# IPJC: The Incremental Posterior Joint Compatibility Test for Fast Feature Cloud Matching

Edwin B. Olson[1] and Yangming Li[2]

***Abstract*— One of the fundamental challenges in robotics is data-association: determining which sensor observations correspond to the same physical object. A common approach is to consider groups of observations simultaneously: a constellation of observations can be significantly less ambiguous than the observations considered individually. The Joint Compatibility Branch and Bound (JCBB) test is the gold standard method for these data association problems. But its computational complexity and its sensitivity to non-linearities limit its practical usefulness.**

**We propose the Incremental Posterior Joint Compatibility (IPJC) test. While equivalent to JCBB on linear problems, it is significantly more accurate on non-linear problems. When used for feature-cloud matching (an important special case), IPJC is also dramatically faster than JCBB. We demonstrate the advantages of IPJC over JCBB and other commonly-used methods on both synthetic and real-world datasets.**

***Index Terms*— Data association, joint compatibility test, SLAM**

## I. Introduction

Data association is the problem of determining which observations correspond to the same object. It is at the core of the Simultaneous Localization and Mapping (SLAM) problem and visual navigation: it is only by re-observing a landmark that a map becomes over-constrained and therefore more robust to the errors associated with any single observation.

SLAM systems are often described in terms of the two "halves" of the problem: the front-end performs the sensor processing and data association, while the back-end computes the maximum-likelihood map subject to the observations and data-associations output by the front-end. In recent years, the raw computational performance of back-ends has increased dramatically: maps with millions of landmarks and observations can be optimized [1].

While back-end systems are now very fast, the quality of their output is entirely dependent on the accuracy of the front-end. In particular, an incorrect data association (in which the front-end erroneously asserts that two physically-distinct landmarks are in fact the same landmark) forces the back-end to distort the map to bring those two landmarks closer together. Even a single data-association error can lead to divergence of the entire map.

Consequently, the quality of a front-end system has an enormous impact on the quality of the resulting map. Too many loop closures (i.e., false positives) lead to catastrophic failures, while too few loop closures (false negatives) lead to a less-constrained map of lower overall quality.

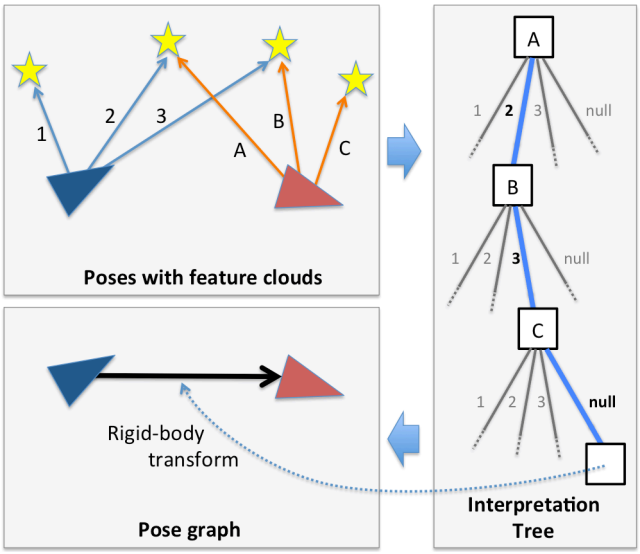



Fig. 1. IPJC Overview. A robot observes "feature clouds" from two different poses (top left), and IPJC matches them by searching a tree and computing a compatibility cost. The rigid-body transformation that results from the matching process can be used in a pose-graph SLAM formulation. IPJC is similar to JCBB, but when applied to feature cloud matching, produces better results in less time.

A common approach to improving the quality of data-association systems is to consider multiple observations as a set. A reasonable analogy is that it is difficult to recognize a star given an image of it, but recognizing a constellation is much easier and less error-prone. When matching groups of features, it is critical to consider the correlations between measurements. In general, the set of observations will not match perfectly with the prior estimates of the landmark locations. Due to the correlations between these observations, some misalignments are more likely than others. For example, suppose an image of a constellation of stars is taken. The individual positions of the stars in the image are highly correlated: they all depend on where the camera was pointing. If all of the stars appeared to be shifted uniformly with respect to their a priori estimated positions, the errors could be easily explained in terms of a camera pointing error. On the other hand, if the stars were shifted randomly with respect to their a priori estimated positions, one might instead conclude that the image is of a different set of stars. In other words, proper consideration of the correlations between observations can have a significant effect on the data

[1] Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 ebolson@umich.edu

[2] Institute of Intelligence Machines, Chinese Academy of Sciences, Hefei, Anhui, 230031 ymli@iim.ac.cn

association process.

The gold standard method is the Joint Compatibility Branch and Bound (JCBB) test [2], which searches for the largest set of data associations subject to a bound on the $\chi^2$ error. Conceptually, JCBB builds an "interpretation tree"; at each level of the tree, an observation is associated with one of the landmarks in the map (or to a "null" hypothesis representing the possibility that no landmark matches the observation.) A path from the root of the tree to any node encodes a set of data-associations, and JCBB explicitly computes a cost related to the probability of that set of data-associations.

The computational cost of JCBB can be substantial; at every node in the tree, the joint compatibility must be computed. Even when computed in a clever incremental fashion, this involves an operation of cost $O(m^2)$ at level $m$ in the tree. This cost can be prohibitive in some applications and has led to a number of alternative approaches.

Several authors have suggested faster but less probabilistically-motivated methods for validating data-association hypotheses. One trend has been to implicitly identify groups of compatible hypotheses by considering their pairwise-consistency; this can be viewed as a max-clique [3] or spectral graph partitioning [4] problem. Finding loop closures, a common task in pose-based SLAM, is an application of data-association. A recurring idea is to look for sequences of loop closures that form a closed topological loop: the composition of the loop closures in the loop should approximately be the identity matrix [5], [6], [7].

In addition to high computational cost, JCBB is sensitive to non-linearities. JCBB estimates the joint compatibility by linearizing around the prior and considering the entire set of data associations as a single large linear update. However, more accurate results could generally be obtained by actually computing the posterior after each observation; this results in a better estimate of the posterior and will generally improve the linearization point used to estimate the compatibility of successive data association pairings.

Many data association algorithms target the case where the location of the landmarks is explicitly estimated—i.e., where the state vector is enlarged to contain the position of each landmark. Alternatively, as seen in pose-based SLAM algorithms and in many camera-based applications, the landmarks are *not* added to the state vector. Instead, the motion between two poses is found by matching the feature observations between those two poses. Each pose records a cloud of features observed from that pose, and the problem becomes one of "feature cloud" matching (see Fig. 1). This approach is often used when each pose observes a large number of landmarks: adding all of these landmarks to the state vector can quickly tax even very fast back-end systems. Systems using cameras or 3D LIDAR sensors can extract hundreds of features from a single robot pose, for example.

In this paper, we propose a new data association method that, like JCBB, is probabilistically rigorous. However, it provides better accuracy in non-linear settings and, in the case of feature cloud matching, dramatic runtime speedups. The contributions of this paper are:

- We propose a posterior-based data association test, motivate it in terms of the $\chi^2$ of a least-squares optimization, and show that it is equivalent to JCBB. This algorithm, Posterior Joint Compatibility (PJC), serves as the basis for the remainder of our algorithms.
- We propose the Incremental Posterior Joint Compatibility (IPJC) test, which exploits the probabilistic structure of feature cloud matching. This method is both more accurate in non-linear settings and dramatically faster than JCBB. We further show how to accelerate the process further, leading to the IPJC-Fast algorithm.
- We demonstrate our proposed methods along side JCBB, RANSAC, and SCNN on a range of synthetic and real-world problems. This data supports our claim that IPJC and IPJC-Fast out-perform other methods.

## II. A REVIEW OF JCBB

Our method is similar in most respects to JCBB [2]: given a set of $m$ observations of $n$ features, we search an "interpretation tree". This tree has $m$ levels, and at each level of the tree, we consider $n+1$ possible data-associations for the $m^{th}$ observation. (Each observation could match any of the $n$ landmarks, or could match none of them.) A path from the root to any node represents a set of data-associations. Our goal is to find a "good" data-association for every observation.

How is the "goodness" of a set of data-associations evaluated? Given a state estimate $x$ with covariance $P$, we use a domain-specific sensor model (assumed known) to compute predicted observations $\hat{z}$. Given the matrix $H$ of partial derivatives of $x$ with respect to $\hat{z}$, the uncertainty of the predicted observations due to our uncertainty of $x$ is simply $HPH^T$.

We assume that our actual observations $z$ are contaminated by noise with covariance $V$, and that the matrix of partial derivatives of the noise variables with respect to $z$ is $G$. The uncertainty of the actual observations due to this underlying sensor noise is simply $GVG^T$. The combined uncertainty of our prior and observation is given by:

$$C = HPH^T + GVG^T \tag{1}$$

The discrepancy between our actual and predicted observations is $e = z - \hat{z}$. We can now write the cost function used by JCBB as a Mahalanobis distance:

$$\chi^2 = e^T C^{-1} e \tag{2}$$

In principle, we might wish to identify the maximum likelihood set of data-associations (or equivalently, the set of data associations with the minimum Mahalanobis distance). To compute this, we would need to know the likelihood of an observation not matching any landmark— a quantity typically not known. Instead, JCBB searches for the largest set of non-null data-associations such that the Mahalanobis distance is less than a threshold. This threshold is typically

expressed in terms of the $\chi^2$ distribution for an appropriate number of degrees of freedom.

A naive approach would be to consider each leaf of the tree, compute its Mahalanobis distance, and select the best. However, the interpretation tree is quite large: it has $(n+1)^m$ leaves. Fortunately, the search space can be pruned by employing the branch-and-bound method. The key idea is that the Mahalanobis distance can be computed for any partial set of data-associations. Since the Mahalanobis distance can only increase with additional data associations, this serves as an admissible lower-bound for all of the node's children. Consequently, we can prune any sub-tree that could not be better than the best-known solution. A more careful description of JCBB can be found in [2].

The major computational cost in JCBB is to the cubic cost of inverting the $C$ matrix in Eqn. 2. As pointed out by the original JCBB paper, the inverse of $C$ at level $m$ can be computed in terms of the inverse of $C$ at level $m-1$, reducing the complexity to quadratic. Even with this optimization, the cost of repeatedly evaluating the Mahalanobis distance quickly becomes a bottleneck— particularly if there are a large number of observations.

## III. Proposed Method

### A. Posterior Joint Compatibility

We begin by deriving a different way of writing the Mahalanobis cost function used by JCBB. Given a putative set of data-associations, suppose we compute the posterior value of $x$, which we denote as $x^+$. (For clarity, we will denote the prior value of $x$ as $x^-$.) This posterior can be computed in a variety of ways, including Extended Kalman Filtering [8], non-linear optimization [9], etc. In this work, we use the Iterated Extended Kalman Filter [10].

Suppose that we update the state estimate using the Extended Kalman Filter as follows:

$$\begin{aligned} C &= HPH^T + GVG^T && (3)\\ K &= PH^TC^{-1} && (4)\\ e &= z - \hat{z} && (5)\\ d &= Ke && (6)\\ x^+ &= x^- + d && (7) \end{aligned}$$

We can compute the $\chi^2$ of the posterior as the sum of the $\chi^2$ of the observations and the prior evaluated at $x^+$. Note that the posterior residual for the observations is not $e$ (because $\hat{z}$ reflects the prior estimate of $z$); due to the change in our state estimate, the posterior observation residual becomes: $z - (\hat{z} + Hd) = e - Hd$.

In summary, the posterior $\chi^2$ can be written as:

$$\chi^2 = (e-Hd)^T(GVG^T)^{-1}(e-Hd) + d^TP^{-1}d \quad (8)$$

We now show that this expression is equivalent to the one used by JCBB. To see this, we begin by substituting $d = Ke$:

$$\chi^2 = (e-HKe)^T(GVG^T)^{-1}(e-HKe) + (Ke)^TP^{-1}(Ke) \quad (9)$$

And now factoring out $e^T$ to the left and $e$ to the right:

$$\chi^2 = e^T\left[(I-HK)^T(GVG^T)^{-1}(I-HK) + K^TP^{-1}K\right]e \quad (10)$$

We'll now consider the cost function used by JCBB, showing that it can be simplified to the same expression as Eqn. 10. Recall that the Mahalanobis distance used by JCBB can be written as $\chi^2 = e^TC^{-1}e$. Let us begin by focusing on the inner term $C^{-1}$.

$$\begin{aligned} & C^{-1} && (11)\\ & \left[I + K^TH^T - (HK)^T\right]C^{-1} && (12)\\ & \left[K^TH^T + (I-HK)^T(GVG^T)^{-1}(GVG^T)\right]C^{-1} && (13)\\ & \left[K^TH^T + (I-HK)^T(GVG^T)^{-1}(C-HPH^T)\right]C^{-1} && (14)\\ & \left[K^T(P^{-1}PH^TC^{-1}C) + (I-HK)^T(GVG^T)^{-1}(I-HK)C\right]C^{-1} && (15)\\ & \left[K^TP^{-1}KC + (I-HK)^T(GVG^T)^{-1}(I-HK)C\right]C^{-1} && (16)\\ & K^TP^{-1}K + (I-HK)^T(GVG^T)^{-1}(I-HK) && (17) \end{aligned}$$

In Eqn. 12, note that $(HK)^T = K^TH^T$. In Eqn. 13, we multiply by $GVG^T$ and its inverse; note also that $I-(HK)^T = (I-HK)^T$. In Eqn. 14, we substitute $GVG^T = C - HPH^T$, which follows from Eqn. 3. In Eqn. 15, we use $P^{-1}P = I$ and $C^{-1}C = I$, and we factor $(C-HPH^T)$ as $(I-HK)C$. In Eqn. 16, we substitute $K = PH^TC^{-1}$. Finally, in Eqn. 17, we distribute the $C^{-1}$ factor.

We can now write the cost function used by JCBB in terms of this final expression for $C^{-1}$:

$$\chi^2 = e^T\left[(I-HK)^T(GVG^T)^{-1}(I-HK) + K^TP^{-1}K\right]e \quad (18)$$

Eqn. 10 and Eqn. 18 are identical; thus, both formulations compute the same value. In some ways, the posterior-based test is more intuitive, since it corresponds to an explicit minimization of the same metric function used by non-linear SLAM systems.

### B. Feature Cloud Matching

In a standard landmark-based SLAM system, the position of each landmark is added to the state vector, and each observation of that landmark improves the estimate of its position. The position of the landmarks become correlated, due to the fact that different sets of landmarks are observed at different points in time.

In contrast, a feature-cloud matching approach does not add landmarks to the state vector, and thus does not attempt to compute optimal estimates of their positions. Instead, landmark detections from two poses $A$ and $B$ are used to estimate the motion of the robot between $A$ and $B$.

Perhaps the most canonical example of feature-cloud matching is scan-matching: two scans are aligned in order to recover the motion of the robot, but the scans do not update a global model of the underlying structure that led to the observations. Iterative Closest Point (ICP) is often used

for both 2D and 3D [11]. However, ICP methods require good initial estimates, which are not always available. In this case, features can be extracted from the data and the features are explicitly associated with each other. The relationship between the two poses can then be computed from the feature correspondence. Feature-cloud matching examples include matching camera data for navigation [12], and computing a rigid-body transformation to align two 3D LIDAR point clouds [13]. The latter case demonstrates how the same feature-cloud matching process applies to object recognition (matching an observation to a model).

A feature-cloud matching system is generally a good choice when many landmarks are detected at every pose. First, it can be impractical to add them all to the state vector: adding hundreds of landmarks at every robot pose would quickly bog down even the fastest SLAM implementations. Second, it is often unnecessary for accurate mapping: when many landmarks are detected simultaneously, the rigid-body transformation relating the two poses tends to be highly over-constrained, which greatly reduces the impact of noise in individual observations.

Feature-cloud matching can be performed using JCBB, but it is expensive to do so. First, the quadratically-increasing cost of incrementally computing the Mahalanobis distance at each level in the interpretation tree quickly becomes a bottleneck. Second, the depth of the interpretation tree is equal to the number of observations, which can measure in the hundreds.

### C. Incremental Posterior Joint Compatibility

The posterior joint compatibility test suggests an alternative approach for performing data association on feature clouds. The basic idea is to exploit the fact that feature observations are conditionally independent given the rigid-body transformation that relates poses $A$ and $B$. In other words, the critical quantity that needs to be estimated is the rigid-body transformation $T$ that projects points from coordinate frame $B$ into coordinate frame $A$. Everything else needed to compute the posterior joint compatibility can be recovered once $T$ is known.

Our approach is summarized below. For clarity, we provide example matrix and vector dimensions assuming that a robot is operating in the plane, i.e., that rigid-body transformations have three degrees of freedom and that landmarks are 2D point features; however, our method is not limited to this case.

1) Assume a prior on $T$, or alternatively, descend sufficiently far down the interpretation tree such that an initial solution to $T$ can be computed. The state vector is $3 \times 1$ and the covariance matrix is $3 \times 3$.
2) At level $i$ of the interpretation tree:
   a) Initialize a landmark based on observation $i$ from pose $A$. This enlarges the state vector to $5 \times 1$ and the covariance to $5 \times 5$. Because this observation was made from position $A$, it does not depend on $T$. Consequently, the covariance matrix will be block diagonal.
   b) When associating observation $i$ from pose $A$ with observation $j$ from pose $B$, perform an EKF-like update. This will update the posterior value of $T$ and the landmark location. Because the observation model is a function of both $T$ and the landmark position, the covariance matrix becomes dense.
   c) We do not need to maintain a full state estimate over $T$ and the landmark position, so we now marginalize out the landmark position. The state vector is now reduced to its original size of $3 \times 1$.

This method incrementally computes the posterior rigid-body transformation $T$ as we traverse the interpretation tree. Critically, the computational time required at each node in the tree is *constant*, as opposed to *quadratically increasing* with JCBB[1].

An advantage of this approach is that the posterior is improving in quality as we travel down the interpretation tree. This means that if the initial estimate of $T$ was poor, JCBB might encounter significant error due to linearization effects. Because JCBB does not update the prior as it traverses the interpretation tree, this error will affect every computation in the tree. With the approach above, the quality of the estimate improves as we traverse the tree, decreasing the effect of linearization errors.

However, in order to use this method in a branch-and-bound type of search, we need to be able to compute the posterior $\chi^2$ at each level of the tree. This might seem to be problematic, since we have marginalized-out the posterior positions of the landmarks. However, as we've shown previously, the desired $\chi^2$ error can be computed as a sum of the $\chi^2$ of both individual sets of observations with respect to the posterior. In other words, the posterior $\chi^2$ can be computed in terms of the posterior $T$ using the following procedure:

1) Compute the $\chi^2$ error associated with the prior on $T$ (characterized by mean $\mu_T$ and covariance $\Sigma_T$), $\chi^2_T = (x - \mu_T)^T \Sigma_T^{-1} (x - \mu_T)$.
2) For each associated pair of observations $i$ and $j$
   a) Compute the posterior position of the landmark given the two observations and $T$. We do this by projecting observation $j$ into coordinate frame $A$ using rigid-body transformation $T$. Note that for the purposes of this projection, $T$ has no uncertainty: it is already the desired posterior transformation.
   b) Combine the uncertain observation $i$ and the projected uncertain observation $j$ using an EKF update-like step.
   c) Compute the $\chi^2$ of both observations with respect to this posterior, and add it to the total.
3) Return the sum of all $\chi^2$ terms computed.

Note that each time we wish to compute the $\chi^2$ error for the set of data associations, we re-compute the posterior po-

[1]The computational complexity can be further reduced by modifying the EKF update step so that the values discarded during marginalization are never actually computed.

sitions for each landmark. This is because the posterior value of $T$ is updated at each level of the tree, and this affects the $\chi^2$ error associated with each of the landmark pairs. If we had not marginalized out the landmark positions at each level of the tree, this re-computation would not be necessary. Despite the seeming inefficiency of this procedure, we have traded the quadratic cost of maintaining the landmark posteriors for a *linear* cost associated with the algorithm above.

In summary, the IPJC algorithm follows the same general pattern as JCBB; an interpretation tree is constructed, and a search is conducted to find the best set of data associations. The principle difference is that the $\chi^2$ error is computed in terms of the posterior, rather than the prior. On linear problems, IPJC computes *exactly* the same result, but IPJC produces better quality results on non-linear problems due to the steady improvement in the quality of the posterior. Even more usefully, we show that by formulating the $\chi^2$ in terms of the posterior, the complexity of the computations performed at each node can be reduced from quadratic asymptotic complexity to linear.

### D. IPJC-Fast

The dominant computational cost of IPJC results from constantly recomputing the posterior positions of the landmarks and the resulting $\chi^2$ scores. We now describe an improvement to IPJC that dramatically decreases computational complexity without compromising the accuracy of the method.

The essential observation is that the $\chi^2$ strictly increases as we progress down the interpretation tree. (This follows from the fact that, at each level of the tree, we compute the state estimate with the minimum $\chi^2$ error. Any future modifications to the state estimate can only increase the error.)

During the first few data associations, the state estimate for $T$ often changes significantly. But as we traverse farther down the interpretation tree, $T$ becomes more confident and the state changes grow smaller. As a result, the $\chi^2$ for earlier observations do not change very much.

This suggests a simple strategy: instead of recomputing the $\chi^2$ cost of each observation at every node in the tree, simply cache the values from the previous level. The resulting $\chi^2$ estimate will be strictly smaller than the true $\chi^2$. If this lower-bound of the $\chi^2$ cost is greater than the $\chi^2$ threshold used in the branch-and-bound search, the sub-tree rooted at that node can be pruned.

Recall also that the branch-and-bound search also prunes nodes whose $\chi^2$ error is worse than the best-known solution. Whenever a node appears to be the new "best" solution, we need to recompute the correct $\chi^2$ error in order to ensure that it is correct.

As our results demonstrate, this "lazy" strategy pays off: this algorithm, which we call IPJC-Fast, produces exactly the same results as IPJC, but does so in a fraction of the time. Neglecting the occasional need to recompute the full $\chi^2$ cost, the asymptotic complexity at each node in the interpretation tree is now $O(1)$. This is in comparison to the quadratic costs of JCBB.

## IV. Results

### A. Simulation Results

In this experiment, and the other synthetic experiments that follow, we simulated a planar robot operating in a field of randomly-placed landmarks. In the linear experiments, the robot has no orientation (or equivalently, has a "perfect" compass) and observes the distance in the $\hat{x}$ and $\hat{y}$ directions to landmarks. In the non-linear experiments, the robot acquires range-bearing observations. The sensor range and obstacle density are configured so that around 15 landmarks are visible from any given position. The robot trajectory is sampled such that, between adjacent poses, there are around 10 matching landmarks.

### B. Accuracy in comparison to the ideal $\chi^2$ distribution

We now wish to demonstrate that IPJC computes better estimates of the true compatibility cost in non-linear problems. Our methodology relies on synthetic datasets in which the magnitude of noise and data association can be known with certainty. In this case, the compatibility cost of the *true* data associations should obey a $\chi^2$ distribution.

Fig. 2 shows histograms of the computed compatibility cost on a large number of trials versus the ideal distribution (given by a $\chi^2$ distribution of the appropriate parameters). Results are shown for JCBB, PJC, and IPJC, for both low-noise and high-noise situations. All algorithms produce reasonable results in low-noise situations, which is sensible since linearization effects are minimized when noise is low.

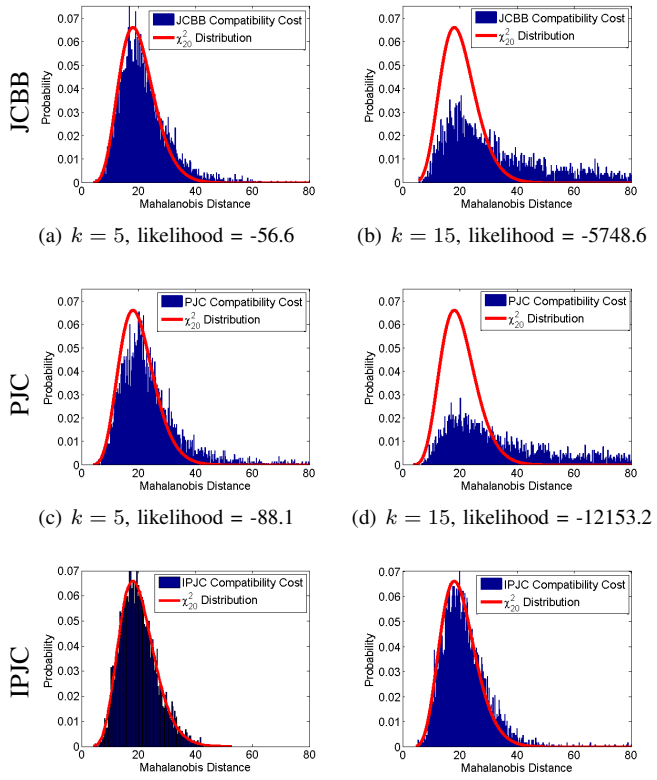
However, in high-noise situations, the performance of the algorithms is radically different. IPJC's compatibility costs follow the correct distribution much more closely than either JCBB or PJC. (Recall that PJC does not incrementally update the posterior, and so does not have the robustness to noise that IPJC has.)

We can quantify the similarity of the histograms to the ideal $\chi^2$ distributions by computing the likelihood of sampling the empirical distribution from the ideal distribution. These likelihoods substantiate our claims: IPJC's log-likelihood is -110.9, whereas JCBB's log-likelihood is -5748.6.

It is clear from Fig. 2 that JCBB has a tendency to over-estimate the compatibility cost of true data associations due to the effects of non-linearities. This effect is further demonstrated by Fig. 3, which plots the compatibility costs computed by our method versus JCBB. The high compatibility cost peaks computed by JCBB exceed the $\chi^2$ threshold and result in false negative data associations. In high-noise settings, JCBB incorrectly rejects many of the correct data associations.

### C. False vs True Positives

False negatives are problematic since they deprive a SLAM solution of information that could improve the quality of the map. However, false positives can be catastrophic,




(a) $k = 5$, likelihood = -56.6 (b) $k = 15$, likelihood = -5748.6

(c) $k = 5$, likelihood = -88.1 (d) $k = 15$, likelihood = -12153.2

(e) $k = 5$, likelihood = -40.1 (f) $k = 15$, likelihood = -110.9

Fig. 2. Compatibility costs versus the ideal distribution. Given ground truth, it is possible to determine the distribution of compatibility costs that a data association algorithm *should* compute; this ideal distribution is shown as a red line. For each algorithm we plot the empirical distribution for two noise levels (algorithms span across rows; noise levels span columns). At low noise levels (left column), each algorithm does fairly well. However, at higher noise levels (right column), IPJC is dramatically more accurate. The log likelihood is shown in each sub-caption and quantifies the similarity between the empirical and ideal distribution; numbers closer to zero represent greater similarity.

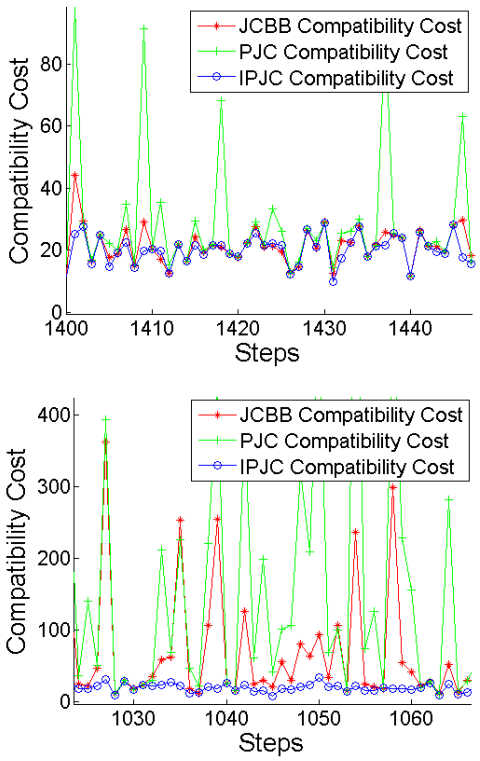



Fig. 3. Comparison of joint compatibility costs. Correct hypotheses are used to calculate the compatibility costs in the proposed methods and JCBB for two different noise levels. Linearization effects cause JCBB and PJC to dramatically over-estimate the compatibility cost, which ultimately causes errors in data association. IPJC computes lower and more accurate compatibility costs.

leading to divergence. We show the false positive and true positive rates in Fig. 4. The performance of the algorithms is plotted as a function of the noise magnitude, which increases in the $x$ axis. The figure demonstrates that IPJC and IPJC-Fast produce higher fidelity results: both lower false positive rates and higher true positive rates. Users can trade-off performance in these categories by adjusting the $\chi^2$ data association threshold.

We have included Sequential Compatibility Nearest Neighbor (SCNN), which greedily matches features one at a time (see [2] for more details). We have also included RANSAC [14] for comparison. For RANSAC, we report results for the consensus threshold that maximized RANSAC's performance, and for two different iteration limits: one which maximized accuracy (15000), and a second that represents a reasonable compromise between quality and speed (5000).

### D. Computational Cost

We now consider the computational cost of our methods versus other data association methods. Fig. 5 demonstrates that IPJC-Fast is consistently faster than JCBB, and that it does not suffer from the spikes in computational complexity arising from linearization error that affect JCBB.

SCNN, as one of the simplest possible data association algorithms, is the fastest method. However, it produces significantly inferior data associations.

The time complexity of JCBB and IPJC-Fast are dependent on the noise level in the problem: as noise increases, more data association hypotheses appear plausible. As a result, more nodes in the interpretation tree must be expanded. As shown by Fig. 6, IPJC and IPJC-Fast are both faster in absolute terms, and exhibit slower growth in time. As expected, SCNN and RANSAC are unaffected by the noise level.

### E. Victoria Park

Finally, we demonstrate our algorithm on a real-world dataset: Victoria Park. We use the standard tree detection method described in [15] for landmark observations. We established ground truth based on the minimum-error configuration using manually-verified data associations.

In order to make the fairest possible comparison to RANSAC, we tuned the RANSAC parameters to optimize its performance. Beginning with a very large number of iterations, we searched for the consensus threshold that minimized the error in the map, arriving at a value of

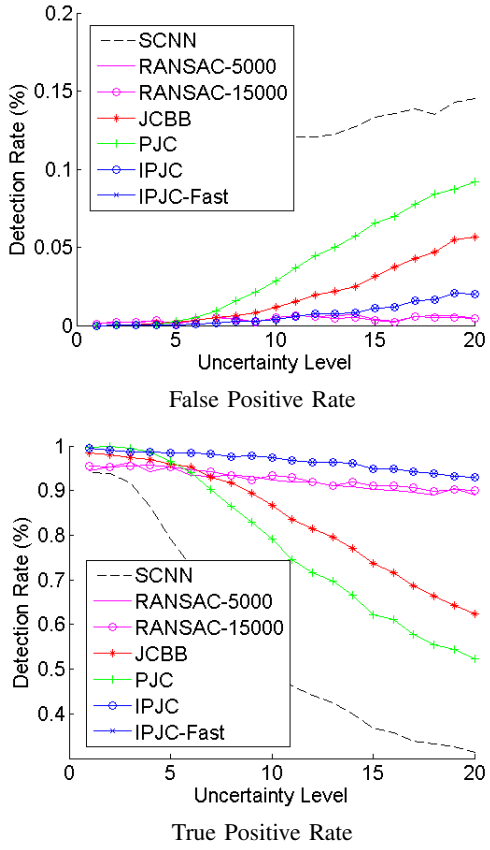



Fig. 4. False and true positive rates. False positives can cause catastrophic errors, but a high true positive rate is necessary to produce accurate maps. For a given $\chi^2$ threshold, IPJC generates fewer false positives (excluding RANSAC) and more true positives than the other methods. Note that IPJC and IPJC-Fast produce precisely the same data. While RANSAC produces fewer false positives at high noise levels, this comes at the cost of much higher computational costs.

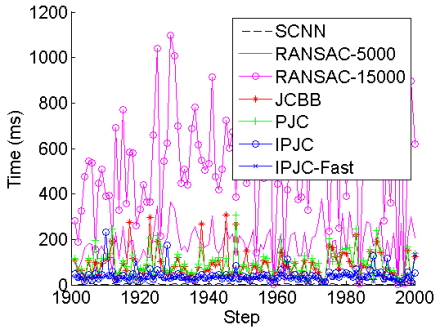



Fig. 5. Computational complexity. Spikes occur when large numbers of features are detected. SCNN is the fastest method, but its accuracy makes it unusable in most problems. IPJC-Fast consistently outperforms JCBB and RANSAC.

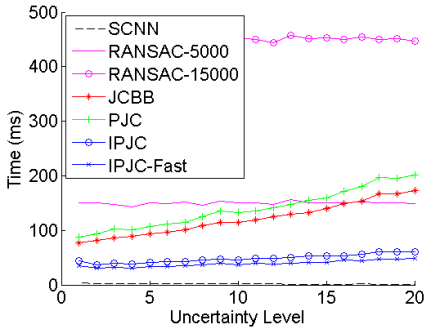



Fig. 6. Computational complexity versus noise level. Joint compatibility methods (including IPJC) tend to require more computation in high noise environments, since more hypotheses appear plausible. However, the growth rate for IPJC-Fast is much lower than for JCBB. RANSAC's complexity is independent of the level of noise, as expected.

1.0 m. We then reduced the number of iterations in order to improve runtime until the quality of the map started increasing rapidly. In this way, we arrived at 5000 RANSAC iterations, which is sensible given that each pose observes around 15 landmarks, and two associations are required to compute a rigid-body transformation.

### *F. Victoria Results*

We built maps using several different data association algorithms and then compute the posterior map using a sparse Cholesky factorization method [16].

TABLE I

VICTORIA PARK MAP ACCURACY. The mean squared error is computed versus a hand-annotated ground truth. IPJC and IPJC-Fast, which produce the same results, are the most accurate of the tested methods.

| | SCNN | RANSAC | JCBB | PJC | IPJC | IPJC-Fast |
|---|---|---|---|---|---|---|
| X | Diverged | 1.0470 | 0.5334 | 0.5731 | 0.2159 | 0.2159 |
| Y | Diverged | 2.0367 | 0.9017 | 0.9934 | 0.5801 | 0.5801 |
| $\theta$ | Diverged | 0.0074 | 0.0002 | 0.0001 | 0.0001 | 0.0001 |

The resulting maps are shown in Fig. 7. SCNN makes data association errors that cause the maps to diverge. The remaining methods, RANSAC, JCBB, PJC, IPJC, and IPJC-Fast produce visually indistinguishable maps. However, the maps are not identical: due to differing true and false positive rates, the quality of the maps varies between methods. Table I shows the mean squared error for the various methods. IPJC-Fast (which produces the same results as IPJC, just faster) produces a higher-quality result than the other methods.

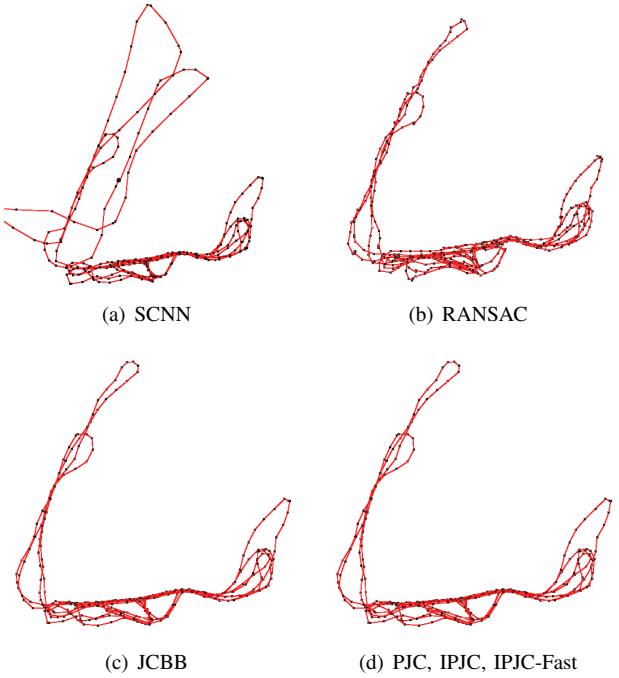
These performance differences can be explained by the true and false positive rates; see Table II. IPJC and IPJC-Fast have the highest true positive rate and the lowest false positive rate of any of the methods considered.

We also show the computational costs associated with the different data association methods in Fig. 8. Naturally, SCNN is the fastest (though its quality is poor); IPJC-Fast outperforms all other methods.

TABLE II

VICTORIA PARK TRUE/FALSE POSITIVE RATES. On this real-world data, IPJC and IPJC-Fast outperform all other methods in both true positives and false positives.

| | SCNN | RANSAC | JCBB | PJC | IPJC | IPJC-Fast |
|---|---|---|---|---|---|---|
| True pos. | 0.2410 | 0.9132 | 0.9565 | 0.9623 | 0.9818 | 0.9818 |
| False pos. | 0.0608 | 0.0121 | 0.0044 | 0.0018 | 0.0004 | 0.0004 |



(a) SCNN (b) RANSAC

(c) JCBB (d) PJC, IPJC, IPJC-Fast

Fig. 7. Victoria Park posterior maps. We used each data association algorithm to produce a pose graph that was then optimized using sparse Cholesky decomposition. The graph generated by SCNN fails to converge due to erroneous data associations. The remaining methods generated visually reasonable graphs (visually indistinguishable in the case of PJC, IPJC, and IPJC-Fast), though a numerical comparison to ground-truth shows that IPJC-Fast's map was the most accurate.

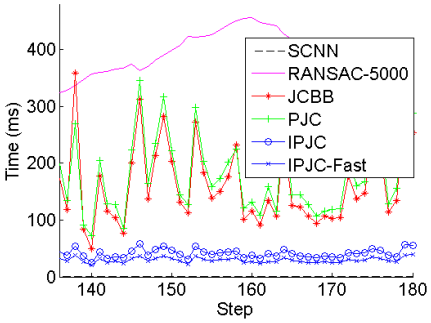



Fig. 8. Victoria Park Computational Complexity. IPJC-Fast was the fastest of the methods that produced a reasonable map. SCNN ran in less time, but the resulting map diverged due to data association errors.

## V. Conclusion

We have presented IPJC-Fast, a new method for computing data associations that is both fast and accurate. It is equivalent to the gold-standard JCBB on linear problems, but is formulated in terms of the posterior distribution. It exploits the probabilistic structure present in feature cloud matching, a task common in both SLAM and in object recognition, to achieve significant speed savings over JCBB. Further, by updating the posterior distribution at each level of the interpretation tree, IPJC-Fast computes more accurate compatibility costs.

We demonstrated IPJC-Fast's performance in both simulation and on real data. With the help of ground-truth data, we were able to show that the accuracy of the compatibility scores were significantly more consistent with those predicted by a $\chi^2$ distribution. We also demonstrated our method on the Victoria Park dataset, illustrating that it is effective on real-world data.

On feature-cloud matching problems, IPJC-Fast represents significant improvements over existing methods, including JCBB, RANSAC, and SCNN. Reference implementations are available at the authors' website, `http://april.eecs.umich.edu`.

## VI. Acknowledgments

This work was supported by NSFC grant 61105090 and U.S. DoD Grant FA2386-11-1-4024.

## References

[1] U. Frese, "Closing a million-landmarks loop," in *In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing. submitted*, 2006, pp. 5032–5039.

[2] J. Neira and J. D. Tardos, "Data association in stochastic mapping using the joint compatibility test," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 890–897, December 2001.

[3] T. Bailey, "Mobile robot localisation and mapping in extensive outdoor environments," Ph.D. dissertation, Australian Centre for Field Robotics, University of Sydney, August 2002.

[4] E. Olson, "Recognizing places using spectrally clustered local matches," *Robotics and Autonomous Systems*, 2009.

[5] M. Bosse, P. Newman, J. Leonard, and S. Teller, "Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework," *International Journal of Robotics Research*, vol. 23, no. 12, pp. 1113–1139, December 2004.

[6] E. Olson, "Robust and efficient robotic mapping," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, June 2008.

[7] E. Olson, J. Strom, R. Morton, A. Richardson, P. Ranganathan, R. Goeddel, M. Bulic, J. Crossman, and B. Marinier, "Progress towards multi-robot reconaissance and the MAGIC 2010 competition," *Journal of Field Robotics*, To appear.

[8] R. Smith, M. Self, and P. Cheeseman, "A stochastic map for uncertain spatial relationships," in *Proceedings of the International Symposium of Robotics Research (ISRR)*, O. Faugeras and G. Giralt, Eds., 1988, pp. 467–474.

[9] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994, pp. 935–938. [Online]. Available: citeseer.ist.psu.edu/lu94robot.html

[10] P. Maybeck, *Stochastic models, estimation and control*, ser. Mathematics in science and engineering. Academic Press, 1982, no. v. 2.

[11] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[12] K. Konolige and M. Agrawal, "Frameslam: From bundle adjustment to real-time visual mapping," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1066 –1077, oct. 2008.

[13] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, "Point feature extraction on 3D range scans taking into account object boundaries," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[14] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis andd automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[15] J. E. Guivant, F. R. Masson, and E. M. Nebot, "Simultaneous localization and map building using natural features and absolute information," *Robotics and Autonomous Systems*, vol. 40, no. 2-3, pp. 79–90, 2002.

[16] F. Dellaert, "Square root SAM," in *Proceedings of Robotics: Science and Systems (RSS)*, Cambridge, USA, June 2005.