models when symmetries are detected in the objects.

* Corresponding author.

http://dx.doi.org/10.1016/j.neucom.2014.07.070
0925-2312/

Regarding geometric hashing, the parameters of symmetric transformations work as some of the indexes of the hash table. Geometric hashing was initially formulated on a 2D problem [8], and then extended to 3D object matching [11] and Bayesian hashing [12]. Wolfson and Rigoustsos [13] propose to reduce the size of the hash table by collapsing the redundant features, which in the case of geometric hashing could lead to storage savings by a factor of two.

In a recent work Thomas [9] reduces the size of the hash table for pose detection, exploiting symmetries. Each object is represented on a hash table that indexes features computed on point pairs and triplet pairs. The storage size of the hash table is reduced by considering rotational symmetries.

In this work we follow the idea of Wolfson and Rigoustsos [13], which is also applied by Thomas [9]. Similar to Thomas [9], we build a hash table indexed by point pairs. We apply the hash table reduction of [13] to the method of [1] and, in contrast to the previous works, we analyze the theoretical computational storage savings and confirm these results with experiments on real data.

3. Method overview

The basic units to describe surface shape are surflets [14] $\mathbf{s}=(\mathbf{p},\mathbf{n})$, where $\mathbf{p}$ represents sample points in the surface and $\mathbf{n}$ are the associated surface normals. Let M be the set of all model surflets, $M=\{\mathbf{s}_i^m,\ i=1..N\}$ and let S be the set of all scene surflets, $S=\{\mathbf{s}_i^s,\ i=1\ldots N\}$.

The recognition process consists in matching scene surflet pairs $(\mathbf{s}_r^s,\mathbf{s}_i^s)$ to model surflet pairs $(\mathbf{s}_r^m,\mathbf{s}_i^m)$. $\mathbf{s}_r$ and $\mathbf{s}_i$ being two surflets, the Point Pair Feature (PPF) $\mathbf{F}\in F\subset\mathbb{R}^4$ is defined as a 4-tuple composed by the distance between the reference, $\mathbf{p}_r$, and secondary, $\mathbf{p}_i$, points; the angle between the normal of the reference point $\mathbf{n}_r$ and the vector $\mathbf{d}=|\mathbf{p}_i-\mathbf{p}_r|$; the angle between the normal of the secondary point $\mathbf{n}_i$ and $\mathbf{d}$; and the angle between $\mathbf{n}_r$ and $\mathbf{n}_i$ as illustrated in Fig. 1. This could be formally described by

$$\mathbf{F}=\mathrm{PPF}(\mathbf{s}_r,\mathbf{s}_i)=(f_1,f_2,f_3,f_4)=(\|\mathbf{d}\|,\angle(\mathbf{n}_r,\mathbf{d}),\angle(\mathbf{n}_i,\mathbf{d}),\angle(\mathbf{n}_r,\mathbf{n}_i)) \quad (1)$$

The data structure chosen to represent the model description was a hash table, for fast lookup during the matching phase, in which the key value is given by the discrete PPF while the mapped value is the respective surflet pair. In a more rigorous sense, the hash table is a multi-valued hash table since one key could be associated with several surflet pairs. Thus, each slot of the hash table contains a list of surflet pairs with a similar discrete feature. The hash function samples the 4-tuple PPF and converts the resulting discrete feature to a single integer I which serves as an index of the hash table. This conversion is given by the multi-dimensional array addressing formula:

$$\begin{aligned} I &= \mathrm{hash}(\mathbf{F},n_\alpha)\\ &= f_1^{\mathrm{bin}}+f_3^{\mathrm{bin}}\cdot n_{\mathrm{d}}\cdot n_\alpha+f_4^{\mathrm{bin}}\cdot n_{\mathrm{d}}\cdot n_\alpha\cdot n_\alpha \end{aligned} \quad (2)$$

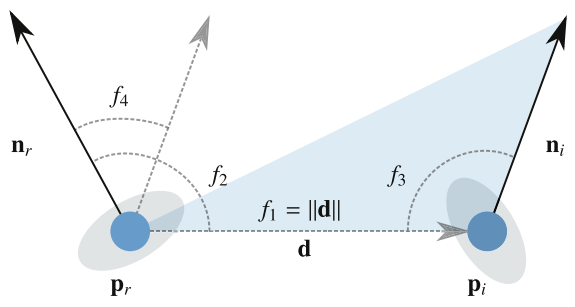

Fig. 1. Point pair feature.

The hash function is thus a mapping from the PPF space F to the model description A, which can be formally expressed by the following:

$$\mathrm{hash}: F\subset\mathbb{R}^4\rightarrow A\subset M^2 \quad (3)$$

3.1. Pose estimation

A set of reference surflets on the scene $R_s\subset S$ is uniformly sampled from S and each of them is paired with all the other surflets on the scene. The number of reference points is given by $|R_s|=\xi|S|$ where $\xi\in[0;1]$ is the reference points sampling ratio control parameter. For each scene surflet pair $(\mathbf{s}_r^s,\mathbf{s}_i^s)\in S^2$, $\mathrm{PPF}(\mathbf{s}_r^s,\mathbf{s}_i^s)$ is computed and set of similar model surflet pairs is retrieved from the hash table. From every match between a scene surflet pair $(\mathbf{s}_r^s,\mathbf{s}_i^s)\in S^2$ and a model surflet pair $(\mathbf{s}_r^m,\mathbf{s}_i^m)\in M^2$, one is able to compute the rigid transformation that aligns the matched model with the scene. This is done first by computing the transformations $\mathbf{T}_{m\rightarrow g}$ and $\mathbf{T}_{s\rightarrow g}$ that align $\mathbf{s}_r^m$ and $\mathbf{s}_r^s$, respectively, to the object reference coordinate frame x-axis, and secondly by computing the rotation α around the x-axis that aligns $\mathbf{p}_i^m$ with $\mathbf{p}_i^s$. The transformation that aligns the model with the scene is then computed considering the ensuing expression:

$$\mathbf{T}_{m\rightarrow s}=\mathbf{T}_{s\rightarrow g}^{-1}\mathbf{R}(\alpha)\mathbf{T}_{m\rightarrow g} \quad (4)$$

In detail, the transformations $\mathbf{T}_{m\rightarrow g}$ and $\mathbf{T}_{s\rightarrow g}$ translate $\mathbf{p}_r^m$ and $\mathbf{p}_r^s$, respectively, to the reference coordinate frame origin and rotates their normals $\mathbf{n}_r^m$ and $\mathbf{n}_r^s$ onto the x-axis. After applying these two transformations, $\mathbf{p}_i^m$ and $\mathbf{p}_i^s$ are still misaligned. The transformation $\mathbf{R}(\alpha)$ applies the final rotation needed to align these two points. The previous reasoning is depicted in Fig. 2.

The transformation expressed in Eq. (4) can be parametrized by a surflet on the model and a rotation angle α. In [1], this pair $(\mathbf{s}_r^m,\alpha)$ is mentioned as the *local coordinates* of the model with respect to reference point $\mathbf{s}_r^s$.

3.1.1. Voting scheme

This method uses a voting scheme similar to the Generalized Hough Transform (GHT) for pose estimation. For each scene reference surflet, a two-dimensional accumulator array that represents the discrete space of local coordinates is created. The number of rows, N_m, is the same as the number of model sample surflets $|M|$, and the number of columns N_α is equal to the number of sample steps of the rotation angle α. A vote is placed in the accumulator array by incrementing the position associated to the local coordinates $(\mathbf{s}_r^m,\alpha)$, by 1 (see Fig. 3). After pairing $\mathbf{s}_r^s$ with all $\mathbf{s}_i^s$, the highest peak – i.e. the position with most votes – in the accumulator – corresponds to the optimal local coordinate.

In the end, all retrieved pose hypotheses whose position and orientation do not differ more than a predefined threshold are clustered together.

3.1.2. Pose clustering

As shown in the pseudo-code of Algorithm 1, the voting process is done for several reference surflets on the scene to ensure that at least one of them lies on the object we want to detect. For each reference surflet, a set of pose hypotheses – each corresponding to a peak in the accumulator array and weighted by its associated number of votes – is generated. After the voting procedure, we consider an additional pose clustering step that brings stability and increases the accuracy of the final result since the retrieved poses only approximate the ground truth due to sensor noise and sampling artifacts. The pose clustering procedure (see Algorithm 1) outlined in this subsection is a clustering algorithm that aggregates all pose hypotheses that do not differ