Latent RANSAC

Simon Korman
Weizmann Institute of Science, Israel

Roee Litman
General Motors, Israel

Abstract

We present a method that can evaluate a RANSAC hypothesis in constant time, i.e. independent of the size of the data. A key observation here is that correct hypotheses are tightly clustered together in the latent parameter domain. In a manner similar to the generalized Hough transform we seek to find this cluster, only that we need as few as two *votes for a successful detection. Rapidly locating such pairs of similar hypotheses is made possible by adapting the recent "Random Grids" range-search technique. We only perform the usual (costly) hypothesis verification stage upon the discovery of a close pair of hypotheses. We show that this event rarely happens for incorrect hypotheses, enabling a significant speedup of the RANSAC pipeline.*

The suggested approach is applied and tested on three robust estimation problems: camera localization, 3D rigid alignment and 2D-homography estimation. We perform rigorous testing on both synthetic and real datasets, demonstrating an improvement in efficiency without a compromise in accuracy. Furthermore, we achieve state-of-the-art 3D alignment results on the challenging "Redwood" loop-closure challenge.

1. Introduction

Despite the recent success of (deep-) learning based methods in computer vision, numerous applications still use "old-fashioned" robust estimation methods for model fitting, such as RANSAC [20]. This is especially true for problems of a strong geometric nature such as image alignment, camera localization and 3D reconstruction. Robust estimation methods of these types largely follow the "hypothesize and test" paradigm which has strong roots in statistics, and are highly attractive due to their ability to fit a model to data that is highly corrupted with outliers. Additionally, they have been successfully applied to many problems in computer vision and robotics achieving real time performance.

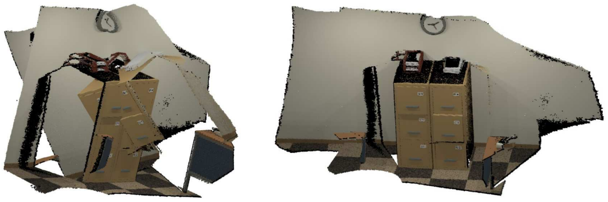

Figure 1. **3D alignment result** of two methods on a pair of fragments from the "Redwood" dataset [12]. Our method (right) produces a correct alignment, even though the putative matches contain a mere 2% of inliers, which is very challenging. On the left, we see a failure case of the method from [45], even though it manages to increase inlier rate up to 7%. Runtimes for this example are 74ms for our method, and 163ms for [45]. Table 4 shows representative examples for the other problems we handle - PnP and 2D-Homography estimation.

As an example, in the field of image (or shape) alignment, novel features and descriptors have been introduced to facilitate matching, including ones that are learned. However, once these features are matched, for a parametric model to be fitted, robust estimation methods like RANSAC are used to cope with corrupted sets of putative matches.

Geometric models that are commonly amenable to such a robust estimation process include: 2D-homography, camera localization, the essential and fundamental matrices that describe epipolar constraints between images, rigid 3D motion and more.

1.1. Background and prior art

Consensus maximization has proven a useful robust estimation approach to solving a wide variety of fitting and alignment problems in computer vision.

Research in this field can be broadly divided into *global* and *local* optimization methods. Global methods [34, 44, 10, 8] use different strategies to explore the entire solution space enjoy the advantage of having a deterministic nature. Our method, however, belongs to the family of local methods which are typically extremely fast randomized algorithms, potentially equipped with probabilistic success guarantees.

While the proposed method is presented in the context of RANSAC, it is closely related-to and inspired-by other works in the field, such as Hough voting. We cover these topics briefly.

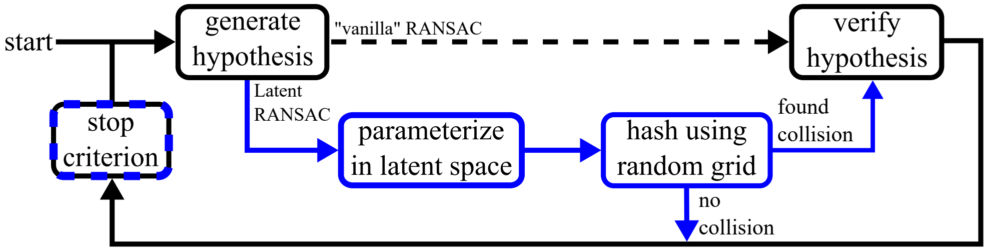

Figure 2. **A flow chart of RANSAC compared to the suggested method.** We propose an alternative flow (in blue), in which after a hypothesis is generated it first undergoes a hash procedure, and only verified if a (valid) collision is detected. The stop criterion has to be modified as well, to ensure a *second* good hypothesis is drawn with high probability.

RANdom SAmple Consensus (RANSAC) [20] is one of the de-facto golden standards for solving robust estimation problems in a practical manner. Under this paradigm, the space of solutions is explored by repeatedly selecting random minimal subsets of a set of given measurements (e.g. putative matches), for which a model hypothesis is fitted. These hypotheses are verified by counting the measurements that agree with them up to a predefined tolerance. This process is repeated until a desired probability to draw *at-least* one pure set of inliers is achieved.

The RANSAC paradigm is a very studied topic, and many methods were suggested to accelerate sampling [39, 13, 21, 22], improve stability [15, 30], or even estimate the tolerance parameter [11, 36]. Some of these extensions are covered in a recent comprehensive survey by Raguram *et al*. [35]. This survey also suggests USAC – a framework that combines some RANSAC extensions, yielding excellent results on a variety of problems in terms of accuracy, efficiency, and stability.

While the previous extensions can be seamlessly applied along with the suggested method, the following extension is similar in nature to ours in that it aims to speed up the verification step, but it does so in a very different manner. The Sequential Probability Ratio Test (SPRT) [14] extension was selected in USAC out of several similar methods [9, 14] SPRT is based on Wald's sequential test [41]. It attempts to reject a "bad" model with high probability, after inspecting only a small fraction of the measurements. While the test is theoretically solid, it relies on two parameters that are assumed to be known a priori, and in practice need to be adaptively estimated at runtime. It is reported to have achieved an improvement of 20% in evaluation time compared to the simpler bail-out test [9].

Generalized Hough transform (GHT) originated from an algorithm for line detection in images [24], which was later generalized to handle arbitrary shapes [19, 7]. The key idea behind this method is that partial observations of the model are casted as votes into a (quantized) solution space, in which the object can be detected as a mode (the location with the most votes). In practice, GHT has not been shown to scale well to solution spaces of high dimensionality (i.e. higher than 3), and typically requires *numerous* votes for a mode to be accurately detected.

Between RANSAC and GHT. Some works bare resemblance to both of the mentioned approaches. Our work can be seen as one of these: While it fits naturally into the RANSAC pipeline, it has some similarities to GHT in the sense that it seeks to find the mode in the parameter domain, only that it needs as few as *two* votes to detect it.

A method by Den-Hollander et al. [16] also lays somewhere between RANSAC and GHT: To increase the probability of obtaining a pure set of inlier matches, a sub-minimal set is drawn. The remaining degrees of freedom (DoF) are resolved using a voting scheme in a low-dimensional setting. As with all Hough-like methods, an adequate parameterization of the remaining DoF is required. The authors of [16] provide such a parameterization for the problem of fundamental matrix estimation.

Our method bares a strong resemblance to the "Randomized Hough Transform" (RHT) [42] of Xu et al. in that a vote is casted into a single cell in the solution domain, generated from a randomly selected minimal set. However, unlike [42], we deal with a hypothesis in *constant time and space*, rather than logarithmic, thanks to the Random Grid hashing mechanism that we adapt. In addition, while RHT deals with robust curve-fitting (of up to 3 dimensions), we successfully apply our method on a variety of problem domains of higher dimensionality (up to 8 dimensions).

1.2. Contributions

The main novelty of the presented method is its ability to handle RANSAC hypotheses in *constant time*, regardless of the number of measurements (e.g. matches). We show that it is beneficial to handle hypotheses in the latent space, due to an efficient parametrization and hashing scheme that we devise, which can quickly filter candidate hypotheses until a pair of correct ones are drawn. While this approach comes at the expense of a *small* increase in the number of hypotheses to be examined, it allows for a significant speedup of the RANSAC pipeline.

The new proposed modifications to RANSAC are accompanied by a rigorous analysis which results in an updated stopping criterion and a well understood overall probability of success. Finally, we validate our method using challenging data in the problems of 2D-homography estimation, 2D-3D based camera localization and rigid-3D alignment, showing state-of-the-art results.

2. Method

The 'vanilla' RANSAC pipeline can be divided into three main components: hypothesis generation, hypothesis verification and the adaptive stopping mechanism. The proposed Latent-RANSAC hypothesis handling fits naturally into the aforementioned pipeline, as can be seen in Figure 2, highlighted in blue. The additional modules we propose act as a 'filter' that avoids the need to verify the vast majority of generated hypotheses: Instead of verifying each hypothesis by applying it on all of the matches (a costly process that takes time linear in the number of matches), we check in *constant time* if a previously generated hypothesis 'collides' with the current one, i.e. whether they are close enough (in a sense that will be clarified below). Only the very few hypotheses that pass this filtering stage progress to the verification stage for further processing. As a result of the proposed change, the RANSAC stopping criterion needs to modified to guarantee a probability of encountering a second good hypothesis rather than just one.

Outline We begin by covering the 3 key components of our method: parametrization of the solution space (Section 2.1), Random Grids hashing (Section 2.2) and the modified stopping criterion (Section 2.3). We conclude this part of the paper in Section 2.4, with an analysis of our Random Grids hashing process.

Preliminary definitions In our setup, the goal is to robustly fit a geometric model (transform) to a set of matches (correspondences), w.l.o.g. in Euclidean space, where a *match* $\mathbf{m} = (\mathbf{p}, \mathbf{q})$ is an ordered pair of points $\mathbf{p} \in \mathbb{R}^d$ and $\mathbf{q} \in \mathbb{R}^{d'}$. For a geometric transform $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ and match $\mathbf{m} = (\mathbf{p}, \mathbf{q})$ the *residual error* of the match $\mathbf{m}$ with respect to f is the Euclidean distance in $\mathbb{R}^{d'}$ given by:

$$\mathrm{err}(f, \mathbf{m}) = \|\mathbf{q} - f(\mathbf{p})\|. \tag{1}$$

Given a set of matches $M = \{\mathbf{m}_i\}$ and a tolerance t, the *inlier rate* achieved by a transform f is defined as the fraction of matches $\mathbf{m}_i \in M$ for which $\mathrm{err}(f, \mathbf{m}_i) \leq t$. We denote the maximal inlier rate for a match-set M by ω.

2.1. Parametrization of the solution domain

In the RANSAC pipeline, matches are used *both* for the generation of hypothesis candidates, as well as for their screening. Since our approach performs the majority of the screening according to some 'similarity' in the space of transformations, we seek a *parametrization* of the transformation space in which distances between transformations can be defined explicitly. More formally, we define such a parametrization by an *embedding* hypotheses into some λ-dimensional space $\mathbb{R}^\lambda$, which we call the *latent* space[1]. We consider the distance between transformations to be given by the ℓ_∞ metric between the embedded (or latent) vectors (λ-tuples). Our goal is to use an embedding in which the *distance* between any pair of hypotheses f_1 and f_2 is tightly related to the *difference* in the way these hypotheses act on matches in the source domain, i.e. to the difference in magnitudes of their residual errors on the matches. Ideally, for *any* set of matches M,

$$\|f_1 - f_2\|_\infty \propto \max_{\mathbf{m} \in M} |\mathrm{err}(f_1, \mathbf{m}) - \mathrm{err}(f_2, \mathbf{m})|. \tag{2}$$

2D homography. We describe here the parameterization we use for the space of 2D homographies, which are given by projective matrices in $\mathbb{R}^{3\times3}$. Following previous works (e.g. [31, 17]), we use the 4pt parameterization [6] that represents a 2D-homography $H \in \mathbb{R}^{3\times3}$ by an 8-tuple $\mathbf{v}_H$, defined by the coordinates in the target image that are the result of applying H on the four corners of the source image, as illustrated in Figure 3.

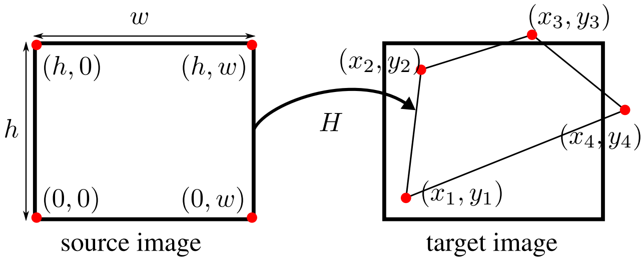

Figure 3. **Illustration of the 4pt homography parametrization [6].** A homography H is represented in the latent space by mapping the location of the four corners of the source image onto the target image, resulting in the 8-tuple $\mathbf{v}_H = (x_1, y_1, \ldots, x_4, y_4)$.

As was noted in [31], this parametrization has the key property that the difference between match errors of two well-behaved homographies is bounded by the ℓ_∞ distance between their 4pt representations.

The special Euclidean group $SE(3)$, used to describe rigid motion in $\mathbb{R}^3$, will be used here to solve the problems of Perspective-n-Point (PnP) estimation and Rigid 3D alignment. We follow a parametrization that was suggested and used in a line of works of Li et al. [43, 8]. The group $SE(3)$ can be described as the product between two subgroups $SE(3) = SO(3) \times \mathbb{R}^3$, namely 3D translations and the special orthogonal group (3D rotations). Each of these

[1] the latent space dimension λ typically being the number of degrees of freedom of the transformation space.

3-dimensional sub-groups is parameterized as a 3-tuple, resulting in a 6-tuple representation (r, t) defined as follows: The axis-angle vector (3-tuple) r represents the 3D rotation matrix given by $R_r = \exp([r]_x)$, where $\exp(\cdot)$ is the matrix exponential and $[\cdot]_x$ denotes the skew-symmetric matrix representation. Such vectors r reside in the radius-π ball that is contained in the 3D cube $[-\pi, \pi]^3$. The translation 3-tuple t is a vector in the cube $[-\xi, \xi]^3$ that contains the relevant bounded range of translations for a large enough ξ.

Similar to the case of the 2D-homography parametrization, it is proved in [43] that the difference between match errors of two rigid motions is bounded by the ℓ_2 distance between their parametrization.

2.2. Random Grids hashing

Given such a embedding of a generated hypotheses, the heart of our method boils down to a nearest neighbor query search of the current vector through all vectors representing previously generated hypotheses. More precisely, the task needed to be performed is a range search query for vectors that are at a distance of up to a certain tolerance t.

A recent work of Aiger et al. [2] turns out to be extremely suitable for this task. They propose Random Grids - a randomized hashing method based on a very simple idea of imposing randomly shifted 'grids' over the vector space, checking for vectors that 'collide' in a common cell. The Random Grids algorithm is very fast, and simple to implement - even in comparison with the closely related LSH-based algorithms [4], since the grid is axis aligned and it is uniform (consists of cells in $\mathbb{R}^d$ with equal side length). Most important, and essential for the speed of our method, is that the range search is done in constant time (i.e. it does not depend on the number of vectors searched against), as opposed to the RANSAC hypothesis validation that requires applying the model and measuring errors on (typically hundreds of) point matches or even the logarithmic-time solution proposed in RHT [42].

Hashing scheme. We are given a representation of the transform f as a vector $\mathbf{v} \in \mathbb{R}^\lambda$ (for a λ-dimensional parameterization). In the Random Grids [2] setting, we hash $\mathbf{v}$ into L hash tables $\{T_i\}_{i=1}^L$, each associated with an independent random grid, which is defined by a uniform random shift $O_i \sim U([0, c]^\lambda)$, where c is the cell side length and λ is the dimension of the latent vector $\mathbf{v}$. The cell index for $\mathbf{v}$ in the table T_i is obtained by concatenating the integer vector $\mathbf{z}_i = \left\lfloor \frac{\mathbf{v}+O_i}{c} \right\rfloor$ into a single scalar (where $\lfloor\cdot\rfloor$ means "floor" operation). The entire hashing process - initialization, insertion and collision checking, is given in detail in Algorithm 1.

```
input: (incremental) A candidate transform (matrix) $f$
parameters: number of tables $L$; tolerance $t$; cell dim. $c$; parametrization dim. $\lambda$;

initialization:
foreach $i = 1, \dots, L$ do
    1. Initialize an empty hash table $T_i$.
    2. Randomize offset $O_i \sim U([0, c]^\lambda)$
end
insertion and collision check for hypothesis $f$:
foreach $i = 1, \dots, L$ do
    1. Let $\mathbf{v}$ be the embedding of $f$
    2. The hash index for $\mathbf{v}$ is: $\tau_\mathbf{v} = hash\left(\left\lfloor \frac{\mathbf{v}+O_i}{c} \right\rfloor\right)$
    3. If the cell $T_i[\tau_\mathbf{v}]$ is occupied by a vector $\mathbf{u}$, report
       a collision of $f$ if $\|\mathbf{v} - \mathbf{u}\|_\infty < t$
    4. Store $\mathbf{v}$ in $T_i[\tau_\mathbf{v}]$
end
```

Algorithm 1: Latent-RANSAC hypothesis handling.

2.3. Latent-RANSAC stopping criterion

The classical analysis of RANSAC provides a simple formula for the number of iterations n required to reach a certain success probability p_0 (e.g. 0.99). It is based on the assumption that it is sufficient to have a single 'good' iteration in which a pure set of inliers is drawn. Note that this assumption is made for the simplicity of the analysis and is only theoretical, since it ignores e.g. the presence of inlier noise and several possible degeneracies in the data.

Formally, let G_n be the random variable that counts the number of such good iterations out of n attempts. For a minimal set of size γ and data with inlier rate of ω, it holds that

$$p_0 = 1 - P[G_n = 0] = 1 - (1-p)^n \qquad (3)$$

where $p = \omega^\gamma$. The number of iterations n required to guarantee a desired success probability p_0 is therefore:

$$n = \frac{\log(1-p_0)}{\log(1-p)} \qquad (4)$$

A similar simplified analysis can be applied to the Latent-RANSAC scheme. Ignoring the presence of inlier noise, the existence of (at least) two 'good' iterations is needed for a collision to be detected and the algorithm to succeed. Therefore, by the binomial distribution we have that

$$p_0 = P[G_n \geq 2] = 1 - (1-p)^n - n{\cdot}p{\cdot}(1-p)^{n-1}. \quad (5)$$

Based on equations (4) and (5), we plot in Figure 4 the *ratio* between the number of required iterations n in the case

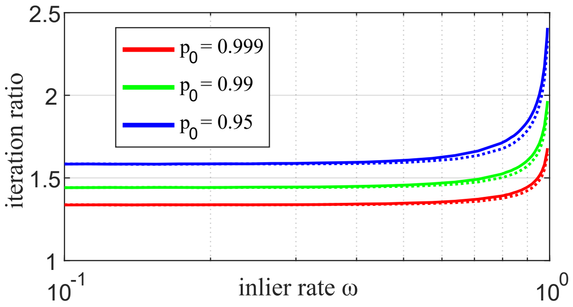

Figure 4. **The ratio between stopping criterions** of Latent-RANSAC (5) and of RANSAC (4). Ratios are shown as a function of inlier rate ω for several success probabilities (color coded) and for γ (minimal sample size) values of 3 (dashed) and 4 (solid). See text for details.

of Latent-RANSAC versus the case of RANSAC. The ratio is given as a function of the inlier rate ω, at 3 different success rates p_0 (color coded), for the two different cases $\gamma = 3$ (e.g. in Rigid 3D motion estimation) and $\gamma = 4$ (e.g. in homography estimation). Interestingly, the ratio attains a small value (less than 2) for inlier rates below $\omega = 0.95$, and converges to small constant values as the inlier rate decreases. The very high inlier rates for which the ratio is large are of no concern, since the absolute number n is extremely low in this range.

In the next section, as part of an analysis of the Random Grid hashing, we derive a more realistic stopping criterion that depends also on the success probability of the Random Grid based collision detection, which clearly depends on the inlier noise level.

2.4. Random Grid analysis

We cover two aspects of Random Grids. First, we extend the stopping criterion from Section 2.3 to consider the probability that a colliding pair of good hypotheses will be detected. Next, we discuss causes of false collision detection, which can have an affect on the algorithm runtime.

stopping criterion. Let $R(i)$ be the event that the random grid component succeeds (detects a collision), given i good iterations out of a total of n. We can now update Equation (5), taking this success probability into account:

$$p_0 = \sum_{i \geq 2} P[G_n = i] \cdot P[R(i)] \;\geq\; P[G_n \geq 2] \cdot P[R(2)] \quad (6)$$

where the inequality holds due to the fact that $P[R(i)]$ monotonically increases with i.

A final lower bound on p_0 (from which the stopping criterion is determined) can be obtained by substituting the expression for $P[G_n \geq 2]$ from (5) into (6) together with a lower bound on $P[R(2)]$ which we provide next.

Recall that $R(2)$ is the event that the random grid hashing succeeds given that two successful hypotheses were generated. We will, more explicitly, denote this event by $R_L(2)$, for a random grid that uses L hash tables.

The analysis in [2] is rather involved since it deals with the Euclidean ℓ_2 distance. Using ℓ_∞ distances we are able to derive the success probability of finding a true collision in our setup, as a function of the random grid parameters, in a simpler manner. Assuming a tolerance t in the latent space, determined by (inlier) noise level of the data, using a random grid with cell dimension c and a *single* table results in

$$P[R_1(2)] \geq \left(1 - \frac{t}{c}\right)^{\gamma} \quad (7)$$

since a pair of pure-inlier transformations (which differ by at most t) must share the same independently offsetted bin indices in each of the γ dimensions.

Finally, using L hash tables, randomly and independently generated, we obtain:

$$P[R(2)] = P[R_L(2)] \geq 1 - (1 - P[R_1(2)])^L \quad (8)$$

False collisions. We now discuss the expected number of false collisions that are found by the hashing scheme. It is important to understand why false collision might happen, as they have an effect on the overall runtime of our pipeline.

Recall that n is the overall number of iterations of the pipeline, and hence it is also the total number of samples inserted into each hash table. There are two kinds of false collisions to consider. The first kind happens due to the fact that the random grid cell size c might be larger than the tolerance t. Following the recommendation in [1] we set the cell size c to be not much larger than the tolerance t, resulting in a small number of such false collisions. In any case, this kind of collision has a small impact on the runtime, since it will be filtered by the tolerance test (step 3 in Algorithm 1) at constant time cost.

The second kind of false collision is one that passes the tolerance test (step 3). Since it is not the true model, it is associated with some inlier rate ζ. If $\zeta \ll \omega$, the probability of this collision appearing before we have reached the stopping criterion is negligible. Empirically, we observe very few (typically less than 15) collisions that pass the tolerance test up to the stopping of the algorithm. These are the only kind of collisions that incur a non-negligible penalty (in runtime only) since they invoke the verification process that every "vanilla" RANSAC hypothesis goes through.

3. Results

In order to evaluate our method, we performed extensive tests on both real and synthetic data. The Latent-Ransac algorithm is applied to the problems of 2D-homography estimation (Section 3.1), Perspective-n-Point (PnP) estimation (Section 3.2) and Rigid 3D alignment (Section 3.3). It is compared with USAC, with or without the well known

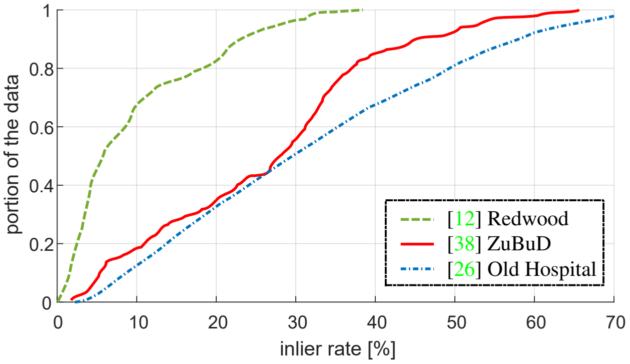

Figure 5. **Inlier rate cumulative distribution** (CDF) of the three real data sets we use. The dashed curve was taken only over Redwood pairs with provided ground truth.

SPRT [14] extension, which is a very different technique for accelerating RANSAC's model verification phase.

Implementation details. Our method naturally extends the standard RANSAC pipeline, according to the changes highlighted in Figure 2. Our implementation (for which we use the shorthand LR) extends the excellent C++ implementation USAC [35], with the noted changes in the specific modules. This enables an easy way to compare with a state-of-the-art RANSAC implementation, and allows our method to enjoy the same extensions used by USAC (such as its local optimization component LO-RANSAC [15]). In addition, their implementation includes the SPRT [14] extension, the most commonly used acceleration of RANSAC's model verification phase. We use the shorthand SPRT to refer to USAC using this extension, and RANSAC when not using it.

In addition, we use the OpenGV library [27] for PnP model fitting (Kneip's P3P algorithm [28]) and for Rigid 3D model fitting (Arun's algorithm [5]). We make our modifications to the code publicly available[2].

The parameters of the proposed method were selected empirically based on synthetic data (disjoint from other experiments), and kept fixed throughout (for details, see [29]). Parameters common to all settings: probability of success $p_0 = 0.99$, number of hash tables $L = 4$; Random Grid cell size $c = 1.8t$, where the LR tolerance t (and the RANSAC threshold used) are specified separately for each experiment; Maximal number of iterations $n = 5 \times 10^6$; Hash table size of $n/10$, resulting in addressing indices of 19 bits and less than 4ms initialization time (see [29] for hash table implementation details).

3.1. 2D-homography estimation

We create a large body of 2D-homography estimation instances using the Zurich buildings data-set [38]. The data-set consists of sequences of 5 snapshots taken from different street-level viewpoints for 201 buildings. The images are typically dominated by planar facade structures and hence each pair of images in a sequence is related by a 2D homography (or perhaps more than one in the case of several planes).

We computed SIFT [32] features for each image and created sets of corresponding features for each of the 10 (ordered) pairs of images in a sequence using the VLFeat library [40]. Following [35], we generated ground-truth by running 10^7 iterations of both RANSAC and LR on each pair, and saved the highest inlier rate detected (along with the resulting homography) as the 'optimal' inlier rate for the image pair. A small set of image pairs (132 out of 2010) with very low inlier-rate was manually removed from the evaluation, since the inlier feature locations did not reside on an actual single plane in the scene, and were *very* noisy.

The 1878 resulting matching instances are challenging: many pairs have low inlier rates, that result from (i) the planar area of interest typically covering only part of each image; (ii) large viewpoint changes; (iii) large presence of repetitive patterns (e.g. windows or pillars). See Figure 5 for the distribution of inlier rates for this data-set.

We ran RANSAC and SPRT with a threshold of 8 pixels to capture the hard cases, and following [29] LR tolerance t was set to 70 pixels in the latent domain. We ran 100 independent trials of each method and summarize the results in Table 1. We arrange the image pairs into four groups according to their 'optimal' inlier rate (defined above), and the size of each group is shown at the bottom of the table. For each group we report the average, and 95-percentile of runtimes for each method. We also report each method's success rate (averaged over all pairs in the group), which is the ratio between the detected inlier-rate and the 'optimal' inlier rate.

		inlier rate range (in %)			
measure	**method**	0-10	10-20	20-40	40-100
runtime avg. (95%) (millisecs)	RANSAC	1,490 (6,594)	35 (97)	5 (10)	**6** (**9**)
	SPRT	**1,129** (4,659)	**23** (**62**)	**4** (**7**)	**6** (**9**)
	LR	1,209 (**4,626**)	32 (85)	7 (11)	9 (12)
success	RANSAC	**93.39**%	95.53%	96.28%	97.33%
	SPRT	88.57%	95.71%	96.27%	97.48%
	LR	93.07%	**95.88**%	**96.55**%	**97.59**%
# of instances		234	378	655	611

Table 1. 2**D Homography fitting on Zurich Buildings [38].** Best results are shown in bold. See text for further details.

As can be seen, SPRT and LR (modestly) accelerate RANSAC at the harder inlier rate ranges, where the overall runtime is longer. LR achieves this with no loss in accuracy, while SPRT fails on some cases in the 0-10 range.

The detailed runtime breakdowns shown in Figure 6 (left) reveals two important points that should be made here. First, in homography estimation, methods that accelerate

[2] github.com/rlit/LatentRANSAC

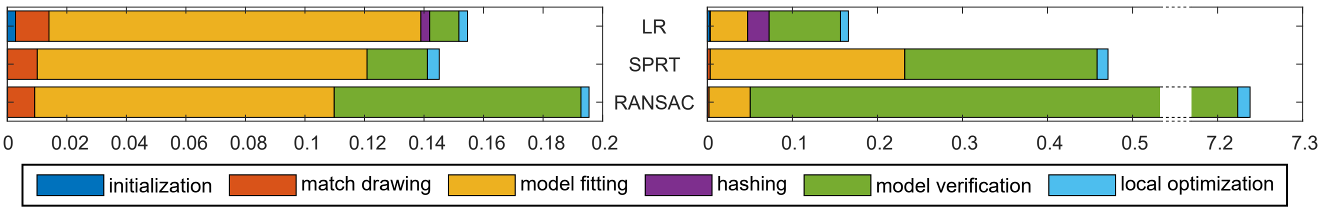

Figure 6. **Runtime breakdown per pipeline module comparing LR to SPRT and RANSAC.** These are average per-instance runtimes (in seconds) for 2D-homography estimation (**left**) and PnP estimation (**right**) taken over each of the entire data-sets used for the evaluation.

the RANSAC evaluation stage (i.e. LR and SPRT) have a relatively small potential improvement gap since the runtime of RANSAC-based homography estimation is dominated by the model fitting stage (this is not the case for the other problems we deal with, as will be seen later). Second, the improved acceleration in the lower ranges is significant when considering the overall time taken to fit the entire data-set, since the majority of time is spent on these difficult cases which are surprisingly not rare (12% and 20% of all pairs are in the 0-10 and 10-20 ranges respectively).

3.2. Perspective-n-Point (PnP) estimation

We chose to use the P3P algorithm [28] for minimal sample model fitting (for all methods) due to its good accuracy-efficiency trade off compared to other alternatives.

Putative 2D-3D matches were generated following image-to-SfM localization pipeline from [25]. We used images from [26], where vocabulary trees and queries were generated based on the train-test split therein. Even though we tested on all scenes from [26], result are presented only for the "Old Hospital" scene as it contains the best balance of moderate and challenging cases in terms of inlier rates (see Figure 5). More result are presented in [29]

We ran RANSAC and SPRT with a threshold of 2.9°, and following [29] LR with a tolerance t of 5° (in the latent domain), and the translation-to-angle ratio of the embedding was $2.1\frac{cm}{rad}$.

The results are summarized in table 2. We grouped the 182 query images (PnP instances) by increasing level of difficulty, using the ground truth inlier rates. For each of the 182 query images we ran 10 independent trials and report average and 95th percentile of the detected inlier rates. It can be seen that LR achieves more than an order of magnitude acceleration compared to RANSAC, at a comparable accuracy (as before, the success rate is the ratio between the detected and optimal inlier rates). SPRT achieves similar acceleration factors and accuracy at the higher inlier rates (above 10%). It is not as efficient or as accurate the lower range (below 10%).

In the PnP problem, the existence of fast fitting algorithms (e.g. [28]), make the verification stage the main time consumer in RANSAC. As can be seen in the time breakdown in Figure 6 (right), the costly verification time (over 95% of RANSAC time) is practically eliminated by LR.

		inlier rate range (in %)			
measure	**method**	0-10	10-20	20-40	40-100
runtime avg. (95%) (millisecs)	RANSAC	4.2e4 (1.7e5)	2,336 (5,347)	190 (471)	41 (71)
	SPRT	2,760 (1.7e4)	40 (**71**)	15 (20)	**12** (16)
	LR	**913 (4,403)**	**39** (72)	**14** (**18**)	**12** (**15**)
success	RANSAC	**95.54%**	98.13%	99.38%	99.12%
	SPRT	91.94%	**98.23%**	**99.40%**	**99.17%**
	LR	94.73%	98.14%	99.39%	99.11%
# of instances		29	27	91	35

Table 2. **PnP fitting on the OldHospital scene from PoseNet [26].** Best results are shown in bold. See text for further details.

3.3. Rigid 3D alignment

To evaluate the Rigid 3D alignment application of Latent-RANSAC, we use the registration challenge of the recent "Redwood" benchmark proposed by Choi *et al.* [12]. This dataset was generated from four synthetic 3D scenes, each divided into 52 point-cloud fragments on average. While from synthetic origin, these fragments contain high-frequency noise and low-frequency distortion that simulate scans created by consumer depth cameras.

The challenge is to perform global 3D registration between every pair of fragments of a given scene, in order to provide candidate pairs for trajectory loop closure. A correctly 'detected' pair is one for which the point clouds overlap by at least 30% and the reported transformation is sufficiently accurate (see [12] for details). The main goal in this benchmark, as stated by [12], is to achieve *high recall* while relying on a post-process to later remove false-matches.

Aside from the benchmark, Choi *et al.* [12] present a simple extension (CZK) to the Point-Cloud-Library (PCL) [23] implementation of [37]. The method of CZK showed state-of-the-art performance, while comparing to previous methods like OpenCV [18], 4PCS [3] and its extension super4PCS [33]. Fast Global Registration (FGR) [45] is a recent novel optimization process presented by Zhou *et al.*, which achieves an order of magnitude runtime acceleration on this dataset, at a competitive recall-precision performance. They perform the costly nearest-neighbor (NN) search only once (unlike previous methods which use them in their inner loop), while introducing several fast and simple methods to filter false matches.

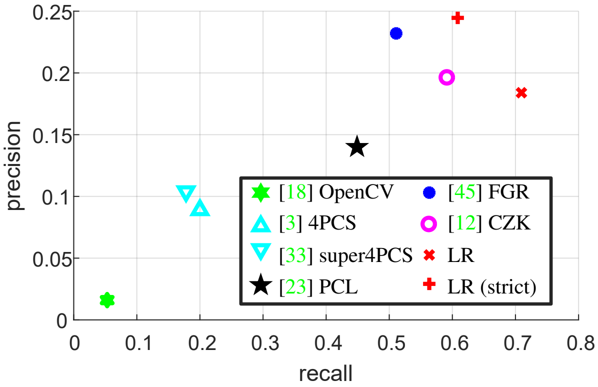

Figure 7. **Performance on the "redwood" benchmark [12].** Our method achieves state-of-the-art *recall* in the standard setting (marked by a red 'x'), while using a stricter threshold (marked by a red '+') dominates all previous result in both precision and recall. See description in the text for further details.

We chose to follow [12, 23, 45] and feed our framework with putative matches based on FPFH features [37]. Following [29], we use LE tolerance t of 24cm in the latent space, and angle-to-translation ratio of $3.6\frac{rad}{cm}$ for the embedding. Inspired by FGR we perform the NN search only once. We then only apply one single filter (also used in [12, 45]), an approximate congruency validation on each minimal sample drawn.

Figure 7 shows a comparison of Latent RANSAC to other results reported in [45]. Our method clearly achieves the highest recall value (the main goal), at a precision slightly below that of CZK. Furthermore, we are able to dominate all previous results in precision and recall simultaneously by using a slightly stricter setting, when reporting only pairs with overlap of over 40% rather than 30%.

We attribute our high performance mainly to the fact that we perform almost no filtering to the putative matches, such as the bidirectional search and tuple filtering done in [45], normal-agreement in [12, 23] or the drawing of a non-minimal set of 4 matches in [12]. Using this "naive" nearest neighbor FPFH feature matching avoids filtering of true correspondences (enabling higher recall), but this comes at the cost of some very low inlier-rates, as can be seen in Figure 5. Our algorithm is able to deal with such inlier rates successfully (and efficiently), as was shown in the other experiments of this section and in Figure 1.

Another attractive property of our method in this benchmark is its runtime, presented in Table 3. Our runtime is close to that of FGR, which we outperform significantly in terms of recall. Note, however, that our method is actually faster than FGR whenever the inlier rate is above 5%, as the number of iterations given by (5) is very low. Verification consumes a considerable part of the runtime of methods like [12, 23], while we perform the costly overlap verification only upon the detection of a collision (13.3 times per run on average). Additionally, we perform overlap calculation only once as done in FGR.

method	PCL [23]	CZK [12]	FGR [45]	LR
avg. time (sec)	3.8	7.5	**0.21**	0.40

Table 3. **Average runtimes on the "redwood dataset"**, excluding normals and FPFH [37] calculation time which are 24ms and 300ms on average, respectively. A breakdown of our method's timing includes 84ms for feature matching, 305ms for the latent RANSAC pipeline, and 13ms for overlap calculation.

4. Future work

In this work we presented Latent-RANSAC: a novel speed-up of the hypothesis handling stage of the RANSAC pipeline. We have shown its advantages on challenging matching problems, that include very low inlier rates, in the domains of homography estimation, camera localization and rigid 3D motion estimation.

Latent-RANSAC has the potential to be extended to additional domains. Of particular interest is finding an appropriate parametrization of the more challenging fundamental matrix domain, which is classically tackled using RANSAC.

The good results that Latent-RANSAC achieves on the "Redwood" benchmark come to show the advantage of being able to handle highly corrupted "raw"s data (over 60% of the fragment pairs have under 10% inlier rate). This is since the alternative of filtering the data to reduce the rate of outliers comes at the risk of loss of informative data. The challenge, however, remains to do so efficiently, especially for search spaces of high dimensionality.

instance	*measure*	USAC	SPRT	LR
'Old Hospital'	inlier rate (%)	2.9±0.1	0.0±0.0	2.9±0.1
seq 8 frame 12	Sampson err.	0.27±0.16	failed	0.025±0.12
#matches: 9,917	#samples	.19±.012	5.0±0	.25±.034
	#fitting	.19±.011	5.0±0	.26±.034
	#verification	.62±.037	16.4±.002	.011±.002
	runtime [sec]	271.0±20.8	49.8±2.3	6.9±1.3
'building 187'	inlier rate (%)	4.2±0.1	4.2±1.3	4.2±0.1
views 3,5	Sampson err.	0.4±0.2	1.8±1.9	0.6±0.0
#matches: 622	#samples	1.9±.31	2.7±.93	2.7±.21
	#fitting	.19±.030	.27±.092	.26±.021
	#verification	.19±.30	.27±.092	.025±.003
	runtime [sec]	2.83±0.65	2.29±0.88	2.24±0.23

Table 4. **Detailed example results.** All measures are reported in terms of median ± std. Numbers of samples as well as fitting and verification invocations are in millions (10^6). **Top**: 10 iterations of PnP estimation in the PoseNet data [26], error in radians. **Bottom**: 100 iterations of Homography estimation in the Zurich Buildings (ZuBuD) [38] data. See Figure 1 for a detailed rigid-3d estimation example.

References

[1] D. Aiger, H. Kaplan, and M. Sharir. Reporting neighbors in high-dimensional euclidean space. *SIAM Journal on Computing*, 43(4):1363–1395, 2014. 5

[2] D. Aiger, E. Kokiopoulou, and E. Rivlin. Random grids: Fast approximate nearest neighbors and range searching for image search. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3471–3478, 2013. 4, 5

[3] D. Aiger, N. J. Mitra, and D. Cohen-Or. 4-points congruent sets for robust pairwise surface registration. *ACM Transactions on Graphics (TOG)*, 27(3):85, 2008. 7, 8

[4] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468. IEEE, 2006. 4

[5] 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*, (5):698–700, 1987. 6

[6] S. Baker, A. Datta, and T. Kanade. Parameterizing homographies. *Technical Report CMU-RI-TR-06-11*, 2006. 3

[7] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981. 2

[8] D. Campbell, L. Petersson, L. Kneip, and H. Li. Globally-optimal inlier set maximisation for simultaneous camera pose and feature correspondence. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1, 3

[9] D. P. Capel. An effective bail-out test for ransac consensus scoring. In *BMVC*, 2005. 2

[10] T.-J. Chin, P. Purkait, A. Eriksson, and D. Suter. Efficient globally optimal consensus maximisation with tree search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2413–2421, 2015. 1

[11] J. Choi and G. Medioni. Starsac: Stable random sample consensus for parameter estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 675–682. IEEE, 2009. 2

[12] S. Choi, Q.-Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 6, 7, 8

[13] O. Chum and J. Matas. Matching with prosac-progressive sample consensus. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 220–226. IEEE, 2005. 2

[14] O. Chum and J. Matas. Optimal randomized ransac. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1472–1482, 2008. 2, 6

[15] O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. In *Pattern Recognition*, pages 236–243. Springer, 2003. 2, 6

[16] R. J. Den Holl and E. A. Hanjalic. A combined ransac-hough transform algorithm for fundamental matrix estimation. In *in 18th British Machine Vision Conference. University of*. Citeseer, 2007. 2

[17] D. DeTone, T. Malisiewicz, and A. Rabinovich. Deep image homography estimation. *arXiv preprint arXiv:1606.03798*, 2016. 3

[18] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 998–1005. Ieee, 2010. 7, 8

[19] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972. 2

[20] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 1, 2

[21] V. Fragoso, P. Sen, S. Rodriguez, and M. Turk. Evsac: accelerating hypotheses generation by modeling matching scores with extreme value theory. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2472–2479. IEEE, 2013. 2

[22] V. Fragoso, C. Sweeney, P. Sen, and M. Turk. Ansac: Adaptive non-minimal sample and consensus. In *BMVC*, page arXiv:1709.09559, 2017. 2

[23] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke. Registration with the point cloud library: A modular framework for aligning in 3-d. *IEEE Robotics & Automation Magazine*, 22(4):110–124, 2015. 7, 8

[24] P. V. Hough. Machine analysis of bubble chamber pictures. In *International conference on high energy accelerators and instrumentation*, volume 73, page 2, 1959. 2

[25] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2599–2606. IEEE, 2009. 7

[26] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2938–2946. IEEE, 2015. 6, 7, 8

[27] L. Kneip and P. Furgale. Opengv: A unified and generalized approach to real-time calibrated geometric vision. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1–8. IEEE, 2014. 6

[28] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2969–2976. IEEE, 2011. 6, 7

[29] S. Korman and R. Litman. Latent ransac supplementary materials. `arxiv.org/src/1802.07045v2/anc/supp.pdf`, 2018. 6, 7, 8

[30] K. Lebeda, J. Matas, and O. Chum. Fixing the locally optimized ransac–full experimental evaluation. In *British machine vision conference*, pages 1–11. Citeseer, 2012. 2

[31] R. Litman, S. Korman, A. Bronstein, and S. Avidan. Inverting ransac: Global model detection via inlier rate estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5243–5251, 2015. 3

[32] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 6

[33] N. Mellado, D. Aiger, and N. J. Mitra. Super 4pcs fast global pointcloud registration via smart indexing. In *Computer Graphics Forum*, volume 33, pages 205–215. Wiley Online Library, 2014. 7, 8

[34] C. Olsson, O. Enqvist, and F. Kahl. A polynomial-time bound for matching and registration with outliers. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 1

[35] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J.-M. Frahm. Usac: a universal framework for random sample consensus. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):2022–2038, 2013. 2, 6

[36] R. Raguram and J.-M. Frahm. Recon: Scale-adaptive robust estimation via residual consensus. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1299–1306. IEEE, 2011. 2

[37] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009. 7, 8

[38] H. Shao, T. Svoboda, and L. Van Gool. Zubud - zurich buildings database for image based recognition. *Computer Vision Lab, Swiss Federal Institute of Technology, Switzerland, Tech. Rep*, 260:20, 2003. 6, 8

[39] B. Tordoff and D. W. Murray. Guided sampling and consensus for motion estimation. In *European conference on computer vision*, pages 82–96. Springer, 2002. 2

[40] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable librar of computer vision algorithms, 2008. 6

[41] A. Wald. *Sequential analysis*. Courier Corporation, 1973. 2

[42] L. Xu, E. Oja, and P. Kultanen. A new curve detection method: randomized hough transform (rht). *Pattern recognition letters*, 11(5):331–338, 1990. 2, 4

[43] J. Yang, H. Li, and Y. Jia. Go-icp: Solving 3d registration efficiently and globally optimally. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1457–1464, 2013. 3, 4

[44] Y. Zheng, S. Sugimoto, and M. Okutomi. Deterministically maximizing feasible subsystem for robust model fitting with unit norm constraint. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1825–1832. IEEE, 2011. 1

[45] Q.-Y. Zhou, J. Park, and V. Koltun. Fast global registration. In *European Conference on Computer Vision*, pages 766–782. Springer, 2016. 1, 7, 8