Geometric Hashing: Rehashing for Bayesian Voting

I. Blayvas, R. Goldenberg, M. Lifshits, E. Rivlin, and M. Rudzsky

Computer Science Department,
Technion, Haifa, Israel

Abstract. Geometric hashing is a model-based recognition technique based on matching of transformation-invariant object representations stored in a hash table. Today it is widely used as an object recognition method in numerous applications. In the last decade a number of enhancements have been suggested to the basic method improving its performance and reliability. Here we consider two of them. One is rehashing, dealing with the problem of non-uniform occupancy of hash bins, and the other is Bayesian approach improving recognition rate in presence of noise. The latter uses an altered matching scheme, where the search is performed over an error dependent voting region around the query. In this paper we propose a scheme that takes the best from both worlds, yielding a hash table with a uniform size of voting regions. This allows to improve the geometric hashing computational performance by minimizing the hash table size and the number of bins accessed, while maintaining optimal recognition rate. Alternatively, the proposed method can be used in classical single bin voting to improve recognition rate.
1 Introduction

Geometric hashing introduced by Lamdan and Wolfson [3], based on the indexing approach by Schwartz and Sharir [8], solves the problem of object recognition formulated in the following way. There is a set of predefined geometric models \( M_1, \ldots, M_n \) and a query image \( Q \), formed from one of the models. The task is to find the corresponding model \( M_i \) given the query image \( Q \).

It is assumed that the models are defined by a set of geometric features (e.g. corner points) and that the same features can be extracted from the query image. The set of possible transformations a model can undergo to form the query image is known. One way to make feature points invariant under a broad class of transformations (e.g. affine, similarity, etc.) is to represent them in the coordinate frame formed from the points themselves. E.g. for an affine transformation we may arbitrarily choose three object points to form a basis and describe the rest of the features in this coordinate frame. As there are multiple ways to choose a basis, we are faced with a combinatorial problem of finding the right one to match a model to a query.

Geometric hashing copes with this problem by shifting the computational burden to the off-line learning stage. Instead of going over all possible bases for the query image, all possible model representations are prepared in advance and stored in a hash table for efficient access. Thus, a query image projected onto an arbitrarily chosen basis has a matching model representation already stored in the hash table.

Ideally, for every feature point of the query image there is a single model point in the corresponding hash table bin. In practice, features generated by other models can fall into the same bin or even coincide. To deal with this problem one may suggest to reduce the bin size. Unfortunately, the feature points are non-uniformly distributed over the hash table. Therefore, for any bin size there will be either overpopulated or empty bins.

Another problem is that the uncertainty in feature point position caused by image noise shifts it away from the corresponding model feature. This could be solved by looking for the matching model in a certain neighborhood of the image feature - voting region. The problem is further complicated by the fact that a position independent noise in the image becomes position dependent in the hash table.

These problems received a lot of attention in the computer vision community over the last 15 years [1–8]. For a comprehensive review we refer the reader to [9]. Here we are interested in two particular approaches. The first one - rehashing - tackles the problem of nonuniform feature distribution by re-mapping [6]. The other uses Bayesian approach to optimize the recognition rate in presence of noise [7].

In this paper we reconsider the rehashing approach for the Bayesian framework. We show that in order to optimize both the recognition rate and the computational performance the rehashing is to be redesigned to equalize voting regions rather than feature density, as suggested before [4,6]. We derive
the rehashing scheme for the case of similarity transformations and evaluate its performance in a series of experiments.

The rest of this paper is organized as follows. Section 2 discusses the optimality criteria for geometric hashing. In section 3 we describe the Bayesian approach yielding an optimal recognition voting scheme. We derive the uniform voting region rehashing scheme in section 4 and present the experiments confirming our theoretical results in section 5.

2 Optimality criteria

Before getting into a discussion about geometric hashing optimization, one has to define the evaluation criteria, i.e. the set of desired properties. Clearly, the most important ones are recognition rate and time and space complexity.

Let us see what factors influence the recognition rate. A query feature point votes for all models in its voting region. False alarms are then caused by hash entries accidently falling within the voting region. Therefore, the recognition rate can be improved either by making voting regions smaller or reducing the entries density in the hash table. Of course, the voting region cannot be made arbitrarily small, as in this case there is a good chance of missing the correct model representative due to a noise induced position error.

![Fig. 1. Circular voting region covered by large (a) and small (b) hash bins. The same voting region can be covered by 4 large or 12 small bins. In the first case (a) there are 10 visited model points outside the voting region (empty dots), whereas in the second (b) there are only 4 of them. On the other hand, 4 out of 12 small covering bins are empty.](image)

Speaking of computational performance, it is generally assumed that only the online stage of the algorithm really counts. There is a trivial lower bound on the number of operations given by the number of model points within the voting regions - all of them are to be visited and voted for. The upper bound, on the other hand, depends on data organization. For a simple classical hash
table, the run time is bounded by the number of visited hash bins plus the number of entries in these bins. From this perspective, one would be interested in covering the whole voting region by small number of bins. This can be achieved by increasing the bin size, but in this case the number of non-relevant hash entries (those outside the voting region) visited during the search grows. Making bins smaller, on the other hand, results in a large number of empty hash bins, thus increasing the run time and space complexity, as the hash table size grows (see Figure 1).

3 Bayesian approach

In section 2 we saw that the voting region size controls the tradeoff between the number of false votes and the chance to miss the correct model. Therefore, choosing an optimal voting region size is crucial for the recognition rate. Rigoutsos and Hummel [7] derive this size for the case of similarity transformations using the maximum likelihood approach, as described below.

Before proceeding, we need to introduce some formal notation.

Let \( \{p_1, \ldots, p_{N_k}\} \) be the feature points of model \( M_k \). For similarity transformations the coordinate frame for invariant representation is uniquely defined by two points. Then there are \( \binom{N_k}{2} \) such bases for the model \( M_k \). For each one of these bases \( B_{\mu\nu} = \{p_\mu, p_\nu\} \) and for every remaining model point \( p_i \) the invariant representation \( h_i \) is computed and the entry \( \{M_k, B_{\mu\nu}, h_i\} \) is stored at the hash table bin indexed by \( h_i \). The invariant entries generated by a query \( Q \) are denoted by \( F_Q = \{q_1, \ldots, q_{N_Q}\} \), and their corresponding hash coordinates are \( \{(u_i, v_i); i = 1, \ldots, N_Q\} \).

Assuming the independence of \( \{q_i\} \), the probability of model \( M_k \) given the query \( Q \) is

\[
P(M_k|Q) \sim P(M_k) \prod_{q_i \in Q} P(M_k|q_i)\prod_{q_i \in Q} P(M_k).
\]

Applying the Bayes theorem we get

\[
P(M_k|Q) \sim P(M_k) \prod_{q_i \in Q} \frac{P(M_k|q_i)}{P(q_i)},
\]

where \( P(q_i|M_k) \) is the probability of “hashing” to the location \( \{u_i, v_i\} \), under the assumption that model \( M_k \) is present, while \( P(q_i) \) is the probability without this assumption. The ratio \( \frac{P(q_i|M_k)}{P(q_i)} \), measuring the factor by which the probability \( P(M_k|Q) \) changes due to the feature point \( q_i \), can be reformulated in terms of density functions:

\[
\frac{P(q_i|M_k)}{P(q_i)} = \frac{\frac{N_Q}{N_Q} \cdot g(q_i) + \frac{N_k}{N_Q} \cdot f(q_i)}{g(q_i)} = 1 + \frac{N_k}{N_Q} \cdot \frac{f(q_i) - g(q_i)}{g(q_i)},
\]

where \( g(q) \equiv g(u, v) \) is the density of invariant entries in the hash space, \( f(u, v) \) is the density of query points corresponding to the model \( M_k \), \( N_Q \) is the number
of query points and \( N_k \) is the number of points in the model \( M_k \). Assuming all model points present in \( Q \), \( \frac{N_k}{N_Q} \) is the probability for \( q_i \) to be one of the model points and \( \frac{N_Q - N_k}{N_Q} \) to be a clatter point.

Actually, Equation (3) expresses the relationship between the Bayesian approach and the geometric hashing voting scheme. Query points satisfying \( f(q) > g(q) \) add to the value of \( P(M_k|Q) \) and therefore vote for the model. The rest of the points decrease \( P(M_k|Q) \) and are not counted as non-relevant to the \( M_k \) hypothesis. The votes are to be weighted according to their contribution to \( P(M_k|Q) \). For numerical stability considerations it is convenient to take the logarithm of Equation (2). Then the query point \( q_i \) contributes

\[
W(q_i) = \ln \left( 1 + \frac{N_k}{N_Q} \cdot \frac{f(q_i) - g(q_i)}{g(q_i)} \right)
\]

(4)
to the votes of \( M_k \).

It was shown, that for similarity transformations [4]

\[
g(u, v) = \frac{12}{\pi} \frac{1}{(4(u^2 + v^2) + 3)^2},
\]

(5)
and

\[
f(u, v) = \sum_{i=1}^{N_k} \frac{1}{2\pi \sqrt{|C_i^{-1}|}} \exp \left( -\frac{1}{2} \left( u - x_i, v - y_i \right) \cdot C_i \cdot \left( u - x_i, v - y_i \right)^T \right),
\]

(6)
where the sum runs over the model points \((x_i, y_i)\),

\[
C_i = \frac{2\|p_v - p_\mu\|^2}{(4(x_i^2 + y_i^2) + 3) \cdot \sigma^2} \cdot I,
\]

(7)
is the covariance matrix at \((x_i, y_i)\), \(\sigma\) is the variance of the model feature position, and \(p_v, p_\mu\) are the model points forming the basis [5, 6]. By substituting (5-7) into the inequality \( f(q) > g(q) \) and solving for \( \rho(u, v) \equiv \|(x, y) - (u, v)\| \) we obtain the optimal radius for the voting region

\[
\rho(u, v) = \epsilon \sqrt{\left(4(u^2 + v^2) + 3\right) \cdot \ln \left( \frac{4(u^2 + v^2) + 3}{12\epsilon^2} \right)},
\]

(8)
where \(\epsilon = \frac{\sigma}{\|p_v - p_\mu\|}\). To simplify the derivation we assume that model points are distant enough from each other, so that \( f(u, v) \) is defined by the Gaussian centered at the closest model point \((x_i, y_i)\), while the rest is negligibly small. The covariance matrix of this Gaussian distribution can be approximated by \(C(u, v)\).

### 4 Rehashing for Bayesian recognition

The Bayesian approach, presented in the previous section optimizes the recognition rate. The other parameter we wish to optimize as discussed in section 2 is the computational efficiency.
As we have already mentioned, the hash table bin size should be chosen to minimize the expected access time to the entries within the voting region. This could be achieved by making bins proportional to voting region. However, from equation (8), the voting region size varies significantly over the hash space. Therefore, there is no fixed optimal bin size for the whole hash table.

To overcome this problem we can re-map the hash entries to make the voting region size constant throughout the hash table. Actually the technique of re-mapping, also known as rehashing, is not new in the field and was used to uniformly distribute hash entries over the table [6]. This makes sense in the classical geometric hashing approach, where the voting is preformed in a single bin. However, in the Bayesian framework such a re-mapping is not optimal.

Formally, we need to define a mapping \( T : (u, v) \rightarrow (u', v') \), such that \( \rho'(u', v') = 1 \). Then
\[
\rho'(u', v') = \rho(u, v)J,
\]
where
\[
J = \begin{vmatrix}
\frac{\partial u'}{\partial u} & \frac{\partial u'}{\partial v} \\
\frac{\partial v'}{\partial u} & \frac{\partial v'}{\partial v}
\end{vmatrix}
\]
is the Jacobian of \( T \). In other words, we are looking for \( T \) solving the following differential equation
\[
det \begin{pmatrix}
\frac{\partial u'}{\partial u} & \frac{\partial u'}{\partial v} \\
\frac{\partial v'}{\partial u} & \frac{\partial v'}{\partial v}
\end{pmatrix} = \frac{1}{\rho(u, v)}.
\]
We find that the transformation \( T \) can be well approximated by
\[
\begin{cases}
  u' = \frac{\pi^2}{\sqrt{r^2 + e^2}} \ln (1 + r) \\
v' = \arctan \left( \frac{v}{u} \right)
\end{cases}
\]
where \( r = \sqrt{u^2 + v^2} \).

Fig. 2. Voting regions: (a) without rehashing, (b) after uniform density rehashing, and (c) after uniform voting region rehashing.
Figure 2 presents (a) the initial voting regions distribution described by Equation (8) and voting regions transformed by (b) uniform density rehashing and (c) by our rehashing scheme. The varying voting region size in cases (a) and (b) lowers the recognition rate for the classical single-bin voting and the computational performance for the Bayesian schemes.

5 Experimental Results

To verify the theoretical results we performed a number of simulations and compared several variations of geometric hashing algorithms.

We experimented with two voting schemes: (a) classical single bin voting - query point votes for all the entries found in the hash bin it falls in, and (b) Bayesian approach - query point votes for all the points within its voting region. For each one of the voting schemes three rehashing options were evaluated: (i) no rehashing, (ii) uniform density rehashing and (iii) the uniform voting region rehashing suggested here. For each one of the six voting/rehashing combinations we tested nine different hash bin sizes.

We used five models, each consisting of 20 points randomly scattered over a unit square. 500 queries were formed by adding a Gaussian noise with $\sigma = 0.025$ to the models and randomly choosing a two-point basis. Small bases ($\|p_\nu - p_\mu\| < 0.2$) were discarded as unreliable. Recognition rate was taken as the relative number of queries for which the correct model/basis combination received the maximal voting score. The computational performance was estimated by counting the number of the accessed hash entries.

Figure 3 presents the recognition rate as a function of the bin size. Three graphs for the single bin voting are shown (no rehashing, uniform density and uniform voting region rehashings). The fourth graph corresponds to the Bayesian voting, for which the recognition rate does not depend on the rehashing scheme or the bin size. As expected, the Bayesian voting yields the best recognition rate. Both rehashing schemes improve the recognition rate for the single bin voting. As one can see, the proposed uniform voting region rehashing outperforms the uniform density rehashing for any bin size.

Obviously, for unreasonably small bin sizes rehashing schemes become ineffective, since only a small part of the voting region is covered by the bin, and therefore most of the bins miss the relevant vote. On the other hand, without rehashing, bins in the densely populated areas still contain the relevant vote.

Figure 4 presents the number of the accessed entries for the single bin voting scheme. This can be seen as an implementation independent indicator of the time complexity. As expected, the uniform density rehashing has the best performance, since any non-uniformity in distribution results in a higher probability to hash into a more populated bin. The proposed uniform voting region rehashing, as a side effect, also distributes hash entries more uniformly. Therefore, the number of the accessed entries in this case decreases as well.
Fig. 3. Recognition rates in classical scheme for three different rehashing schemes and Bayesian scheme.

Fig. 4. Number of accessed entries for the single bin voting scheme.
6 Summary

In this paper we presented a new rehashing scheme for geometric hashing. The scheme is consistent with the Bayesian voting approach and optimizes its computational performance. In addition the scheme can be used for the traditional single bin geometric hashing yielding a higher recognition rate compared to the known uniform density rehashing. Our theoretical results were confirmed by a comparative experimental analysis. Simulations demonstrate the advantage of the proposed method.

References