BANSAC: A dynamic BAyesian Network for adaptive SAmpLe Consensus

Miraldo, Pedro; Piedade, Valter

TR2023-124 October 02, 2023

Abstract
RANSAC-based algorithms are the standard techniques for robust estimation in computer vision. These algorithms are iterative and computationally expensive; they alternate between random sampling of data, computing hypotheses, and running inlier counting. Many authors tried different approaches to improve efficiency. One of the major improvements is having a guided sampling, letting the RANSAC cycle stop sooner. This paper presents a new adaptive sampling process for RANSAC. Previous methods either assume no prior information about the inlier/outlier classification of data points or use some previously computed scores in the sampling. In this paper, we derive a dynamic Bayesian network that updates individual data points’ inlier scores while iterating RANSAC. At each iteration, we apply weighted sampling using the updated scores. Our method works with or without prior data point scorings. In addition, we use the updated inlier/outlier scoring for deriving a new stopping criterion for the RANSAC loop. We test our method in multiple real-world datasets for several applications and obtain state-of-the-art results. Our method outperforms the baselines in accuracy while needing less computational time. The code is available at https://github.com/merlresearch/bansac.

IEEE International Conference on Computer Vision (ICCV) 2023
BANSAC: A dynamic BAyesian Network for adaptive SAmple Consensus

Valter Piedade  
Instituto Superior Técnico, Lisboa  
valter.piedade@tecnico.ulisboa.pt

Pedro Miraldo  
Mitsubishi Electric Research Labs  
miraldo@merl.com

Abstract

RANSAC-based algorithms are the standard techniques for robust estimation in computer vision. These algorithms are iterative and computationally expensive; they alternate between random sampling of data, computing hypotheses, and running inlier counting. Many authors tried different approaches to improve efficiency. One of the major improvements is having a guided sampling, letting the RANSAC cycle stop sooner. This paper presents a new adaptive sampling process for RANSAC. Previous methods either assume no prior information about the inlier/outlier classification of data points or use some previously computed scores in the sampling. In this paper, we derive a dynamic Bayesian network that updates individual data points’ inlier scores while iterating RANSAC. At each iteration, we apply weighted sampling using the updated scores. Our method works with or without prior data point scorings. In addition, we use the updated inlier/outlier scoring for deriving a new stopping criterion for the RANSAC loop. We test our method in multiple real-world datasets for several applications and obtain state-of-the-art results. Our method outperforms the baselines in accuracy while needing less computational time. The code is available at https://github.com/merlresearch/bansac.

1 Introduction

Outliers are one of the primary causes of poor performance in computer vision. Robust estimators are essential since imaging sensors suffer from several types of noise and distortions. Removing outliers is one of the initial and more relevant steps in many computer vision tasks, such as relative pose estimation [12, 28, 32, 35, 39, 50], camera localization [10, 42, 43, 51], and mapping [22, 26, 29, 36, 44, 45].

The gold-standard robust estimator is RANSAC (RAmdom Sample Consensus), introduced in [23]. RANSAC-based algorithms are iterative methods that, at each iteration: sample minimal sets, estimate a model, and run inlier counting. The output is the solution with the largest consensus.

The original RANSAC dates back to 1981. Over the years, many authors changed the original loop to alleviate some of its limitations. All these alternatives focus on improving the sampling process, getting a better hypothesis, improving the stopping criteria, or changing the inlier counting. Most modifications add significant gains in computational efficiency. This paper focuses on improving the sampling efficiency even further. The main question we want to tackle in this paper is: Will changing the scoring weights over iterations help in sampling and defining the stopping criteria? To answer this question, we propose BANSAC, a new sampling strategy for the RANSAC loop. Figure 1 illustrates our approach.

Previous methods such as [4, 11, 16, 37, 47, 48] focus on exploiting scoring priors or considering some geometric relationships. However, the best-performing methods...
keep these scores fixed while running RANSAC. We focus on updating the scores online and using them for sampling minimal data. By modeling the problem with probabilities, we propose a new sampling strategy that uses a dynamic Bayesian network for updating the scores. These are the paper’s main contributions:

- A novel adaptive sampling strategy that uses a dynamic Bayesian network to update data points’ inlier scores. Our method does not need any prior information about the quality of the data, although it can use it;
- A new simple and intuitive stopping criterion using the updated scores; and
- Several experiments with multiple datasets show that our approach outperforms the best baselines in accuracy, being also more efficient.

We implemented BANSAC using C++, within the OpenCV USAC framework [1].

2 Related Work

Over the years, RANSAC has been improving in a variety of areas. [40] offers a single universal framework (USAC) that unites several improvements. Below we summarize some RANSAC improvements split into sampling and non-sampling strategies.

2.1 Sampling strategies

The original RANSAC [23] assumes that every data point has the same likelihood of being an inlier. Several new sampling improvements have been proposed. We split these methods into: heuristic [4, 14, 16, 27, 37, 48], probabilistic [8, 24, 33, 47, 49], and learning approaches [9, 11, 15].

Heuristic-based strategies: Heuristic-based strategies take advantage of problem-specific characteristics to guide sampling. NAPSAC [48] assumes that points in high-density areas are more likely to be inliers. The algorithm chooses the first point randomly and completes the sample within a certain distance from the first. NAPSAC often leads to local or degenerate models for more complex problems. PNAPSAC [4] improves some of NAPSAC issues by iteratively increasing the search space. One of the most used sampling strategies is PROSAC [16]. Using, e.g., similarity scores between point matches, PROSAC prioritizes the sampling of points with better scores. A drawback of this method is that it cannot be applied in general since it requires some previously computed score. CS-RANSAC [27] argues that the matched features should neither be collinear nor adjacent to avoid degeneracies. CS-RANSAC defined the problem using a Constraint Satisfaction Problem (CSP) for homography matrix estimation. GroupSAC [37] assumes that data can be split into groups according to their coordinates or based on the number of images observing the points.

Probability-based strategies: Probabilistic-based sampling strategies such as [24, 34, 47] focus on estimating prior probabilities for the data. These probabilities guide data selection during the sampling step. MLESAC [49] improves the hypotheses verification process in fundamental matrix estimation. Guided-MLESAC [47] further develops MLESAC by introducing a guided sampling modeled by two distinct distributions, one for matches and the other for mismatches. EVSAC [24] uses a Gamma and a Generalized Extreme Value distribution to model inliers and outliers, respectively. Although the problem differs from ours, [33] derives an approach that uses a probability for modeling inlier/outlier classification over iterations utilizing multiple match hypotheses (a single feature on an image matches more than one feature on the second image).

The closest work to ours is BAYSAC [8], which updates the inlier probabilities iteratively, using it to guide the sampling. At each iteration, after choosing a minimal set of data points and computing the respective model hypothesis, the method updates the probability of the data points in the minimal set based on how good the hypothesis was. Although this method updates the inlier probability at each iteration, these updates are limited to the sampled points, which do not perform well without a good prior. In this paper, we propose a new approach in which all the data points’ inlier probabilities are updated every iteration based on the inlier/outlier classifications.

Learning-based strategies: There has been widespread use of neural networks in many areas of computer vision, RANSAC sampling being no exception. NG-RANSAC [11] focuses on sampling by learning to estimate matching scores for the input correspondences for relative pose problems. Instead of scoring data points, DSAC [9] learns to score a set of previously computed hypotheses. NeFSAC [15] predicts the probability that a minimal sample leads to an accurate solution, thus preventing the estimation of models using bad minimal samples.

Our method does not require training. However, pre-computed matching scores from learning-based solutions can be given as input to BANSAC.

2.2 Non-sampling strategies

Below we list some key works on improvements to RANSAC concerning the inlier threshold, inlier counting, local optimization, and stopping criteria.

Inlier threshold: RANSAC uses the inlier ratio to select the best model (largest consensus). To compute the inlier ratio, a problem-dependent threshold is required. To avoid setting this parameter, MINPRAN [46] proposes to model
it using the model parameters. Alternatively, MAGSAC [6] and MAGSAC++ [7] reformulate the problem to use a weighted least squares fitting for model evaluation, using point scores as weights.

**Inlier counting:** Inlier/outlier classification is computationally heavy since, in each iteration, all the data needs to be checked. Some authors have developed strategies to avoid scoring all the points every iteration, [3,13,17,30,31]. Others check if the estimated models are valid, avoiding the scoring process for invalid models, such as [3,19–21,25]. To avoid scoring unnecessary data points, a bail-out test is proposed in [13]. The scoring stops when the current model fails to have a higher inlier count than the best model. SPRT [17, 31] estimates a likelihood ratio to decide if a model is good using the minimum possible amount of data.

**Local optimization:** To improve accuracy, some authors added a new step called local optimization. LO-RANSAC in [18] recompute the model parameters when a new best model is found, using only the inliers. In [5], a method called Graph-Cut RANSAC is proposed. It takes advantage of the spatial coherence of the data to refine the estimated model by assuming that close neighbor points should have an equal classification in inlier/outlier. The data is arranged in a graph with edges between nearby points and is minimized by an energy cost function that penalizes neighbor points with different classifications.

**Stopping criteria:** The stopping criterion checks if RANSAC found a good enough solution and can exit the loop. The vanilla RANSAC in [23] estimates how many iterations are needed until one all-inlier model is found, using only the inliers. In [5], a method called Graph-Cut RANSAC is proposed. It takes advantage of the spatial coherence of the data to refine the estimated model by assuming that close neighbor points should have an equal classification in inlier/outlier. The data is arranged in a graph with edges between nearby points and is minimized by an energy cost function that penalizes neighbor points with different classifications.

An intuitive way of deriving a sampling technique with the above assumptions is to have changeable inlier/outlier scores of data points per iteration. Our method updates the scores based on inlier/outlier classifications from previous iterations. For modeling changeable scores over iterations, we define unknown variables, namely \( x_n^k \) representing the best guess for inlier/outlier classification for \( x_n \) at iteration \( k \). For modeling the scores for each data point at each iteration, we use probabilities, i.e., \( P(x_n^k | c_{1:n}^k) \), where \( x_n^k \) can take the values of inlier and outlier. Unfortunately, we do not have a direct way of measuring \( P(x_n^k = \text{inlier}) \). Instead, in this paper, we use inliers/outliers classifications obtained from previous iterations as evidence, i.e., for sampling we use \( P(x_n^k = \text{inlier} | c_{1:n}^k) \), where \( C_{1:n}^k \triangleq c_n^k = \text{inlier} \). Table 1 lists some important notations we use in our derivations.

In the following subsection, we give a method overview.

### 4 BANSAC Method

This paper focuses on deriving an efficient sampling of data points, i.e., getting \( S^k \). We note that getting \( S^k \) from \( S^k \) is problem-dependent; BANSAC is independent of the problem. We take two simple assumptions:

1. We assume that sampling data points with higher inlier scores gives a better model hypothesis; and
2. As we iterate through RANSAC, we get a better sense of whether \( x_n \) is an inlier or outlier.

### 3 Background and Notations

RANSAC is an iterative method for solving a generic problem of type \( f(x, \theta) = 0 \), where \( x \) is some data satisfying the model \( \theta \). For simplicity, with a small abuse of notation, we call \( x \) a data point, because it can represent other types of features such as matches. The method iterates for a maximum of \( K \) iterations while alternating between, 1) sampling \( S^k \subset Q \) data points, where \( Q \triangleq \{x_1, ..., x_N\} \); 2) computing model hypothesis \( \theta^k \); and 3) doing inlier counting, i.e., get \( C^k \triangleq \{c_1^k, ..., c_N^k\} \), where \( c_n^k \) is the inlier/outlier classification of \( x_n \) at iteration \( k \). The output is the best-scored model and best inlier/outlier classification set, here denoted as \( \{\theta^*, C^*\} \).

### 4.1 Algorithm outline

Algorithm 1 outlines the proposed method. The RANSAC-based loop starts with an optionally given \( P^0 \), which can be obtained from matching scores (or other initial guesses), or set to a predefined value (i.e., not using previously pre-computed scores). At each iteration \( k \), we generate a minimal set \( S^k \subset Q \) via weighted sampling,
Algorithm 1: BANSAC algorithm outline

Input – Data \( Q \) and (optional) pre-computed scores \( p^0 \)
Output – Best model \( \theta^* \), and \( C^* \)

1. \( k \leftarrow 1; \)
2. while \( k < K \) do
3. \( S^k \leftarrow \text{weighted_sampling}(Q, p^{k-1}); \quad \triangleright \text{Sec. 4.3} \)
4. \( \theta^k \leftarrow \text{hypothesis}(S^k); \)
5. \( C^k \leftarrow \text{model_evaluation}(Q, \theta^k); \)
6. \( \theta^*, C^* \leftarrow \text{best_model}(C^k, \theta^k); \)
7. \( p^k \leftarrow \text{update_probabilities}(C^k, x^{0:k-1}); \quad \triangleright \text{Sec. 4.2} \)
8. if stopping_criterias \( (p^k) \) break; \quad \triangleright \text{Sec. 4.4} \)
9. \( k \leftarrow k + 1; \)

using \( p^{k-1} \) as weights (see notations in Tab. 1), which is shown in Line 3. Next, we compute the hypothesis model \( \theta^k \), run inlier counting, and update the best model if needed, as described in Lines 4 to 6. Line 7 updates the probabilities for the next iteration, i.e., computes \( p^k \). In addition to the new sampling and updating the probabilities, using \( p^k \), we derive a new stopping criterion for our RANSAC-based loop, which we use at Line 8.

Section 4.2 derives the probabilistic model, Sec. 4.3 describes the weighted sampling strategy, and Sec. 4.4 introduces the new stopping criterion.

4.2 Probabilistic model and inference

We describe our method by modeling data points’ inlier/outlier probability.

Probabilistic model: Since we want the data point probabilities to change over iterations, we use a dynamic Bayesian network (DBN) as our probabilistic model. A DBN is a probabilistic graphical model that uses variables (nodes representing states and observations) and their conditional dependencies (edges) in a directed acyclic graph (see [41]). Starting with the variables, at each iteration \( k \), we have the data points state and the inlier/outlier classifications (evidence) obtained so far (i.e., from 1 to \( k \)). So, for iteration \( k \), we have nodes \( X \) and \( C \) (see notations in Tab. 1).

For the graph edges, we have the following constraints:

1. We want our sampling to be general. Then, for a certain iteration \( k \), we assume that the inlier/outlier probabilities of different data points and the classifications are independent of each other. At each iteration \( k \), the probability of \( x_n^k \) is updated based on the \( x_n \)’s previous probabilities and the previous inlier/outlier classifications (our evidence needs to constrain only the next probability estimate);
2. The inlier/outlier evidence at each iteration, \( c_n^k \), depends only on the model \( \theta^k \), which depends on \( S^k \) and, by consequence, on \( x_n^{k-1} \).

Formally, for iteration \( k \), we have the following constraints:

\[
\begin{align*}
x_n^k & \perp X \setminus X_n, C \setminus c_n^k \mid X_n, c_n^k \quad (1) \\
c_n^k & \perp X \setminus x_n^{k-1}, C \mid x_n^{k-1} \quad (2)
\end{align*}
\]

The first important consequence of these conditionally independent constraints is that we have an independent DBN for each data point, \( x_n \), each with its own weights. Then, for iteration \( k \), we define the DBN per data point as shown in Fig. 2.

Markov assumptions: The DBN derived above has the unbounded problem of increasing exponentially with the number of iterations\(^1\). To solve this problem, we follow the typically used Markov assumptions. For simplicity, here we derive the first-order Markov assumption for our problem. The second and third-order assumptions are tested in Sec. 5.1, and the derivations are provided in the supplementary materials; they are similar but slightly more intricated. In addition to the conditionally independent constraints derived above, we have

\[
x^j_n \perp x_n^{0:j-2} \mid x_n^{j-1}, c_n^j, \forall j, \quad (3)
\]

which means

\[
P(x_n^j \mid x_n^{0:j-1}, c_n^j) = P(x_n^j \mid x_n^{j-1}, c_n^j), \forall j. \quad (4)
\]

Now, by applying the chain rule of probabilities, we write the joint probability at iteration \( k \) as

\[
P(x_n^{0:k}, c_n^{1:k}) = P(x_n^0) \prod_{j=1}^{k} \phi_j(x_n^{0:j}, c_n^{1:k}), \quad (5)
\]

where

\[
\phi_j(x_n^{0:j}, c_n^{1:k}) = P(x_n^j \mid x_n^{j-1}, c_n^j)P(c_n^j \mid x_n^{j-1}). \quad (6)
\]

\(^1\)Would need huge conditional probability tables (CPT) and the increase in computational cost.
Exact inference: In our sampling strategy, we use \( P(x_n^k = \text{inlier} \mid C_n^{1:k}) \), for all \( n \). This means that we are doing inference of \( x_n^k \), with evidences \( C_n \), and hidden variables \( X_n \). Given Eq. 6, after some derivations, the exact inference is given by (see [41, Sec.14.4]):

\[
P(x_n = \text{inlier} \mid C_n^{1:k}) = \alpha \Phi(x_n^{k} = \text{inlier}, x_n^{0:k-1}, C_n^{1:k}),
\]

where \( \alpha \) is a normalization factor, and

\[
\Phi(x_n^{k}, x_n^{0:k-1}, C_n^{1:k}) = \\
\sum_{x_n^{-1}} \phi_k(x_n^{0:k}, c_n^{1:k}) \sum_{x_n^{-2}} \phi_{k-1}(x_n^{0:k}, c_n^{1:k}) \\
\cdots \sum_{x_n} \phi_2(x_n^{0:k}, c_n^{1:k}) \sum_{x_n^{0}} \phi_1(x_n^{0:k}, c_n^{1:k}).
\]

Notice \( x_n^k \) can take two values; it can be either inlier or outlier. This means that \( \Phi(.) \) is a 2-dimension tuple; in Eq. 7 we pick the case \( x_n^k = \text{inlier} \). In addition, the summations for \( x_n^k \) have two terms for all \( j \).

A convenient result of Eq. 8 is that \( \Phi(.) \) can be computed recursively as follows:

\[
\Phi(x_n^{k}, x_n^{0:k-1}, C_n^{1:k}) = \\
\sum_{x_n^{-1}} \phi_k(x_n^{0:k}, c_n^{1:k}) \Phi(x_n^{k-1}, x_n^{0:k-2}, C_n^{1:k-1})
\]

for \( k \geq 1 \), and \( \Phi(x_n^{0}, -,-) = P(x_n^{0}) \). This means that at each iteration \( k \), we can use the \( \Phi(.) \) computed from the previous iteration, with no additional computational cost when increasing the number of iterations. The pseudo-code for the probability update is in the supplementary materials.

We tried several alternatives for the Conditional Probability Tables (CPT) in Eq. 6. In our experiments, for \( P(c_n^{k} \mid x_n^{k-1}) \), we use a Leaky ReLU, weighted using the inlier counting. For the CPT of \( P(x_n^{k} \mid x_n^{k-1}, c_n^{k}), \) we get the probabilities empirically. Due to space limitations, we present more details about the CPTs in the supplementary material, including some experiments.

4.3 Weighted sampling

Weighted sampling aims at getting the minimal set \( S^k \subset Q \) for computing hypothesis \( \theta^k \). To increase the chances of only selecting inliers, we take the estimated probabilities \( \mathcal{P}^{k-1} \) and create a weighted discrete distribution. Data points with a higher probability of being an inlier will have a higher chance of being sampled.

In addition to directly using \( \mathcal{P}^{k-1} \) for weighting the discrete distribution, we tested using various activation functions such as leaky ReLU, sigmoid, and tanh functions. In the experiments, we directly use \( \mathcal{P}^{k-1} \). Due to space constraints, we show results with other activation functions in the supplementary materials.

4.4 Stopping criterion

Besides using the probabilities \( \mathcal{P}^k \) in sampling, we also exploit them in defining a stopping criterion. We know that \( \mathcal{P}^k \) will influence the sampling, meaning that, after reaching a low probability threshold (which we denoted as \( \tau \)), we can say that a data point will not be considered for sampling. Based on this idea, we derive a new, simple stopping criterion. At each iteration \( k \), we add the following steps to Algorithm 1:

1. In Line 6, we keep the smallest number of outliers (best case scenario so far), denoted as \( O^* \); and

2. After updating the probabilities in Line 7, we compute the number of data points with \( P(x_n^k = \text{inlier} \mid C_n^{1:k}) \) lower than a predefined threshold \( \tau \), which we denote here as \( \hat{O}^k \).

Our stopping criterion is triggered when \( \hat{O}^k \geq O^* \), which we check at each iteration in Line 8. This means that the current best model has a bigger or equal number of inliers than the set of \( P(x_n^k = \text{inlier} \mid C_n^{1:k}) \) being selected for sampling; it can be assumed we have only inliers in the sampling set. We exit the loop because there is a low chance of getting a better solution.

Our stopping criterion can be used alone or added to existing ones, such as the RANSAC [23], SPRT [31] or PROSAC [16]. We do experiments on the combinations of these criteria in the supplementary materials.

4.5 Degenerative configurations

Degenerate configurations in minimal solvers can lead to poor RANSAC estimates. BANSAC is susceptible to such settings as other RANSAC-based methods. In a worst-case scenario, since the probability updates depend on the inlier ratio, after sampling degenerate configurations, BANSAC can create a bias towards degenerative sampling. However, it may take several iterations of consecutive degenerate configurations before that bias has some impact on the sampling. Moreover, since each data point always has a chance of being selected, even if it is minimal, BANSAC can reverse that bias as soon as new non-degenerate solutions are obtained.

Although possible, we highlight that we did not experience any issues with degenerative configurations during our experiments. In addition, we are using the OpenCV USAC framework that has built-in methods to handle many of those cases. When those degenerative configurations are detected, the probabilities are not updated.
5 Experiments

We evaluate BANSAC in three computer vision problems: calibrated relative pose, uncalibrated relative pose, and homography estimation. We start our experiments with an ablation study to access the need for different Markov assumption orders in Sec. 5.1. Next, we evaluate the efficiency of our method concerning a varying number of fixed iterations (no stopping criterion is used) and for a varying inlier ratio, in Secs. 5.2 and 5.3, respectively. Section 5.4 offers results for 1) calibrated relative pose, 2) uncalibrated relative pose, 3) homography estimation, and 4) same as 1), 2) and 3) with the addition of local-optimization.

Evaluation metrics: We use the mean Average Accuracy (mAA) with thresholds at 5 and 10 degrees for the calibrated and uncalibrated relative pose problems, and at 5 and 10 pixels for homography estimation. We borrow the evaluation scripts for rotation, translation, and homography error metrics from the “RANSAC in 2020” tutorial package. Additionally, we show the average execution time.

Methods: We utilize two variations of our method: with and without pre-computed scores. Without pre-computed scores, we refer to our method as BANSAC, and we use a combination of BANSAC (Sec. 4.4) and SPRT [31] stopping criteria. When using pre-computed scores, we refer to our method as P-BANSAC, and we use a combination of BANSAC and PROSAC’s stopping criteria [16]. The stopping criteria for BANSAC and P-BANSAC are chosen for better accuracy (keeping a reasonable speed) and computational efficiency, respectively. In addition, BANSAC and P-BANSAC have some parameters that need to be set, namely the CPT values, which we kept fixed for all experiments.

As baselines, we use RANSAC [23] and NAPSAC [48] when not using pre-computed scores, and P-NAPSAC [4] and PROSAC [16] otherwise. Only the sampling and the stopping criterion vary between all methods. All the remaining RANSAC components are the same.

For all methods, at the end of the RANSAC cycle, the inliers of the best model are used to refine the final solution using a non-minimal solver, following a typical robust estimation pipeline. By default, no local optimization is run inside the RANSAC loop, unless we explicitly say so.

Experiments evaluating BANSAC using different stopping criteria and CPT parameters are present in the supplementary materials. BaySAC [8] is not shown in the paper since its results do not differ much from RANSAC. In the supplementary materials, we evaluate BANSAC against RANSAC and BaySAC using synthetic data.

Settings: For the calibrated relative pose problem, we use an error threshold of 1e−3 (normalized points), 1000 maximum iterations, and a confidence of 0.999. For the uncalibrated relative pose problem, we use an error threshold of 0.5 pixel, 10000 maximum iterations, and a confidence of 0.999. For the homography estimation problem, we use an error threshold value of 1 pixel, 1000 maximum iterations, and a confidence of 0.999. In all these problems, we set our proposed stopping criteria threshold τ to 0.01 in BANSAC and 0.1 in P-BANSAC (see Sec. 4.4).

Datasets: For the relative pose problems (essential and fundamental matrices estimation), we use the dataset “CVPR IMW 2020 PhotoTourism challenge”4, which has 12 scenes with around 100K pairs and 2 sequences with around 5K pairs (inlier rates vary between 30 and 60%, approximately). For the homography estimation problem, we use the EVD5 and HPatches [2], with 7 and 145 pairs of images, respectively (we used the validation set since the test set does not provide ground-truth). Matches and pre-computed weights were obtained with RootSIFT features and nearest-neighbor matching. Dataloaders were borrowed from the “RANSAC in 2020” tutorial webpage.

For both relative pose problems, results were obtained by taking, for each scene, the first 4K pairs and repeating each trial 5 times to ensure we have replicable accuracy and computational time readings for different runs. In the homography matrix results, we use all the available pairs and repeat each trial 10 times. For the experiments with different Markov order assumptions, varying number of fixed iterations, and varying inlier ratio, we use the sacre.coeur sequence entirely (around 5K pairs). We also use this sequence to tune our BANSAC parameters (stopping criterion and probability update model).

5.1 Different Markov assumption orders

We start the experiments by running an ablation study on the Markov assumption described in Sec. 4.2. We use the 1st, 2nd, and 3rd Markov assumption orders (the 2nd and 3rd are derived in the supplementary materials) and run BANSAC for a calibrated relative pose problem. Results

---

5http://cmp.felk.cvut.cz/wbs/ [August 18, 2023]
are shown in Tab. 2.

We observe that there are minor differences between the different Markov assumption orders. Rotation and translation errors are lower on the 1st-order assumption, and execution time is lower on the 3rd-order assumption. This occurs because probabilities converge more rapidly for higher orders, activating the stopping criterion faster. Prioritizing accuracy, in the following experiments, both BANSAC and P-BANSAC use the 1st-order Markov assumption in the probability modeling.

5.2 Varying number of fixed iterations

In this experiment, we aim to evaluate the efficiency of the sampling process. We run experiments fixing the number of iterations for all the methods; no early stopping criterion is used. We vary the number of iterations between 1000 and 10000 on an uncalibrated relative pose problem and measure the rotation error, translation error, and execution time. The results obtained are shown in Fig. 3.

As expected, overall, we verify that with an increasing number of iterations, the error decreases for all the methods. We observe that BANSAC and P-BANSAC have the lowest rotation and translation errors. Concerning execution time, although both our methods require additional steps to update the probabilities every iteration, we notice that the results are marginally the same.

5.3 Varying inlier ratio

The inlier ratio has a strong impact on RANSAC-based methods performance. To evaluate how each method performs for different inlier ratios, we vary the confidence threshold for filtering matches from 0.82 to 0.92, which gives us inlier rates ranging between around 60% to 30%, respectively. Results for rotation and translation errors and execution time for an uncalibrated relative pose problem are shown in Fig. 4.

In contrast to the baselines, we observe that the decrease in the inlier ratio (by filtering fewer matches) increases the accuracy for BANSAC and P-BANSAC. Concerning execution time, it increases in all methods similarly, except in PROSAC where it grows less.

5.4 Results

Next, we present results for three computer vision problems with and without a local-optimization step.

Calibrated relative pose: Results for the calibrated relative pose (essential matrix estimation) are shown in Tab. 3. They show that BANSAC is the most accurate method, followed by P-BANSAC across all the scenes. In execution time, P-BANSAC is the fastest method, followed by BANSAC, indicating that the pre-computed scores help BANSAC on exiting the RANSAC loop earlier.

Uncalibrated relative pose: Results for the uncalibrated relative pose (fundamental matrix estimation) are shown in Tab. 3. Similarly to the previous results, we observe that BANSAC is consistently the most accurate method, and P-BANSAC is the second most accurate in most scenes. In runtime, P-BANSAC is the fastest and BANSAC the second fastest. Figure 5 shows the probabilities values after 10, 100, 1000, and 10000 iterations in an image pair from the sacre_coeur sequence, using BANSAC.

Homography estimation: Results for homography estimation are shown in Tab. 3. In this experiment, P-BANSAC is the best in accuracy, with BANSAC being second best. In runtime, RANSAC is the fastest. We note that BANSAC requires an additional loop over all data points per iteration for updating scores (see the discussion in Sec. 6) when compared with RANSAC. This extra computational effort is visible when BANSAC does not exit the loop sufficiently earlier than the baselines, as shown in Fig. 3. Figure 1 shows the initial probabilities of some randomly chosen matches (all started at 0.5) and the updated probabilities after 10, 100, and 1000 iterations in an image pair from the HPatches dataset using BANSAC, demonstrating the probability updates over iterations. The ground truth and the estimated homography are marked in green and red, respectively.
Table 3. Experimental results for the calibrated relative pose, uncalibrated relative pose, and homography estimation. We compare BANSAC and P-BANSAC with RANSAC [23], NAPSAC [48], P-NAPSAC [4], and PROSAC [16]. We show results with and without the local-optimization step of LO-RANSAC [18].

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Without Local Optimization</th>
<th>With Local Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RANSAC</td>
<td>N-PAPSAC</td>
</tr>
<tr>
<td>Rotation mAA (5°) ↑</td>
<td>0.568</td>
<td>0.158</td>
</tr>
<tr>
<td>Rotation mAA (10°) ↑</td>
<td>0.645</td>
<td>0.226</td>
</tr>
<tr>
<td>Translation mAA (5°) ↑</td>
<td>0.422</td>
<td>0.0810</td>
</tr>
<tr>
<td>Translation mAA (10°) ↑</td>
<td>0.532</td>
<td>0.137</td>
</tr>
<tr>
<td>Avg. execution time [ms] ↓</td>
<td>25.5</td>
<td>40.1</td>
</tr>
</tbody>
</table>

| Homography mAA (5 px) ↑ | 0.467 | 0.206 | 0.460 | 0.464 | 0.500 | 0.478 | 0.514 | 0.499 | 0.517 | 0.511 | 0.526 | 0.501 |
| Homography mAA (10 px) ↑ | 0.559 | 0.308 | 0.557 | 0.560 | 0.589 | 0.571 | 0.595 | 0.572 | 0.600 | 0.595 | 0.610 | 0.589 |
| Avg. execution time [ms] ↓ | 14.4 | 26.3 | 12.2 | 12.5 | 9.80 | 7.88 | 15.4 | 18.4 | 13.6 | 13.5 | 11.4 | 8.87 |

6 Conclusion

This paper proposes BANSAC, a new sampling strategy for RANSAC using dynamic Bayesian networks. The method performs weighted sampling using probabilities for scoring data points. These probabilities are adaptively updated every iteration based on the successive inlier/outlier classifications. Additionally, we propose a stopping criterion using the estimated probabilities.

We present results on challenging real-world datasets showing that the proposed algorithm can learn the data inlier probability and that these probabilities can guide the sampling efficiently; the updates to RANSAC bring improvements in accuracy and execution time.

We note that BANSAC updates the scoring weights depending on the quality of the hypothesis, which is obtained from inlier counting. This means that we need an extra loop cycle every iteration for updating the scores. Although we do not observe a significant increase in computational cost compared to the best baselines, there is room for improvement. In future work, we plan to include a more efficient hypothesis prediction for incorporating the probability update in the inlier counting loop.

Acknowledgments

Valter Piedade was supported by the National Centre for Research and Development under the Smart Growth Operational Programme as part of project POIR.01.01.00-00-0102/20 and by the LARSyS–FCT Project UIDB/50009/2020. We thank all the reviewers and ACs for their valuable feedback.
References


BANSAC: A dynamic BAyesian Network for adaptive SAmple Consensus  
(SUPPLEMENTARY MATERIALS)

Valter Piedade  
Instituto Superior Técnico, Lisboa  
valter.piedade@tecnico.ulisboa.pt  

Pedro Miraldo  
Mitsubishi Electric Research Labs  
miraldo@merl.com  

These supplementary materials present new quantitative experiments (Appendix A) and some additional derivations and pseudo-code (Appendix B).

Contents

A Additional Experiments 11  
A.1 Calibrated relative pose . . . . . . . . . . . 11  
A.2 Uncalibrated relative pose . . . . . . . . . 11  
A.3 Synthetic data . . . . . . . . . . . . . . . . 12  
A.4 Ablation studies . . . . . . . . . . . . . . . 12  
A.4.1 Conditional probability tables . . . 13  
A.4.2 Weighted sampling . . . . . . . . . 13  
A.4.3 Stopping criteria . . . . . . . . . . . 14  
B Other Markov Assumptions 14  
B.1 Second-order Markov assumption . . . . . 14  
B.2 Third-order Markov assumption . . . . . 15  
B.3 Probability Updates Pseudo-code . . . . . 16

A Additional Experiments

This section provides additional experiments with real-world and synthetic data. Appendices A.1 and A.2 show results with each scene from the PhotoTourism dataset for the essential and fundamental matrices estimation. Appendix A.3 offers results for curve and circle fitting problems using synthetic data. Appendix A.4 contains ablation studies on the conditional probability tables (CPTs), sampling weights, and stopping criteria.

For both of the relative pose problem experiments (Appendices A.1 and A.2), we use the following scenes from the PhotoTourism dataset, with a matching score cutoff of 0.85: 0) brandenburg_gate with 43% inliers; 1) palace_of_westminster with 32% inliers; 2) westminster_abbey with 49% inliers; 3) taj_mahal with 57% inliers; 4) prague_old_town_square with 32% inliers; and 5) st_peters_square with 46% inliers; 6) buckingham_palace with 45% inliers; 7) colosseum_exterior with 36% inliers; 8) grand_place_brussels with 31% inliers; 9) notre_dame_front_facade with 46% inliers; 10) pantheon_exterior with 62% inliers; 11) temple_nara_japan with 60% inliers; 12) trevi_fountain with 33% inliers; and 13) sacre_coeur with 51% inliers. As in the main document, we use 4K pairs for each scene and repeated each trial 5 times.

All experiments presented in this document and on the main paper were performed on an Intel(R) Core(TM) i7-7820X CPU @ 3.60GHz processor.

A.1 Calibrated relative pose

This subsection presents additional results for the calibrated relative pose estimation problem, comparing BANSAC and P-BANSAC against the baselines (RANSAC, NAPSAC, P-NAPSAC, and PROSAC). As estimation parameters, we use an error threshold of $10^{-3}$ (normalized points), 1000 maximum iterations, and a confidence of 0.999, and set the BANSAC stopping criteria threshold $\tau$ to 0.01 in BANSAC and 0.1 in P-BANSAC (same parameters as in the main paper’s results). Results are shown in Tab. A.4.

We observe that, in accuracy, BANSAC and P-BANSAC are the best methods overall. In execution time, P-BANSAC is the best, with BANSAC second best in most scenes.

A.2 Uncalibrated relative pose

This subsection presents further results for the uncalibrated relative pose estimation problem, using the same baselines as in the previous subsection. As estimation parameters, we use an error threshold of 0.5, 10000 maximum iterations, and a confidence threshold of 0.999, and set the BANSAC stopping criteria threshold $\tau$ to 0.01 in BANSAC and 0.1 in P-BANSAC (same parameters as in the main paper’s results). Results are shown in Tab. A.4.

The results obtained are similar to those obtained in estimating the essential matrix. BANSAC is the best method in accuracy, followed by P-BANSAC in most scenes. Both are also the fastest methods overall.
### A.3 Synthetic data

We consider two simple problems: curve and circle-fitting. For each, we ran several experiments, varying the inlier rate between 15% and 50%. Each experiment has 300 data points ranging between $[-1, 1]$. Inliers are disturbed by a Gaussian noise of mean 0 and variance 0.02, and outliers are modeled by a uniform distribution with a maximum absolute value of 1.0. We evaluate BANSAC against RANSAC and BaySAC, which we implemented from scratch since no code is available. As estimation parameters, we use an error threshold value of 0.02, 3000 maximum iterations, and an estimation confidence of 0.99. In BANSAC, the initial probabilities $P^0$ are set to 0.5 for all data points, and the stopping criterion threshold $\tau$ is set to 0.01. We measure the root mean squared error (RMSE) of the geometric distance of points in the estimated model to the desired model and the number of iterations made. We present the mean values obtained after 1000 randomly generated trials. The results are shown in Fig. A.6.

We observe that BANSAC has an accuracy similar to or better than the baselines requiring significantly fewer iterations, even for lower inlier rates. Figure A.7 illustrates the BANSAC probability update for the curve fitting problem.

### A.4 Ablation studies

Next, we test different configurations for three components of the proposed algorithm. We present experiments using diverse conditional probability tables (CPTs) parameters, various activation functions for sampling, and components.
of different inlier rates.

Figure A.7. Example of BANSAC inlier probability update over iterations for a curve fitting problem (color code at the right). In the first row, from left to right, we show iterations 0, 2, and 4. Second row shows iteration 9, 11, and 14.

Table A.5. Conditional probability table of \( P(c_{n}^{k} | x_{n}^{k-1}) \) and \( P(x_{n}^{k} | c_{n}^{k}, x_{n}^{k-1}) \).

| Inlier | Inlier | \( \gamma(e^{k}) \) | Inlier | Outlier | \( 1 - \gamma(e^{k}) \) | Inlier | Outlier | \( P(x_{n}^{k} | c_{n}^{k}, x_{n}^{k-1}) \) |
|-------|-------|----------------|-------|---------|----------------|-------|---------|----------------|
|       |       |                |       |         |                |       |         |                |

Table A.6. Evaluation of BANSAC using different activation functions to define the parameters of the conditional probability table of \( P(c_{n}^{k} | x_{n}^{k-1}) \).

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Activation functions</th>
<th>( \gamma_{1}(\psi) )</th>
<th>( \gamma_{2}(\psi) )</th>
<th>( \gamma_{3}(\psi) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotation mAA (5°) ↑</td>
<td>( 0.836 )</td>
<td>( 0.793 )</td>
<td>( 0.790 )</td>
<td></td>
</tr>
<tr>
<td>Rotation mAA (10°) ↑</td>
<td>( 0.864 )</td>
<td>( 0.827 )</td>
<td>( 0.827 )</td>
<td></td>
</tr>
<tr>
<td>Translation mAA (5°) ↑</td>
<td>( 0.775 )</td>
<td>( 0.738 )</td>
<td>( 0.732 )</td>
<td></td>
</tr>
<tr>
<td>Translation mAA (10°) ↑</td>
<td>( 0.825 )</td>
<td>( 0.795 )</td>
<td>( 0.791 )</td>
<td></td>
</tr>
<tr>
<td>Avg. execution time [ms] ↓</td>
<td>( 13.9 )</td>
<td>( 17.4 )</td>
<td>( 17.5 )</td>
<td></td>
</tr>
</tbody>
</table>

\( P(c_{n}^{k} | x_{n}^{k-1}) \) are defined using a function \( \gamma(\cdot) \). We want this function to give a high probability to classifications made by good models and vice versa. Since the quality of a model is defined by its inlier ratio, we define this function as \( \gamma(e^{k}) \), where \( e^{k} \) is the inlier ratio at iteration \( k \). We test the following functions \( \gamma(e^{k}) \) (variations of these functions with different values were tested, we are listing the ones that produced the best results):

\[
\gamma_{1}(e^{k}) = \begin{cases} 
0.62 \cdot e^{k} + 0.5, & e^{k} < 0.7143 \\
0.2 \cdot e^{k} + 0.8, & \text{otherwise}
\end{cases}
\]  

(10)

\[
\gamma_{2}(e^{k}) = \begin{cases} 
0.5, & e^{k} < 0.3 \\
0.5 + e^{k-10(e^{k}-0.3)}, & \text{otherwise}
\end{cases}
\]  

(11)

\[
\gamma_{3}(e^{k}) = \tanh(3 \cdot e^{k}).
\]  

(12)

We present results using these functions with BANSAC and P-BANSAC in Tab. A.6.

We achieved the best results in accuracy and execution time when using \( \gamma_{1}(\psi) \). Based on these experiments, we decided to use \( \gamma_{1}(\psi) \) in \( P(c_{n}^{k} | x_{n}^{k-1}) \) in all the experiments. In the experiment shown in the main paper where we use the 2nd and 3rd orders of the Markov assumption, we use the CPTs shown in Tabs. A.7 and A.8, respectively. Similar to the CPT for the 1st order of the Markov assumption, the outlined parameters were obtained empirically.

A.4.2 Weighted sampling

In each iteration \( k \), we perform a sampling weighted by the probabilities estimated in the previous iteration \( P^{k-1} \). Instead of simply using the probability values directly, we test the use of activation functions to increase the range of weights. The goal is to increase the chances of choosing

A.4.1 Conditional probability tables

To infer \( P(x_{n}^{k} | \text{inlier} | c_{n}^{1:k}) \) we need to define the CPTs of \( P(c_{n}^{k} | x_{n}^{k-1}) \) and \( P(x_{n}^{k-1} | c_{n}^{k}, x_{n}^{k-1}) \) for the 1st order Markov assumption. We present these CPTs in Tab. A.5. The values for the CPT of \( P(x_{n}^{k-1} | c_{n}^{k}, x_{n}^{k-1}) \) were obtained empirically after testing different variations. We found that probability update is robust to slight variations of the reported parameters. The parameters of the CPT of
points with higher inlier probabilities. We tested the following activation functions (different functions were tested, and we are showing the ones that gave the best results):

\[
\rho_1(\psi) = \psi \cdot 100 \tag{13}
\]

\[
\rho_2(\psi) = \begin{cases} 
100 \cdot \psi & \psi > 0.3 \\
10 \cdot \psi & \text{otherwise}
\end{cases} \tag{14}
\]

\[
\rho_3(\psi) = \frac{100}{1 + e^{-10(\psi-0.5)}} \tag{15}
\]

\[
\rho_4(\psi) = 130 \cdot \tanh(\psi) \tag{16}
\]

where \(\psi \triangleq P(x^k_n \mid c_{n}^{1:k})\) is the estimated probability for the \(n^\text{th}\) data point at iteration \(k\). In Tab. A.9, we show results using these activations functions in BANSAC.

Of the tested functions, only \(\rho_1(\psi)\) is linear. This function equally converts all points probabilities to the desired sampling range. The remaining give greater weights to points with higher probabilities and vice versa. Overall, we observed that \(\rho_1(\psi)\) was the one that gave better results in accuracy and execution time. Based on these experiments, we chose to use \(\rho_1(\psi)\) in all other experiments.

### A.4.3 Stopping criteria

Finally, we assess the different kinds and combinations of stopping criteria we can use with our method: RANSAC, SPRT, PROSAC, BANSAC, and BANSAC combined with RANSAC, SPRT, or PROSAC. We show results using these different combinations of stopping criteria in Tab. A.10.

We observe that, although BANSAC stopping criteria ensure the output results are accurate, it is slow. However, when we combine our stopping condition with others, we consistently improve execution time with a slight drop in accuracy.

### B Other Markov Assumptions

In this section, we present new derivations on probability updates. We show how to get exact inferences for second and third-order Markov assumptions.

#### B.1 Second-order Markov assumption

For the second-order assumption, in addition to the conditional independence constraints presented in the main paper, we have

\[
x_n^j \perp x_n^{0:j-3}, x_n^{j-1}, x_n^{j-2}, c_n^j \quad \forall j, \tag{17}
\]

which means

\[
P(x_n^j \mid x_n^{0:j-1}, c_n^j) = P(x_n^j \mid x_n^{j-2:j-1}, c_n^j), \quad \forall j. \tag{18}
\]

Now, similar to what is done in the main document, by applying the chain rule of probabilities, we write the joint
\[
\begin{align*}
\text{Algorithm 2: BANSAC algorithm outline. In the algorithm below, I means inlier and O outlier.} \\
\text{Input – Data Q, and without pre-computed scores} \\
\text{Output – Best model } \theta^*, \text{ and } C^* \\
\begin{align*}
1 & \quad k \leftarrow 1; \\
2 & \quad \Phi^+_n \leftarrow 0.5, \forall n; \\
3 & \quad \Phi^-_n \leftarrow 0.5, \forall n; \\
4 & \quad P_n = \Phi^+_n + \Phi^-_n; \\
5 & \quad \text{while } k < K \text{ do} \\
6 & \quad \text{... Other RANSAC steps as listed in the main paper;} \\
7 & \quad \text{... for all } n \text{ do} \\
8 & \quad \text{if } c_n^k = I \text{ then} \\
9 & \quad \quad \Phi^+_n \leftarrow P(x_n^k = I, c_n^k = I, c_n^{k-1} = I)P(x_n^k = I, x_n^{k-1} = I) \Phi^+_n + P(x_n^k = I, c_n^k = I, x_n^{k-1} = 0)P(c_n^k = I, x_n^{k-1} = 0) \Phi^-_n; \\
10 & \quad \quad \Phi^-_n \leftarrow P(x_n^k = O, c_n^k = I, x_n^{k-1} = I)P(x_n^k = O, x_n^{k-1} = I) \Phi^-_n + P(x_n^k = O, c_n^k = I, x_n^{k-1} = 0)P(c_n^k = I, x_n^{k-1} = 0) \Phi^+_n; \\
11 & \quad \quad \Phi^+_n \leftarrow \Phi^+_n; \\
12 & \quad \quad \Phi^-_n \leftarrow \Phi^-_n; \\
13 & \quad \quad P_n = \Phi^+_n + \Phi^-_n; \\
14 & \quad \text{... Other RANSAC steps as listed in the main paper;} \\
15 & \quad \text{...}
\end{align*}
\]

As in the main document, a convenient result of Eq. 23 is that \( \tilde{\Phi}^1(\cdot) \) can be calculated recursively as follows:

\[
\tilde{\Phi}^1(x_n^k, c_n^1:k, c_n^{1:k}) = \\
\left\{ \begin{array}{ll}
\sum_{x_n^{k-2}} \phi(x_n^k, x_n^0:k-1, c_n^1:k) & j \geq 2 \\
\phi(x_n^0, C_n^1)P(x_n^0) & j = 1
\end{array} \right.
\]

For the second-order Markov assumption experiments, the only difference compared to what is described for the first-order is the use of the conditional probability in Eq. 21 as the sampling weights.

**B.2 Third-order Markov assumption**

For the third-order Markov assumption, we have the conditional independence constraints

\[
x_n^j \perp x_n^{0:j-4} \mid x_n^{j-1}, x_n^{j-2}, x_n^{j-3}, c_n^j \quad \forall j,
\]

which means

\[
P(x_n^j \mid x_n^{0:j-4}, c_n^j) = P(x_n^j \mid x_n^{j-3:j-1}, c_n^j) \quad \forall j.
\]

Again, by applying the chain rule of probabilities, we write the joint probability at iteration \( k \) as

\[
\tilde{\Phi}(x_n^0:k, c_n^{0:k}) = P(x_n^0) \prod_{j=1}^k \phi(x_n^j, c_n^j),
\]
where
\[
\overline{\phi}(x_n^k, c_n^k) = \begin{cases} 
P(x_n^i | x_{n}^{i-3:j-1}, c_n^i)P(c_n^i | x_{n}^{i-1}), & j \geq 3 \\
P(x_n^2 | x_{n}^{0:1}, c_n^2)P(c_n^2 | x_n^0), & j = 2 \\
P(x_n^1 | x_n^0, c_n^1)P(c_n^1 | x_n^0), & j = 1 
\end{cases}
\]  
(28)

Following the same steps shown in the main document, from Eqs. 27 and 28 the exact inference is given by
\[
P(x_n^k = \text{inlier} | C_n^{1:k}) = \alpha \overline{\Phi}(x_n^k = \text{inlier}, x_{n}^{0:k-1}, C_n^{1:k}),
\]  
(29)

where again \(\alpha\) is the normalization factor, and
\[
\overline{\Phi}(x_n^k, x_{n}^{0:k-1}, C_n^{1:k}) = \sum_{x_n^{k-1}} \sum_{x_n^{k-2}} \overline{\Phi}(x_n^k, x_n^{0:k-1}, C_n^{1:k}),
\]  
(30)

where
\[
\overline{\Phi}(x_n^k, x_n^{0:k-1}, C_n^{1:k}) = \\
\sum_{x_n^{k-3}} \overline{\phi}(x_n^k, C_n^k) \sum_{x_n^{k-4}} \overline{\phi}(x_n^{k-1}, C_n^{k-1}) \cdots \\
\sum_{x_n^4} \overline{\phi}(x_n^4, C_n^4) \sum_{x_n^3} \overline{\phi}(x_n^3, C_n^3) \overline{\phi}(x_n^2, C_n^2) \overline{\phi}(x_n^1, C_n^1) P(x_n^0).
\]  
(31)

Again, we can write Eq. 31 in a recursive way:
\[
\overline{\Phi}(x_n^k, x_n^{0:k-1}, C_n^{1:k}) = \\
\begin{cases} 
\sum_{x_n^{k-3}} \overline{\phi}(x_n^k, C_n^k) \overline{\Phi}(x_n^{k-1}, x_n^{0:k-2}, C_n^{1:k-1}) & k \geq 3 \\
\overline{\phi}(x_n^2, C_n^2) \overline{\phi}(x_n^1, C_n^1) P(x_n^0) & k = 2 \\
\overline{\phi}(x_n^1, C_n^1) P(x_n^0) & k = 1 
\end{cases}
\]  
(32)

Note for \(k = 1\), Eq. 31 does not sum in \(x_n^{k-2}\), since there is no such variable.

Finally, the weights for the sampling are taken from the inference in Eq. 29.

**B.3 Probability Updates Pseudo-code**

The probability updates derived in this code are easy to implement. An algorithm with the pseudo-code for the first-order Markov assumption is shown in Algorithm 2, in which probabilities are taken from Tab. A.5. The second and third-order constraints are derived similarly.