Pattern Recognition Letters 33 (2012) 2127–2133

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

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

Hierarchical Normal Space Sampling to speed up point cloud coarse matching

Yago Diez *, Joan Martí, Joaquim Salvi

Edifici P-4, Depart. d'Arquitectura i Tecnologia de Computadors, Campus Montilivi, Universitat de Girona, 17071 Girona, Spain

ARTICLE INFO

Article history:
Received 28 November 2011
Available online 20 July 2012

Communicated by S. Sarkar

Keywords:
Coarse point cloud matching
Normal space sampling
Hierarchical algorithms
Data Structures

ABSTRACT

Point cloud matching is a central problem in Object Modeling with applications in Computer Vision and Computer Graphics. Although the problem is well studied in the case when an initial estimate of the relative pose is known (fine matching), the problem becomes much more difficult when this a priori knowledge is not available (coarse matching). In this paper we introduce a novel technique to speed up coarse matching algorithms for point clouds. This new technique, called Hierarchical Normal Space Sampling (HNSS), extends Normal Space Sampling by grouping points hierarchically according to the distribution of their normal vectors. This hierarchy guides the search for corresponding points while staying free of user intervention. This permits to navigate through the huge search space taking advantage of geometric information and to stop when a sufficiently good initial pose is found. This initial pose can then be used as the starting point for any fine matching algorithm. Hierarchical Normal Space Sampling is adaptable to different searching strategies and shape descriptors. To illustrate HNSS, we present experiments using both synthetic and real data that show the computational complexity of the problem, the computation time reduction obtained by HNSS and the application potentials in combination with ICP.

1. Introduction

Laser Scanners or range finders are widely used in industrial applications such as reverse engineering, mould fabrication and quality control. Point clouds are one of the most frequent outputs of laser scanners as they capture the shape of objects with high precision, can be used as the starting point of more complex object representations (planar patches and union of surfaces of superior order) and are used as the input of many object modeling algorithms (object alignment (Mian et al., 2006), merging of views (Matabosch et al., 2008; Wei et al., 2010) and manufactured piece-model comparison (Salvi et al., 2008)). The goal of point cloud matching is to find the Euclidean motion(s) between two (or more) range images of a given object in order to represent these images in the same coordinate system. Following the classification provided in Salvi et al. (2007), we divide matching methods in two groups: *fine* and *coarse* matching.

In fine matching, an initial alignment is required (Xie et al., 2010) and the goal is to converge to the most accurate matching possible. This is achieved by iteratively minimizing the distances between temporal correspondences (also known as closest points). Important issues in all these algorithms include the use of efficient nearest-neighbour search structures, the matching strategy, the distance considered and the robustness in terms of noise and false correspondences. ICP (Iterative Closest Point), introduced by Besl and McKay (Besl and McKay, 1992) experienced several improvements dealing with matters such as robustness (Trucco et al., 1999), algorithm speed up (Zinsser et al., 2003) and the use of hierarchy in data (Jost and Hügli, 2002; Turk and Levoy, 1994). Overall, ICP is the most common fine matching method and has been shown to get very good results. However, ICP usually presents problems of convergence, is sensitive to noise in data and might require expensive running times. Also, in some cases it converges to local minima and fails to provide the desired solution.

On the other hand, coarse matching can compute an initial estimate of the motion between two point clouds that are quite apart. The main difference with fine matching is that coarse matching is generally non-iterative in the sense that it does not converge to the best possible solution. Coarse matching is usually the first step to a posterior fine matching. A popular approach is to compute motion-invariant features (or descriptors) for both sets in order to further describe the objects and speed up the search for matching. Coarse matching algorithms differ on shape feature descriptors (for example Point Signature (Chua, 1997) or Spin Image (Johnson, 1997)) and on the matching methods used to stablish correspondences between objects (Genetic Algorithms (Brunnström and Stoddart, 1996), Principal Component Analysis "PCA" (Chung and Lee, 1998) or RANdom Sample And Consensus "RANSAC" (Fischler and Bolles, 1981)). Coarse matching suffers from high computational costs, needs to avoid local minima and its performance drops when data is noisy. PCA, additionally, cannot be used if the percentage of overlap between the two sets is low, although it is

* Corresponding author. Tel.: +34 972418013/8494/8483; fax: +34 972418976.
E-mail addresses: yago@eia.udg.edu (Y. Diez), joanm@eia.udg.edu (J. Martí), qsalvi@eia.udg.edu (J. Salvi).

0167-8655/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2012.07.006

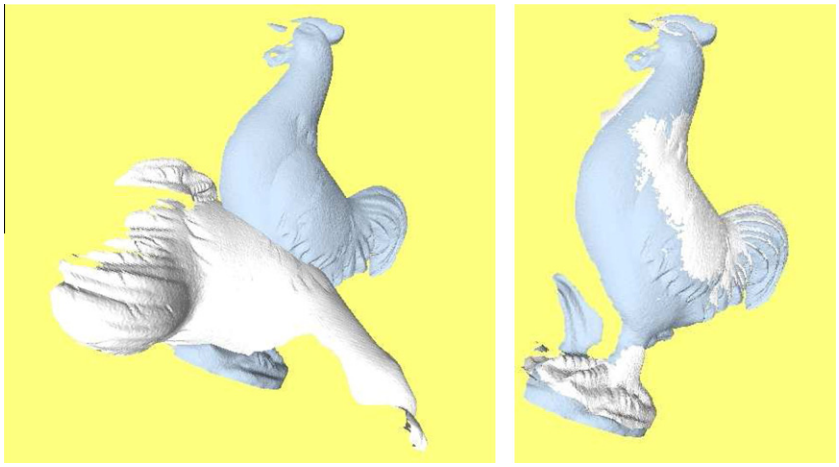

Fig. 1. Matching sample, initial position (left), output estimated pose (right). Point clouds are presented with a superimposed texture for better visualization.

sometimes regarded as the standard method when noise in data is low and the overlap is high.

In order to address these problems, we present a speed-up technique for coarse matching algorithms and apply it in the RANSAC approach. RANSAC was first proposed by Fischler and Bolles (Fischler and Bolles, 1981). The problem that RANSAC solves is to estimate the parameters of a certain model starting from a set of data that contains an "outlier" subset that, if used to determine the model, does not lead to a correct solution. RANSAC has been extensively used both in fine (Masuda and Yokoya, 1995) and coarse (Aiger et al., 2008) matching.

Let $\mathcal{A} = \{a_1, \ldots, a_n\}$ and $\mathcal{B} = \{b_1, \ldots, b_m\}$ be two sets of points with $a_i, b_j \in \mathbb{R}^3$. Our goal is to determine a rigid transformation that brings $\mathcal{B}$ closer than a given threshold to $\mathcal{A}$ (see Fig. 1 for an example). The matching threshold, expressed in terms of Sum of Squared Differences (SSD), is set beforehand and must be sufficient for the subsequent fine matching to succeed (see details in Section 3). In order to determine the rigid motion we need to find at least two corresponding triplets $((a_{i1}, a_{i2}, a_{i3}), (b_{j1}, b_{j2}, b_{j3})$ with $a_{ik} \in \mathcal{A},\ b_{jl} \in \mathcal{B})$. The rigid motion is then computed by SSD minimization between both triplets using singular value decomposition (Sorkine, 2007). In some cases, more than three points per set are used (Aiger et al., 2008).

RANSAC applied to point cloud matching consists in finding these corresponding triplets taking into account that some "outlier" points do not lead to a solution. RANSAC samples randomly the motion space. Enumerating all possible motions stands for choosing the two corresponding triplets needed to determine each of them. This represents a computational cost of O (n^3m^3) that, although mitigated by the random sampling of RANSAC, may easily be unusable in practice. A key observation is that coarse matching does not really need to traverse all this huge search space. Instead, the goal is to search for a sufficiently good solution (not necessarily the best) and to stop the algorithm as soon as such a solution is found.

Existing speed-up solutions for RANSAC-based coarse matching include the use of descriptors (Chua, 1997; Johnson, 1997) and Normal Space Sampling (NSS) (Rusinkiewicz and Levoy, 2001). Descriptors prune the search space by eliminating non-related couples in terms of shape, while the goal of NSS is to scale down the problem by using vectors locally normal to the surface described by the point cloud. NSS selects more points in those regions with less frequent normals and downsamples regions with uniform normals. However, should the sampling include too few points, it might prevent the finding of a correct matching. Additionally, NSS does not interact with the subsequent search, so the question: "how many points to sample?" becomes both determinant for the result and very difficult to answer without human intervention.

We address all these problems in detail in Section 2. Section 3 contains the experimental results obtained using both synthetic data and real data. The paper ends with the conclusions in Section 4.

2. HNSS for Coarse matching

In this section we present the Hierarchical Normal Space Sampling (HNSS), a new and more efficient organization of data. HNSS organizes data in a hierarchy (Fig. 2, right) of sets of increasing cardinality taking into account the surface normal vectors at every surface point. HNSS can be used in combination with any RAN-

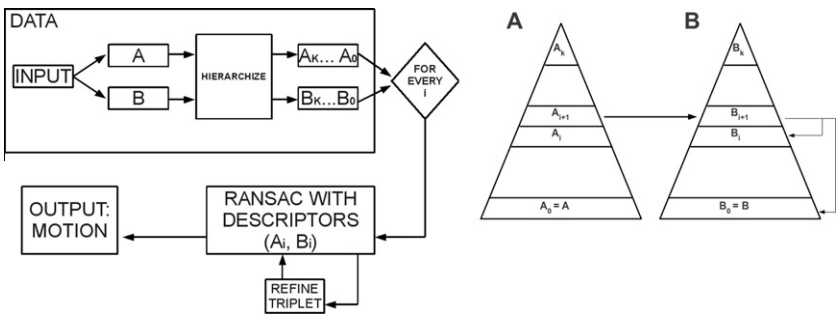

Fig. 2. HNSS Overview: HNSS schema (left), and data hierarchy (right).

Table 1
Time reduction due to use of data hierarchy.

SET	RANSAC				HNSS 4 levels			
	$\#IT$	$t_{res}(s)$	$t_{total}(s)$	Last level	$\#IT$	$t_{res}(s)$	$t_{total}(s)$	Last level
T100	390	$\ll 0.01$	0.01	0	123	$\ll 0.01$	0.01	2
C1000	12117.5	0.02	1.3	0	373.5	0.01	0.09	1.1
T1000	313005	1.67	33.65	0	2652	0.01	0.27	1.5
C5000	1079520.8	1.68	460.2	0	1850.31	0.14	0.44	1.9
T5000	66477531.8	420.9	27640	0	3312.9	0.13	0.54	2.1
	HNSS 7 levels				HNSS 10 levels			
	$\#IT$	$t_{res}(s)$	$t_{total}(s)$	Last level	$\#IT$	$t_{res}(s)$	$t_{total}(s)$	Last level
T100	403.8	$\ll 0.01$	0.01	4	467.2	$\ll 0.01$	0.01	7.0
C1000	132.8	$\ll 0.01$	0.03	2.9	1046	0.01	0.88	6.2
T1000	1178	0.02	0.12	3.3	20003.4	0.04	0.75	4.8
C5000	937.1	0.08	0.3	4.2	42663.1	0.89	1.27	5.2
T5000	487.4	0.07	0.14	5.1	187438.6	0.74	5.77	6.6

SAC-based coarse matching algorithm, possibly using descriptors, such as (Aiger et al., 2008). In order to focus on the improvements given by HNSS and not on the advantages of a chosen descriptor or search strategy, we present results of HNSS with a RANSAC schema without descriptors.

HNSS improves the search for matching: First, the use of descriptors and HNSS allows the identification of search space areas that are more likely to produce better results. Second, HNSS is able to refine partial results, so it can dig "deeper" in those parts of the search space where interesting results are found. HNSS is also a way to make use of NSS in a fully automated way that does not require decision on how many points to sample. Bearing in mind that the goal is not to find the best possible matching, but to provide a "good enough" initial guess for fine matching to start, we claim that exploring the (huge) search space with HNSS is discriminant, very adaptable and more efficient. Section 3 experimentally supports these claims.

Following NSS, we classify the surface points according to their normals, but we sort and group them in levels according to singularity. For example, the topmost level contains the points with more singular normals. The lowermost level of the hierarchy contains all the points, but the goal of HNSS is to direct the search through the higher levels that contain less points (See Table 1, Section 3.1 for details). The idea of organizing data hierarchically has been widely used in combination with ICP (Jost and Hügli, 2002; Turk and Levoy, 1994). However, this is, to the best of our knowledge, the first time it is used in combination with NSS.

The data hierarchy is traversed using a 3-point RANSAC algorithm for every hierarchy level (Fig. 2, left). This corresponds to a search for matching amongst points with normals that are similar in terms of singularity. Potential correspondences of triplets of points in $\mathcal{A}$ to triplets of points in $\mathcal{B}$ are checked. Every time a correspondence is determined, the corresponding rigid motion in terms of SSD is computed (Hoppe et al., 1992; Sorkine, 2007) and, if it is considered good enough the search stops. The hierarchical organization of data makes it possible to explore first the points in set $\mathcal{A}$ that are more promising in terms of their normals. We can say thus, that the hierarchy of set $\mathcal{A}$ guides the search for promising triplets of points $a_{i_1}, a_{i_2}, a_{i_3} \in \mathcal{A}$. Promising triplets are those that induce at least one motion between the sets. On the other hand, when a promising triplet of points is found, the algorithm may explore subsequent levels in the hierarchy in order to refine the matching. Consequently, the hierarchy of set $\mathcal{B}$ allows the algorithm to find the best correspondence for promising triplets and, thus, spend more computing time in the areas where interesting results are being found.

Concerning theoretical complexity, the total cost of the classical RANSAC algorithm can be shown to be $O(n^3 m^4 \log n)$. HNSS transforms the problem into a series of smaller ones with asymptotic cost $\sum_{i=k}^{l} O(|\mathcal{A}_i|^3 |\mathcal{B}_i|^4 \log(|\mathcal{A}_i|))$, (where l is the level where the matching is found). This asymptotic cost is the same than classical RANSAC as full sets conform the basis of the hierarchy. Actual running time, however, depends mainly on the level where the matching in found. Section 3 shows how HNSS does help reduce computation time (see Table 1 for levels explored and Section 3.2 for computation time reduction).

Hierarchy construction

We focus on the decomposition of set $\mathcal{A}$, resulting in the hierarchy: $\mathcal{A}_0(=\mathcal{A}), \mathcal{A}_1, \ldots, \mathcal{A}_k$. The number of hierarchy levels is $k+1$, as k levels are built on top of the base set $\mathcal{A}$. The topmost level $\mathcal{A}_k$ contains, in light of the existing literature (Rusinkiewicz and Levoy, 2001), $\lfloor n/100 \rfloor$ points. The lowermost level $\mathcal{A}_0 = \mathcal{A}$, contains all n points. Intermediate levels increase their cardinality from top to bottom following: $|\mathcal{A}_i| = \lfloor \frac{n}{100} \rfloor + (n - \lfloor \frac{n}{100} \rfloor)^{\frac{k-i}{k}}$ for $0 < i < k$. Having less points in the higher levels is chosen over having many points with lots of redundant information. Section 3 presents results for hierarchies with varying number of levels.

Points in each level of the hierarchy are chosen considering that points with less frequent normals are more distinctive and should, thus, appear in higher levels of the hierarchy. First, the normals for all points in $\mathcal{A}$ are computed. The normal at each point is estimated as the normal to the fitting plane obtained by applying the total least square method to the k-nearest neighbours of the point (Hoppe et al., 1992). For the choice of k we used (Mitra et al., 2004). Then, following (Rusinkiewicz and Levoy, 2001), unitary normals are considered in the unit sphere. Using spherical coordinates, regular angle buckets are built by dividing the $]0, 2\pi[\times]0, 2\pi[$ parameter space in regular cells. Every point is assigned to its corresponding angle bucket. Finally, the points in each level are chosen:

- First we decide how many points are sampled from every bucket: The average contribution per bucket is noted $c_1 = \frac{|\mathcal{A}_i|}{128}$. Those buckets that have less than c_1 points contribute all their points. Then we iterate the process and compute the average contribution c_2 for the remaining buckets and number of points. Again, buckets with less than c_2 remaining points contribute all their points and the rest advance to the next iteration. The process stops when we reach the desired total contribution. Notice how points in buckets with lots of points are less likely to be sampled.
- Once we know the point contribution per bucket, we perform a uniform spatial sample with a random component to choose the points. For each bucket, we store points in a uniform 3D grid. This grid is made up of regular, axes-parallel, 3D cubical cells. Each of these cells contributes the same number of points so

the sample is uniform in space. Inside each grid cell, the points to be contributed are sampled randomly. Points not yet chosen for previous levels are sampled first. Only if not enough such points exist may points already present in other levels be chosen. There are two reasons for this type of sampling. First, to ensure that downsampled areas do not disappear in lower levels and second to have points as different as possible in each level.

The bucketing of angles results in down-sampling those buckets that have more points and, thus, focusing on those more descriptive in terms of normals. Sampling uniformly in each bucket helps us maintain a global approach and sample points from all over the set. This allows us to capture more details from the shape of set $\mathcal{A}$. Each level represents an NSS instance (Rusinkiewicz and Levoy, 2001).

RANSAC for point cloud matching

In this section we present details on the RANSAC algorithm used to illustrate the contribution of HNSS. Given any rigid motion μ we consider two parameters in order to characterize how good the matching between the point clouds is:

- The fraction of points in $\mu(\mathcal{B})$ that have a "matching" neighbour in $\mathcal{A}$. Specifically, given a d_{max} limit value, a point $\mu(b_j) \in \mu(\mathcal{B})$ is considered matched if $\exists a_i \in \mathcal{A}$ such that $d(a_i, \mu(b_j)) < d_{max}, d$ being the Euclidean distance. We note the objective value $final_p$.
- The second value corresponds to the average residue in terms of SSD for the matched points: The *residue* of a rigid motion μ is defined as: $res(\mu) = \frac{\sqrt{\sum_{k=1}^{m} contribution(\mu(b_k))^2}}{\#matched\_points}$ where

$$contribution(\mu(b_k)) = \begin{cases} (d(a_l, \mu(b_k)) & \text{if } (d(a_l, \mu(b_k)) <= d_{max} \\ 0 & \text{otherwise} \end{cases}$$

and l holds that $(d(a_l, \mu(b_k)) <= d(a_i, \mu(b_k)) \forall 0 < i <= n$. The objective value is $final_{res}$.

These two objective values ($final_p$, $final_{res}$) tailor the search depending on the type of matching expected: In some cases most of the points in set $\mathcal{B}$ are expected to be matched ("surface-to-model" matching). Alternatively some parts of one of the objects might not appear in the other, decreasing the required matching percentage ("surface-to-surface" matching). Demanding a lower residue will further limit the matchings accepted. Setting a maximum value for the nearest neighbour distance is important to prevent isolated points that do have a closest neighbour although it is comparatively very far away, distorting the residue. See Section 3 for details on experimental values for d_{max}, $final_p$ and $final_{res}$.

Descriptors

In some situations it is clear that one particular point $a_i \in \mathcal{A}$ does not correspond to a certain $b_j \in \mathcal{B}$. This happens, for example, when the local shapes around the points are not similar. Descriptors characterize the shape of the surface locally. Many descriptors exist as, for example, Point Signature (Chua, 1997) or Spin Image (Johnson, 1997). In order to focus on the main contributions of HNSS, we have not pruned the search using descriptor information. Notice, however, how HNSS advantages are also preserved when using descriptors.

RANSAC – HNSS interaction

HNSS permits to search for matchings automatically and to decide when a certain area of the search space has to be explored deeper. These decisions are guided by: (1) how descriptive points are in terms of normals and (2) when promising results are found. HNSS data hierarchy considers the first aspect. The second aspect is expressed by the parameters that describe the quality of matching (percentage of matched points and residue).

Several RANSAC instances are run using the sets in the HNSS hierarchy. For every particular RANSAC ($\mathcal{A}_i, \mathcal{B}_i$), the percentage of randomly sampled points decreases with i. For the topmost level an exhaustive search is performed (100% RANSAC sampling for the most descriptive points). The percentage decreases linearly up to 10% (classic RANSAC over the full set). By adaptively adjusting these percentages the algorithm pays more attention to points that stand out in terms of normals. Additionally, HNSS also improves promising matchings. This behavior is coded in the "Refine Matching" function in Algorithm 1: Whenever a triplet of points $a_{i_1}, a_{i_2}, a_{i_3} \in \mathcal{A}$ induces at least a motion between the sets, this triplet is considered to be *promising*. When one of this promising triplets is found the algorithm goes down one level or, if the percentage of matched points is closer to 10% of $final_p$, it descends to the bottom level. The algorithm then runs with the fixed $a_{i_1}, a_{i_2}, a_{i_3}$ triplet in the following level to improve the matching. The algorithm will invoke "Refine Matching" again if it manages to improve the percentage of matched points in this following level. Notice how the specific behavior of this function (concerning the percentage or the hierarchy level from which to start refining) can be fine-tuned for every particular application.

HNSS outline

HNSS looks for two triplets of points $(a_{i_1}, a_{i_2}, a_{i_3}) \in \mathcal{A}$, $(b_{j_1}, b_{j_2}, b_{j_3}) \in \mathcal{B}$ (defining three couples $(a_{i_1}, b_{j_1}), (a_{i_2}, b_{j_2}), (a_{i_3}, b_{j_3})$) so the objective residue and matched point percentage are met (Algorithm 1 presents an overview). In addition to using of normals and descriptors, we further prune the search space by considering distance restrictions. For example, if (a_{i_1}, b_{j_1}) is a couple, then: $d(a_{i_1}, a_{i_2}) = d(b_{j_1}, b_{j_2})$. Considering noise (Section 3) $d(a_{i_1}, a_{i_2}) \approx d(b_{j_1}, b_{j_2})$.

In order to save computing time during residue computations we use a Monte-Carlo (Black, 1999; Rajeev and Prabhakar, 1995) approach. We randomly choose a percentage of the points and obtain approximate values for the residue and for the percentage of matched points. When the algorithm is close enough to $final_p$ and $final_{res}$, full residue computations are performed. The percentage of sample points chosen was 10% in light of existing literature (Rusinkiewicz and Levoy, 2001).

Algorithm 1. Search for matching $(\mathcal{A}_i, \mathcal{B}_i)$

for all Points $a_{i_1} \in \mathcal{A}_i$
  for all Points $b_{j_1} \in \mathcal{B}_i$ which are descriptor compatible with a_{i_1}
    for all Points $a_{i_2} \in \mathcal{A}_i$
      for all Points $b_{j_2} \in \mathcal{B}_i$ which hold $d(a_{i_1}, a_{i_2}) \approx d(b_{j_1}, b_{j_2})$ and b_{j_2} is descriptor compatible with a_{i_2}
        for all Points $a_{i_3} \in \mathcal{A}_i$
          for all Points $b_{j_3} \in \mathcal{B}_i$ which hold $d(a_{i_1}, a_{i_3}) \approx d(b_{j_1}, b_{j_3})$ and $d(a_{i_2}, a_{i_3}) \approx d(b_{j_2}, b_{j_3})$ and b_{j_3} is descriptor compatible with a_{i_3}
          {Compute the Rigid Motions μ that brings $(b_{j_1}, b_{j_2}, b_{j_3})$ as close as possible to $(a_{i_1}, a_{i_2}, a_{i_3})$ in SSD terms,
          Compute the residue and percentage of matched points for μ}
          if μ is an acceptable matching, output μ and STOP
          else Refine Matching$(i, a_{i_1}, a_{i_2}, a_{i_3})$
end

3. Results and discussion

This section presents experiments (Johnson, 2002) to support the claims made throughout the paper. Experiments consider both real and synthetic data and where run on a desktop computer with 4 Intel i3 2,93 GHz nuclei and 8 RAM GByte (of which no more than 10% was needed during the execution of the algorithms). In order

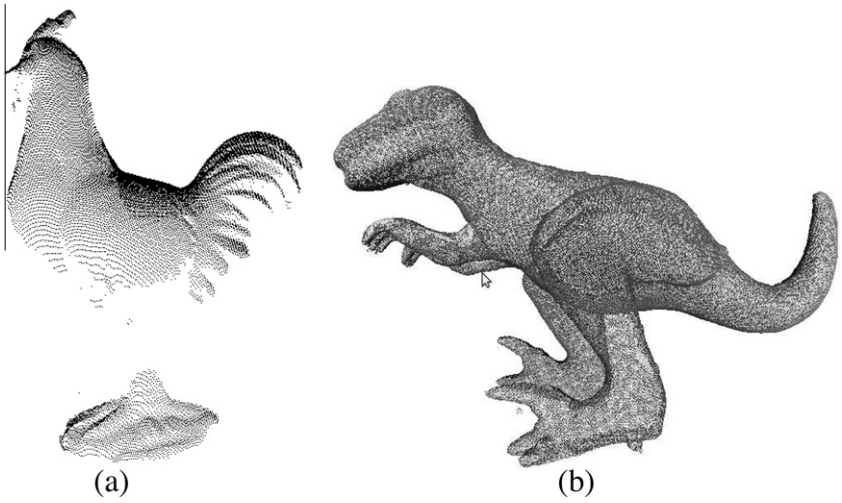

Fig. 3. Examples of the data used in synthetic experiments (a) chicken_view15 (b) Trex_full.

to focus on the novel algorithmic aspects, all executions where kept serialized and without compiler optimizations. C++ source code is available at: http://eia.udg.edu/~qsalvi/HNSSCODE.zip. We have used the public data (AJMAL) made available by Dr. Ajmal S. Mian (Mian et al., 2006; Ajmal Mian et al., 2006). Details on the sets used are provided at each experiment to ensure the reproducibility of results.

Sections 3.1 and 3.2 use synthetic data so that ground truth is known and noise tuned. Specifically:

- We chose a "real-life" $\mathcal{A}$ dataset among those at (AJMAL). We used mainly "Trex_full.ply" with 176508 points and "chicken_-view15.ply" with 21280 points (see Fig. 3), as well as several uniform smaller samples of them noted Tx, Cx (x indicating cardinality).
- We built the synthetic dataset $\mathcal{B}$ as a random rigid transformation of $\mathcal{A}$. The rotation angle was considered random in the interval $(0, 2\pi)$ and the translation ensured no initial overlapping between the two sets.
- We added a fixed, small, quantity of random noise. Although several noise values were considered, unless stated otherwise the maximum noise in experiments is 10% of the average nearest neighbour distance (Van Wamelen et al., 2004). This is higher than representation and rounding errors but still not big enough to be considered impractical and we consider it to represent the general tendencies observed for the noise values tested.

In order not to be mislead by particular results, every calculation was repeated 10 times with different random motions. Average values are reported. Concerning d_{max} (maximum allowed nearest-neighbour distance), we considered it to be twice the average neighbour distance, following (Van Wamelen et al., 2004).

Section 3.3 contains a study on the behavior of HNSS in 105 real life scenarios. All sections deal with full matching scenarios (see Fig. 1 for an example).

3.1. Run time reduction due to Hierarchical Normal Space Sampling

This test supports the claim that using HNSS reduces computing time and quantifies this reduction. In order to focus in the effect of the HNSS data hierarchy we considered the RANSAC sampling percentage to be 10% (Rusinkiewicz and Levoy, 2001) in all levels. Matchings were not refined to make the comparison fair for the classic RANSAC (which cannot do so). Best matching percentages for our sets usually were between 98% and 100% (see Section 3.2). In this case, we established a 90% matching percentage in order to set up a not-so-demanding situation.

Table 1 presents results obtained by classic RANSAC compared to HNSS with 4, 7 and 10 hierarchy levels. We present the number of sextets examined by each algorithm ($\#IT$ column), time needed for residue computations (t_{res}), total time t_{total} and the level (0 being the lowermost) where the matching was found ("Last level"). As results are averages of 10 executions "Last level" is often not an integer. Computing times are in seconds. Sets were kept of small cardinality due to the huge computation times required by classic RANSAC.

HNSS reduces average computing time in all executions. In the most extreme, this reduction is of more than 25000 s (seven hours). We have observed how the time (even taking into account the 10 repetitions) depends greatly on when a good enough sextet is found, so sets of the same cardinality might present quite different computing times. Hence, it is important to search the parameter space in a way that permits the fast location of such sextets.

From the 49 datasets considered, the classic Ransac was faster only in one (involving T100), in the remaining 48 HNSS algorithms run faster. This shows the usability of HNSS. Moreover, the variability of the average level where matchings were found highlights the difficulty to decide beforehand how many points to take for a classic NSS.

3.2. Does HNSS matching reach "far enough"?

Being HNSS a sampling strategy, a reasonable concern is whether it misses meaningful solutions. To show how that is not the case, in this experiment we executed HNSS with varying number of levels setting highly demanding requirements for the matching. As set $\mathcal{B}$ is basically a rigid motion of set $\mathcal{A}$ with added noise, the maximum percentage of matched points was expected to be high. In order to get a precise ground truth alignment, we picked three known corresponding couples and computed the percentage of matched points and residue of the associated motion. Values higher than 95% of matched points were generally obtained.

We executed HNSS with the improvements described throughout the paper (adaptive hierarchy with varying numbers of levels and monte-carlo residue calculation). Fig. 4 shows compared computing times (average values for 10 executions). The left subfigure corresponds to subsets of varying cardinality of the "chicken_-

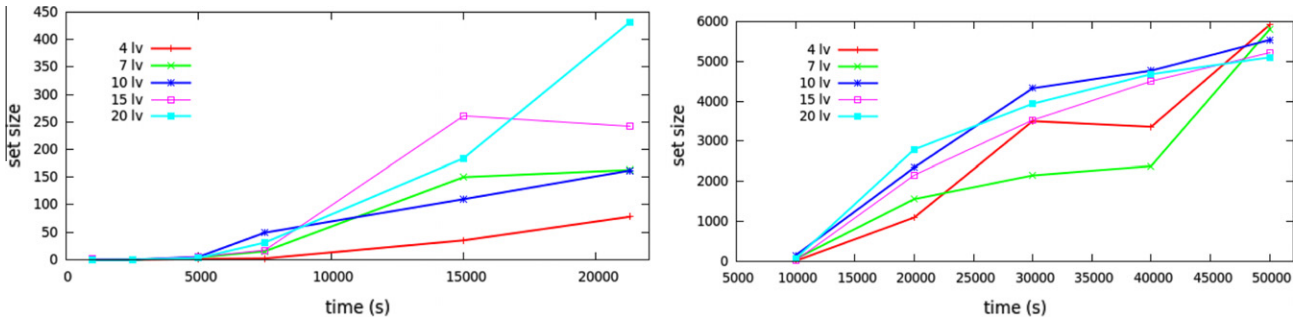

Fig. 4. Performance of HNSS with different number of levels (synthetic data) and for different samples of two different objects: chicken_view_15 (left) and TREX_full (right). Horizontal axes depict the number of points in the set and vertical axes show computing time (in seconds).

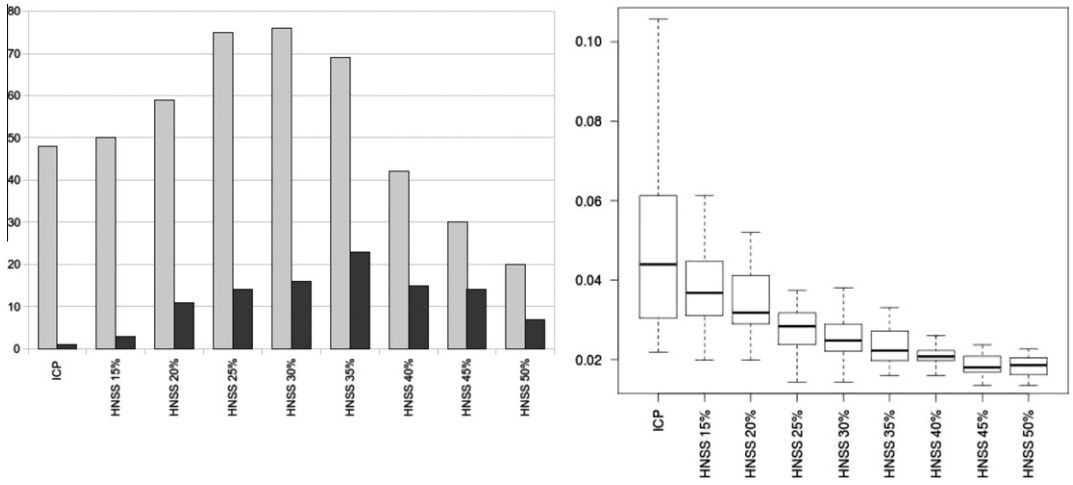

Fig. 5. HNSS + ICP with real data. Percentage of success (gray) and of best matching (black) (left), average residue in successful alignments (right).

view_15" object. The right subfigure depicts data of the TREX_full object.

Results of standard RANSAC was left out of Fig. 4 in order to focus in HNSS (as stated in Section 3.1 RANSAC was much slower than HNSS). HNSS with a two level hierarchy, which stands for a classical 1% NSS worked "very fast" in approximately 29% of the cases, as it managed to find a matching in the first level and "very slowly" in the remaining 71%, as it resorted to a classical RANSAC exploration. This shows how HNSS is a more robust alternative to NSS. To sum up, we can say that HNSS performs faster than RANSAC and is also more accurate than NSS.

3.3. Performance in real-life situations

Finally, we tested HNSS using real data to illustrate how coarse matching is a necessary initialization for ICP: We demonstrate how,unless a good enough initial alignment is known, ICP might converge to a local minimum or might not converge at all. Computing times and insight on the strengths and challenges of HNSS-based algorithms are also presented.

We used 15 views of to the chicken object available at (AJMAL). Considering them in pairs yielded to 105 matching data sets. The algorithms tested were:

- ICP without previous coarse matching. This stood for an execution of ICP with a "naive" initial guess.
- 7 level HNSS hierarchy + RANSAC coarse matching algorithm followed by ICP. This represents a typical use of coarse matching as ICP initialization.

ICP improvement

A few of these "naive" alignments were good initial approximations for ICP. The rest caused ICP not to converge or stall at local minima (see Fig. 1 for an example). Specifically, ICP failed to converge in 57 out of the 105 data sets. Concerning local minima, ICP gave a solution in 48 data sets though the overlap percentage and average residue was improved by using HNSS in 47 of them. Consequently, HNSS has been successfully used in a real-life experiment to enhance the application scope of ICP and improve the results obtained.

HNSS performance analysis

For these real data sets ground truth is not available. Moreover, every matching data set was expected to have its own optimal matched point percentage and average residue. Consequently, the evaluation of results becomes more difficult. We tested a 7-level HNSS hierarchy in the 105 data sets requesting matching percentages from 15% to 50% in 5% increments.

Intrinsic to any coarse matching, in some cases the overlap requested was not demanding enough and in others it was too demanding to be reached. In the first, HNSS was fast but risked not being a good enough initial approximation causing ICP not to converge or stall at a local minimum. In the second, the adaptable search was wasted looking for a solution that was not there.

Fig. 5 (left) shows the percentage of successful alignments over the 105 data sets and the percentage of these alignments that represented the best alignment possible. As expected, ICP with naive alignment is greatly improved by HNSS. HNSS obtains better results and we observe how the percentage of best matching in successful alignments is higher for higher percentages of requested

overlap. This shows how, whenever an alignment exists, HNSS is able to find better solutions. The highest concentration of best matchings is found in the 30%–40% range of requested matching percentage, which is consistent with our own observation of the data.

Additionally, Fig. 5 (right) depicts box plots of the average residues for successful alignments. Demanding higher matching percentage gave alignments with smaller average residue. Consequently, setting the right matching percentage results in finding a better solution.

4. Conclusions

In this paper we have presented a new speed-up technique for coarse matching called HNSS (Hierarchical Normal Space Sampling). The code can be downloaded at: http://eia.udg.edu/~qsalvi/HNSSCODE.zip. We have shown its effectiveness compared to classical RANSAC.

HNSS is an improvement of Normal Space Sampling (NSS) that relies on normal sampling and distance restrictions to explore the search space in a way that finds results much faster and, as we are dealing with a threshold problem, stops the search earlier. HNSS refines the matchings found in order to spend more time in the areas of the search space that show better expectations. HNSS runs faster than the classic RANSAC. In Section 3.1 we have presented examples when computations of seven hours were reduced to less than a second. Another characteristic of HNSS is its capacity to adapt automatically to different sets without human intervention.

HNSS performed better than NSS in the sense that, while the NSS had to resort to working with the full sets in 71% of the cases, HNSS algorithms did not need to do it in any of them. Besides, the reduction in computation time achieved by HNSS does not mean that the search is less effective. In Section 3.2 we have seen how HNSS performed well even in very demanding situations.

In Section 3.3 we have tested HNSS in real life. We have shown how ICP with naive alignment is often unable to converge. Additionally, even when ICP converges, an initial alignment given by HNSS managed to increased the matched point percentage and decrease the average residue in 104 out of 105 data sets. This shows the necessity of HNSS. We have also explored the effect of a right choice of the overlap percentage and shown how HNSS adapts to find the best alignment in terms of average residue.

Finally, some ideas that will guide our future work: HNSS could benefit from using state of the art shape descriptors as well as from parallel computing and also from the combination with other RANSAC-based algorithms such as (Aiger et al., 2008).

Acknowledgements

This work has been supported by FP7-ICT-2011-7 projects: PANDORA Persistent Autonomy through Learning, Adaptation, Observation and Re-planning (Ref 288273) funded by the European Commission and RAIMON Autonomous Underwater Robot for Marine Fish Farms Inspection and Monitoring (Ref CTM2011-29691-C02-02) funded by the Spanish Ministry of Science and Innovation.

References

Aiger, D., Mitra, N.J., Cohen-Or, D., 2008. 4-Points congruent sets for robust surface registration. ACM Trans. Graphics 27 (3), 1–10.
Dr. AJMAL, S. MIAN's Home Page. http://www.csse.uwa.edu.au/ajmal/
Black, Paul E., 1999. Algorithms and Theory of Computation Handbook. CRC Press, Monte Carlo algorithm, in Dictionary of Algorithms and Data Structures [online], ed., U.S. National Institute of Standards and Technology. 17 December 2007. Available from: http://xw2k.nist.gov/dads/HTML/monteCarlo.html.
Besl, P., McKay, N., 1992. A method for registration of 3-d shapes. IEEE Trans. Pattern Anal. Machine Intell. 14 (2), 239–256.
K. Brunnström, A. Stoddart, 1996. Genetic algorithms for free-form surface matching. In: International Conference of Pattern Recognition, Vienna, pp. 689-693.
Chua, C.J.R., 1997. Point signatures: a new representation for 3D object recognition. Internat. J. Comput. Vision 25 (1), 63–85.
Chung, D., Lee, Y.D.S., 1998. Registration of multiple-range views using the reverse-calibration technique. Pattern Recognition 31 (4), 457–464.
Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24, 381–395.
Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W., 1992. Surface reconstruction from unorganized points. Comput. Graphics 26 (2), 71–78.
Johnson, A., 1997. Spin-images: a representation for 3-D surface matching, Ph.D. thesis, Carnegie Mellon University, USA.
Johnson, D.S., 2002. A Theoretician's Guide to the Experimental Analysis of Algorithms. In: Goldwasser, M, Johnson, D.S., McGeoch, C.C. (Eds.), Proceedings of the fifth and sixth DIMACS implementation challenges, American Mathematical Society, Providence, RI, pp. 215–250.
Jost, T, Hügli, H, 2002. A Multi-Resolution Scheme ICP Algorithm for Fast Shape Registration. In: Proceedings 3DPVT2002, pp. 540–543.
Mian, Ajmal S., Bennamoun, M., Owens, R., 2006. A novel representation and feature matching algorithm for automatic pairwise registration of range images. Internat. J. Comput. Vision (IJCV) 66 (1), 19–40.
Mian, Ajmal, Bennamoun, M., Owens, R., 2006. 3D model-based object recognition and segmentation in cluttered scenes. IEEE Trans. Pattern Anal. Machine Intell. (IEEE TPAMI) 28 (10), 1584–1601.
Matabosch, C., Fofi, D., Salvi, J., Batlle, E., 2008. Registration of surfaces minimizing error propagation for a one-shot multi-slit hand-held scanner. Pattern Recognition 41 (6), 2055–2067.
Mitra, N.J., Nguyen, A., Guibas, L., 2004. Estimating Surface Normals in Noisy Point Cloud Data, Special issue of Int. J. Comput. Geometry Appl. 14 (4–5), 261–276.
Rajeev, Motwani, Prabhakar, Raghavan, 1995. Randomized Algorithms. Cambridge University Press, New York, ISBN 0521474655.
Masuda, Takeshi, Yokoya, Naokazu, 1995. A robust method for registration and segmentation of multiple range images. CVIU 61 (3), 295–307.
Rusinkiewicz, Szymon, Levoy, Marc, 2001. Efficient Variants of the ICP Algorithm. In: Third International Conference on 3D Digital Imaging and Modeling (3DIM 2001) pp. 145–152.
Salvi, J., Matabosch, C., Fofi, D., Forest, J., 2007. A review of recent range image registration methods with accuracy evaluation. Image Vision Comput. 25, 578–596.
Sorkine, O., 2007. Least-Squares Rigid Motion Using SVD. ARAP project technical note. <http://igl.ethz.ch/projects/ARAP/svdrot.pdf>.
Salvi, J., Batlle, B., Matabosch, C., Llado, X., 2008. Overview of surface registration techniques including loop minimization for 3D modelling and visual inspection. J. Electron. Imaging 17 (3), 031103.
Trucco, E., Fusiello, A., Roberto, V., 1999. Robust motion and correspondences of noisy 3-D point sets with missing data. Pattern Recognition Lett. 20 (9), 889–898.
Turk, G., Levoy, M., 1994. Zippered Polygon Meshes from Range Images. In: Proceedings SIGGRAPH94, pp. 311–318.
Van Wamelen, P.B., Li, Z., Iyengar, S.S., 2004. A fast expected time algorithm for the 2-D point pattern matching problem. Pattern Recognition 37 (8), 1699–1711.
Wei, H., Zhang, L., Liu, S., Shi, C., 2010. An algorithm on registration of multi-view range images based on SIFT feature matching. Jisuanji Fuzhu Sheji Yu Tuxingxue Xuebao/J. Computer-Aided Design Comput. Graphics 22 (4), 654–661.
Xie, Z., Xu, S., Li, X., 2010. A high-accuracy method for fine registration of overlapping point clouds. Image Vision Comput. 28 (4), 563–570.
Zinsser, T., Schnidt, H., Niermann, J., 2003. A refined ICP algorithm for robust 3-D correspondences estimation. Internat. Conf. Image Process., 695–698.