Efficient Algorithms for Synchronizing Localization Sensors under Interval Uncertainty*

Raphael Voges, Bernardo Wagner
The Real Time Systems Group (RTS)
Institute of Systems Engineering (ISE)
Leibniz University of Hannover
30167 Hannover, Germany
voges@rts.uni-hannover.de,wagner@rts.uni-hannover.de

Vladik Kreinovich†
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
vladik@utep.edu

Abstract

In this paper, we show that a practical need for synchronization of localization sensors leads to an interval-uncertainty problem. In principle, this problem can be solved by using the general linear programming algorithms, but this would take a long time – and this time is not easy to decrease, e.g., by parallelization since linear programming is known to be provably hard to parallelize. To solve the corresponding problem, we propose more efficient and easy-to-parallelize algorithms.

Keywords: interval computations, interval uncertainty, localization, robot localization, sensor synchronization, geodesy
AMS subject classifications: 65G30, 65G40, 65D19, 68T40, 70E60, 86A30

1 Formulation of the Practical Problem

Need for clock synchronization. In many real-life situations, to improve the accuracy, instead of using a single sensor, we fuse the results of several different sensors measuring related quantities. For example, in robotics applications, images generated by a videocamera (see, e.g., [9]) are fused with measurements performed by an Inertial

*Submitted: June 12, 2017; Revised: ; Accepted: .
†This work was performed when Vladik was a visiting researcher with the Geodetic Institute of the Leibniz University of Hannover, a visit supported by the German Research Foundation. This work was also supported in part by the US National Science Foundation (NSF) grant HRD-1242122 and by the German Research Foundation (DFG) as part of the DFG Research Training Group GRK2159 "i.c.sens – Integrity and Collaboration in dynamic sensor networks".

Measurement Unit (IMU) (see, e.g., [1, 2, 3]) to increase the accuracy and robustness of localization results.

To properly fuse the results of these two sensors, we need to make sure that both measurements are performed at the exact same moment of time. For this purpose, both sensors are equipped with clocks.

In principle, there exist super-precise clocks that we could use in both sensors, but such clocks are expensive. It is more economically efficient to use less expensive clocks. These less expensive clocks, however, need periodic calibration.

A general way to calibrate sensors – and, in particular, to synchronize clocks. In general, to calibrate a sensor, we compare its measurement results with the results of measuring the same quantity by using a much more accurate measuring instrument. In principle, we can use the same idea to synchronize the clocks: namely, we can calibrate each clock by comparing its results to the time measured by some super-precise clocks.

In spite of the fact that this method also uses super-precise clocks, this method uses these clocks only for a short period of time, during the calibration and is, therefore, much more economically efficient than using such clocks all the time.

Still, the need for using super-precise clocks makes this procedure rather expensive. So, a naturally question appears: can we avoid using super-precise clocks altogether?

Can we avoid using super-precise clocks: an idea. A natural idea is to synchronize the clocks by measuring the same entity with both sensors. In our case, we measure the orientation of our sensors relative to a common starting point. In general, this orientation measurement is three-dimensional. However, in this paper we only consider the most dominant orientation axis, and consequently measure the angle around this axis only. Then, by comparing the moments of time when both sensors detected each angle, we can find the match between the times shown by different clocks – and thus, synchronize the clocks.

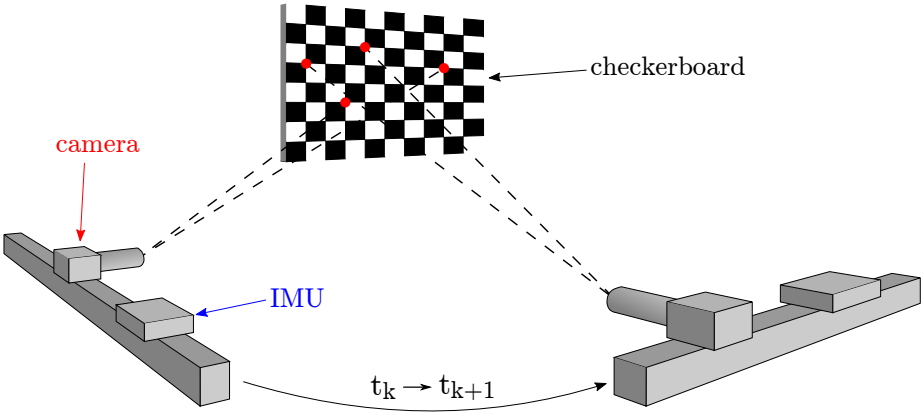

Figure 1: The main idea to generate common measurements using both camera and IMU.

We can perform different angle measurements by rotating the sensor system consisting of a camera and an IMU – which are rigidly attached to each other – in front

of a standard chessboard pattern (which is used for camera calibration). This can be done with a high speed of about 90 degrees per second. From each image taken by the camera, we can extract the angle describing the image's orientation relative to the chessboard pattern. Similarly, from each IMU measurement, we can extract the IMU's orientation relative to its starting position. We assume that the sensor system was spatially calibrated, so we know the exact spatial location of the sensors relative to each other. Thus, we know the relation between the two orientation angles at the same moment of time. Figure 1 shows the idea.

When the clocks on the two sensors are not perfectly synchronized, at the moment when the first sensor's clock shows time t, the clock on the second sensor shows a slightly different time $t' \neq t$. As a result, the angle measured by the second sensor when its clock shows t is slightly different from the angle measured by the first sensor when the clock on the first sensor shows the same time t. So, by comparing the angles measured by the two sensors when their clock readings are the same, we should be able to determine the difference between the clock readings and thus, synchronize the two sensors; see, e.g., [4, 10, 11].

The IMU unit can perform measurements every 10 milliseconds; the camera can produce images every 20 milliseconds. So, during a short period of time – of a few seconds – we can have hundreds of angle measurements performed by both sensors.

How are clocks related. Both clocks are reasonably uniform, but they can slightly differ in the starting point and in what each of them considers to be a unit of time.

Because of the difference in a starting point, time t' shown by the second clock differs from the time t measured by the first sensor by a constant $t' = t + b$, for some constant b known as the *offset*.

Because of the possible difference in time units, there is an additional change: namely, 1 second on the first clock may correspond to $a \approx 1$ second on the second one; the difference between a and 1 is known as the *drift*. As a result, the moment equal to t seconds on the first clock correspond, in general, to

$$t' = a \cdot t + b$$

seconds on the second clock, for some constants a and b.

Our goal is to find this relation between the two clocks, i.e., to find the corresponding coefficients a and b.

We need to take into account the accuracy of angle measurement. If both sensors measured the angles absolutely accurately, then we could simply match the moments of time when the sensors show the same measurement result. In practice, of course, angles are measured only with accuracy, this accuracy is usually about 1 degree.

What do we know about the accuracy of the angle measurements? In principle, we could calibrate the angle measurements, by comparing them with a much more accurate angle-measuring device. However, similarly to time calibration, this calibration is an expensive procedure. To avoid such expenses, we simply rely on the minimal data about the measurement error supplied by the manufacturer. This information often consists of simply one number Δ, the guaranteed upper bound on the measurement error; see, e.g., [8].

In this case, if at some moment of time t_k, the result of measuring the angle is α_k, this means that the actual value of the corresponding angle can be anywhere from $\alpha_k - \Delta$ to $\alpha_k + \Delta$.

Possibility to use linear interpolation. Based on the values α_k corresponding to moments of time t_k, we would like to estimate the values $\alpha(t)$ that would have been measured if we also performed the measurements at all intermediate moments of time $t \in (t_k, t_{k+1})$.

Each such moment of time t can be represented as $t = t_k + \Delta t$, where $\Delta t \le t_{k+1} - t_k$. As we have mentioned, we perform our measurements as frequently as possible. As a result, during the short time between the two measurements, we can safely ignore terms which are quadratic (or of higher order) in terms of Δt. Thus, we can expand the unknown dependence $\alpha(t) = \alpha(t_k + \Delta t)$ in Taylor series in Δt and keep only linear terms in this expansion:

$$\alpha(t_k + \Delta t) = \alpha(t_k) + c \cdot \Delta t = \alpha_k + c \cdot \Delta t,$$

where we denoted $c \stackrel{\text{def}}{=} \dfrac{\partial \alpha}{\partial t}_{|t=t_k}$. We do not know the dependence $\alpha(t)$, but we can determine the coefficient c by taking into account that the above formula should also be true for $\Delta t = t_{k+1} - t_k$. For this Δt, this formula takes the form $\alpha(t_{k+1}) = \alpha_k + c \cdot (t_{k+1} - t_k)$. Since we know the value $\alpha(t_{k+1}) = \alpha_{k+1}$, we can thus find c as

$$c = \frac{\alpha_{k+1} - \alpha_k}{t_{k+1} - t_k}$$

and hence, estimate $\alpha(t)$ for all $t \in (t_k, t_{k+1})$, as

$$\alpha(t) = \alpha_k + \frac{\alpha_{k+1} - \alpha_k}{t_{k+1} - t_k} \cdot (t - t_k).$$

This is a well-known linear interpolation formula. Thus, we can use linear interpolation to come up with a dependence $\alpha(t)$.

Let us apply linear interpolation to the results of both sensors. For the first sensor, based on the measured values α_k, we use linear interpolation to estimate the values $\alpha(t)$ as the values that we would have gotten if we measured the angle at all intermediate moments of time t.

Since the measurement accuracy is Δ, we can conclude that at each moment of time t, the actual value of the corresponding angle can be anywhere from $\underline{\alpha}(t) \stackrel{\text{def}}{=} \alpha(t) - \Delta$ to $\overline{\alpha}(t) \stackrel{\text{def}}{=} \alpha(t) + \Delta$.

Similarly, by processing the angles measured by the second sensor, we can estimate, for every moment of time t' (as measured by the second sensor), the interval $[\underline{\alpha}'(t'), \overline{\alpha}'(t')] = [\alpha'(t') - \Delta', \alpha'(t') + \Delta']$ that is guaranteed to contain the actual value of the angle at this moment of time; here, $\alpha'(t')$ is the interpolated value that we would have gotten if we performed the measurement by the second sensor at moment t', and Δ' is the measurement accuracy of the second sensor.

Now, we are ready to formulate the computational problem. Our main idea was to find, for each measurement result α_k of the first sensor, the moment of time when the actual angle was equal to α_k.

In the ideal case of exact measurement, this moment of time would be exactly equal to t_k. However, since there is measurement uncertainty, the measurement result is somewhat different from the actual value. Thus, the time when the actual value of the angle was equal to α_k is, in general, somewhat different from the moment of time t_k when the measured value was equal to α_k.

At each moment of time t, the actual value of the angle α can be anywhere from $\underline{\alpha}(t)$ to $\overline{\alpha}(t)$. In these terms, our goal is to describe all the moments of time t for which

$$\underline{\alpha}(t) \le \alpha_k \le \overline{\alpha}(t).$$

In the first cycle of rotation, when the angle increases, both functions $\underline{\alpha}(t)$ and $\overline{\alpha}(t)$ are increasing functions of time. Thus, the set of all moments t satisfying the above inequality is the interval $[\underline{t}_k, \overline{t}_k]$, where:

- $\underline{t}_k$ is the moment of time when $\overline{\alpha}(\underline{t}_k) = \alpha_k$, and
- $\overline{t}_k$ is the moment of time when $\underline{\alpha}(\overline{t}_k) = \alpha_k$.

Both functions $\underline{\alpha}(t)$ and $\overline{\alpha}(t)$ are piecewise linear, thus it is easy to find the corresponding moments of time.

These moments of time are described as measured by the first sensor's clock. In terms of the time shown by the second sensor's clock:

- the first-clock moment of time $\underline{t}_k$ takes the form $a \cdot \underline{t}_k + b$, and
- the first-clock moment of time $\overline{t}_k$ takes the form $a \cdot \overline{t}_k + b$.

Thus, in terms of the second-clock time, we conclude that the moment of time when the actual angle was equal to α_k is located somewhere on the interval $[a \cdot \underline{t}_k + b, a \cdot \overline{t}_k + b]$.

Similarly, for each k, by using the measurements performed by the second sensor, we can find the interval $[\underline{t}'_k, \overline{t}'_k]$ of all moments of time at which the actual angle was possible equal to α_k.

We can similarly find the intervals $[\underline{t}_k, \overline{t}_k]$ and $[\underline{t}'_k, \overline{t}'_k]$ when the angle reaches its turning point or decreases.

The actual (unknown) moment of time t_k^{actual} when the actual angle was equal to α_k belongs to both intervals $[\underline{t}'_k, \overline{t}'_k]$ and $[a \cdot \underline{t}_k + b, a \cdot \overline{t}_k + b]$. Thus, we can conclude that for every k, the intervals $[\underline{t}'_k, \overline{t}'_k]$ and $[a \cdot \underline{t}_k + b, a \cdot \overline{t}_k + b]$ have a non-empty intersection, see Figure 1.

Since we use the measurements of multiple rotations around our target, there are multiple moments of time when the actual angle was equal to α_k. However, the mapping between both intervals $[\underline{t}_k, \overline{t}_k]$ and $[\underline{t}'_k, \overline{t}'_k]$ is still unambiguous, because a is close to 1 and b is much smaller than the time that passes between two observations of the same actual angle α_k.

So, we arrive at the following problem.

2 Resulting Mathematical Problem

Resulting problem. We have n intervals $[\underline{t}_k, \overline{t}_k]$ and $[\underline{t}'_k, \overline{t}'_k]$, $k = 1, 2, \ldots, n$. We know that for some unknown values $a > 0$ and b, for all k from 1 to n, the intervals $[\underline{t}'_k, \overline{t}'_k]$ and $[a \cdot \underline{t}_k + b, a \cdot \overline{t}_k + b]$ have some points in common:

$$[\underline{t}'_k, \overline{t}'_k] \cap [a \cdot \underline{t}_k + b, a \cdot \overline{t}_k + b] \ne \emptyset.$$

We want to find the range of possible values of a and b under this condition.

Equivalent reformulation. One can easily see that the two intervals $[x, y]$ and $[x', y']$ have a non-empty intersection if and only if:

- the lower endpoint of the first interval is smaller than the upper endpoint of the second interval and, vice versa,

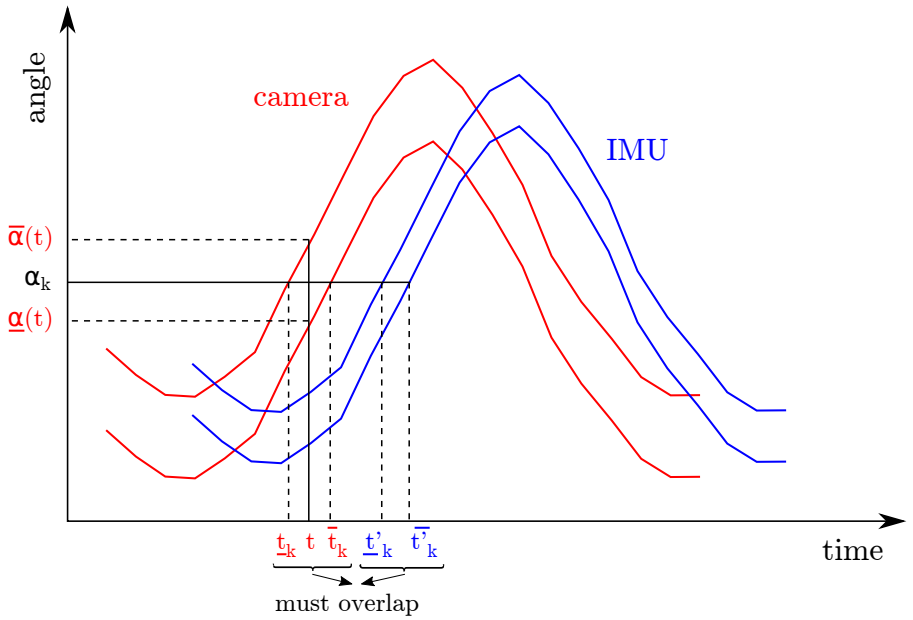

Figure 2: The main idea behind synchronization, as described in the main text

- the lower endpoint of the second interval is smaller than or equal to the upper endpoint of the second interval,

i.e., if and only if $x \le y'$ and $x' \le y$. Indeed:

- If these two intervals have a common point z, then $x \le z \le y'$ and $x' \le z \le y$, so we indeed have $x \le y'$ and $x' \le y$.
- Vice versa, if $x \le y'$ and $x' \le y$, then one can easily see that, e.g., the value $z = \max(x, x')$ belongs to both intervals.

Thus, the above condition can be reformulated as follows: for every k, we must have

$$a \cdot \underline{t}_k + b \le \overline{t}'_k \tag{1}$$

and

$$\underline{t}'_k \le a \cdot \overline{t}_k + b. \tag{2}$$

We need to find the smallest and the largest possible values of a and b under these constraints. In other words, we need to solve the following four optimization problems. Namely, we need to find:

- the smallest possible value $\underline{a}$ of the coefficient a under the constraints (1) and (2);
- the largest possible value $\overline{a}$ of the coefficient a under the constraints (1) and (2);
- the smallest possible value $\underline{b}$ of the coefficient b under the constraints (1) and (2); and

- the largest possible value $\overline{b}$ of the coefficient b under the constraints (1) and (2).

In principle, we can use linear programming. In all four problems, we are optimizing a function which is linear in the unknowns a and b under the constraints which are linear inequalities in terms of the unknowns. Thus, all four of our problems are a particular case of the general class of linear programming problems, for which efficient algorithms are known; see, e.g., [6].

Can we do it faster? While these algorithms for solving linear programming problems are reasonably efficient, their computation time grows as $O(n^{3.5})$, which for large n, may be large. Can we do it faster?

In general, a usual way to speed up computations is to use several processor working in parallel. However, this is only possible if an algorithm is parallelizable, and linear programming is known to be provably the hardest to parallelize; see, e.g., [7].

Thus, to speed up computations, we need to develop new faster algorithms for solving the above problems (ideally, algorithms that will be easy to parallelize).

What we do in this paper. In this paper, we propose faster $O(n^2)$ algorithms for solving our four optimization problems. All these algorithms are easily parallelizable.

Comment. These algorithms are based on the ideas used in [5, 12].

3 Analysis of the Problem

Bounds on a. Let us first derive the bounds on a. To find these bounds, we need to eliminate the unknown b from the constraints (1) and (2). In other words, we need to formulate the conditions on a under which constraints (1) and (2) are satisfied for some value b.

To find such condition, let us reformulate the inequalities (1) and (2) in terms of bounds on b. We can do it if in each inequality, we keeping the unknown b in the same side of the inequality as it originally was, and move all other terms to the other side. As a result, we get the following equivalent reformulations of the inequalities (1) and (2):

$$b \le \overline{t}'_k - a \cdot \underline{t}_k \tag{3}$$

and

$$\underline{t}'_k - a \cdot \overline{t}_k \le b. \tag{4}$$

Similarly to the above argument, the existence of a value b that satisfies all these inequalities is equivalent to requiring that every lower bound for b is smaller than each upper bound. Indeed:

- If such a b exists, then this condition is also satisfied.
- Vice versa, if all these conditions are satisfied, then we can take, e.g., b equal to the largest of the lower bounds.

Thus, the existence of the values a and b for which inequalities (1) and (2) (or, equivalently, inequalities (3) and (4)) are satisfied is equivalent to the following inequalities that should hold for all k and ℓ:

$$\underline{t}'_k - a \cdot \overline{t}_k \le \overline{t}'_\ell - a \cdot \underline{t}_\ell. \tag{5}$$

By moving all the terms related to a to the right-hand side and all other terms to the left-hand side, we get an equivalent inequality:

$$\underline{t}'_k - \overline{t}'_\ell \le a \cdot (\overline{t}_k - \underline{t}_\ell). \tag{6}$$

If the coefficient at a is positive, we can divide by this coefficient and get a lower bound on a:

$$\frac{\underline{t}'_k - \overline{t}'_\ell}{\overline{t}_k - \underline{t}_\ell} \le a. \tag{7}$$

The value a should be larger than or equal to all these lower bounds, so it must be larger than or equal to the largest of them. Thus, we get the following lower endpoint $\underline{a}$ for the interval $[\underline{a}, \overline{a}]$ of possible values of a:

$$\underline{a} = \max_{k,\ell:\ \overline{t}_k > \underline{t}_\ell} \frac{\underline{t}'_k - \overline{t}'_\ell}{\overline{t}_k - \underline{t}_\ell}. \tag{8}$$

When the coefficient at a in the formula (6) is negative, then by dividing by this coefficient, we get an upper bound on a:

$$a \le \frac{\underline{t}'_k - \overline{t}'_\ell}{\overline{t}_k - \underline{t}_\ell}. \tag{9}$$

The value a should be smaller than or equal to all these upper bounds, so it must be smaller than or equal to the smallest of them. Thus, we get the following upper endpoint $\overline{a}$ for the interval $[\underline{a}, \overline{a}]$ of possible values of a:

$$\overline{a} = \min_{k,\ell:\ \overline{t}_k < \underline{t}_\ell} \frac{\underline{t}'_k - \overline{t}'_\ell}{\overline{t}_k - \underline{t}_\ell}. \tag{10}$$

(If the coefficient at a is 0, we do not get any constraint on a at all.)

Let us now find the bounds on b. To get the bounds on b, we must now eliminate a from the inequalities (1) and (2). For this purpose, let us reformulate the inequalities (1) and (2) in the equivalent form, as bounds on a. To do thus, in each of the inequalities (1) and (2), we keep the term containing a in one side, move all the other terms to the other side, and then then divide both side of the inequality by the positive coefficient at a. As a result, we get the following equivalent bounds:

$$a \le \frac{\overline{t}'_k}{\underline{t}_k} - \frac{1}{\underline{t}_k} \cdot b \tag{11}$$

and

$$\frac{\underline{t}'_k}{\overline{t}_k} - \frac{1}{\overline{t}_k} \cdot b \le a \tag{12}$$

The existence of a value a that satisfies all these inequalities is equivalent to requiring that every lower bound for a is smaller than each upper bound, i.e., that for every k and ℓ, we have

$$\frac{\underline{t}'_k}{\overline{t}_k} - \frac{1}{\overline{t}_k} \cdot b \le \frac{\overline{t}'_\ell}{\underline{t}_\ell} - \frac{1}{\underline{t}_\ell} \cdot b. \tag{13}$$

We can simplify this inequality if we multiply both sides by a positive number $\overline{t}_k \cdot \underline{t}_\ell$, then we get:

$$\underline{t}'_k \cdot \underline{t}_\ell - \underline{t}_\ell \cdot b \le \overline{t}_k \cdot \overline{t}'_\ell - \overline{t}_k \cdot b. \tag{14}$$

By moving all the terms containing b to the left-hand side and all the other terms to the right-hand side, we get an equivalent inequality

$$(\overline{t}_k - \underline{t}_\ell) \cdot b \le \overline{t}_k \cdot \overline{t}'_\ell - \underline{t}'_k \cdot \underline{t}_\ell. \tag{15}$$

When the coefficient at b in the formula (15) is positive, then by dividing by this coefficient, we get an upper bound on b:

$$b \le \frac{\overline{t}_k \cdot \overline{t}'_\ell - \underline{t}'_k \cdot \underline{t}_\ell}{\overline{t}_k - \underline{t}_\ell}. \tag{16}$$

The value b should be smaller than or equal to all these upper bounds, so it must be smaller than or equal to the smallest of them. Thus, we get the following upper endpoint $\overline{b}$ for the interval $[\underline{b}, \overline{b}]$ of possible values of b:

$$\overline{b} = \min_{k,\ell:\ \overline{t}_k > \underline{t}_\ell} \frac{\overline{t}_k \cdot \overline{t}'_\ell - \underline{t}'_k \cdot \underline{t}_\ell}{\overline{t}_k - \underline{t}_\ell}. \tag{17}$$

When the coefficient at b in the formula (15) is negative, then by dividing by this coefficient, we get a lower bound on b:

$$b \ge \frac{\overline{t}_k \cdot \overline{t}'_\ell - \underline{t}'_k \cdot \underline{t}_\ell}{\overline{t}_k - \underline{t}_\ell}. \tag{18}$$

The value b should be greater than or equal to all these lower bounds, so it must be greater than or equal to the largest of them. Thus, we get the following lower endpoint $\underline{b}$ for the interval $[\underline{b}, \overline{b}]$ of possible values of b:

$$\underline{b} = \max_{k,\ell:\ \overline{t}_k < \underline{t}_\ell} \frac{\overline{t}_k \cdot \overline{t}'_\ell - \underline{t}'_k \cdot \underline{t}_\ell}{\overline{t}_k - \underline{t}_\ell}. \tag{19}$$

Thus, we arrive at the following algorithms:

4 Resulting Algorithms

The description of the algorithms. The range of possible values of a is $[\underline{a}, \overline{a}]$, where:

$$\underline{a} = \max_{k,\ell:\ \overline{t}_k > \underline{t}_\ell} \frac{\underline{t}'_k - \overline{t}'_\ell}{\overline{t}_k - \underline{t}_\ell} \tag{8}$$

and

$$\overline{a} = \min_{k,\ell:\ \overline{t}_k < \underline{t}_\ell} \frac{\underline{t}'_k - \overline{t}'_\ell}{\overline{t}_k - \underline{t}_\ell}. \tag{10}$$

The range of possible values of b is $[\underline{b}, \overline{b}]$, where

$$\underline{b} = \max_{k,\ell:\ \overline{t}_k < \underline{t}_\ell} \frac{\overline{t}_k \cdot \overline{t}'_\ell - \underline{t}'_k \cdot \underline{t}_\ell}{\overline{t}_k - \underline{t}_\ell} \tag{19}$$

and

$$\overline{b} = \min_{k,\ell:\ \overline{t}_k > \underline{t}_\ell} \frac{\overline{t}_k \cdot \overline{t}'_\ell - \underline{t}'_k \cdot \underline{t}_\ell}{\overline{t}_k - \underline{t}_\ell}. \tag{17}$$

These algorithms are indeed faster than logic programming. In all four formulas, we need to consider all possible pairs (k, ℓ) of indices. There are n^2 such pairs, so we need $O(n^2)$ computational steps. This is indeed smaller than the time $O(n^{3.5})$ needed to linear programming.

Our algorithms are easy to parallelize. Our algorithm require finding minima and maxima over all possible pairs of indices. If we have p processors, we can:

- divide all n^2 pairs into p groups,
- use each processor to compute the min and max over pairs from the corresponding group, and then
- take min and max of the resulting p intermediate results.

Thus, our algorithms are indeed easy to parallelize.

5 Conclusions and Future Work

Given one-dimensional measurements of the same entity from two sensors, this paper introduces an efficient algorithm to estimate the offset and the drift between both sensor clocks. Although our problem formulation and resulting algorithm are generic, we show how such measurements can be generated using the example of camera and IMU synchronization.

In the future, we aim to formulate the synchronization problem in the context of constraint programming over dynamical systems and use so-called *tubes* [1] to efficiently work with real measurement data. Besides, we plan to extend the approach to different sensor combinations.

References

[1] J. Farrell, *Aided Navigation: GPS with High Rate Sensors*, McGraw-Hill, New York, 2008.

[2] J. Farrell and M. Barth, *The Global Positioning System and Inertial Navigation*, McGraw-Hill, New York, 1998.

[3] P.D. Groves, *Principles of GNSS, Inertial and Multisensor Integrated Navigation Systems*, Artech House, Norwood, Massachusetts, 2008.

[4] J. Kelly, N. Roy, and G. S. Sukhatme, "Determining the time delay between inertial and visual sensor measurements", *IEEE Transactions on Robotics*, 2014, Vol. 30, No. 6, pp. 1514–1523.

[5] S. Kumkov, V. Kreinovich, and A. Pownuk, "In System Identification, Interval (and Fuzzy) Estimates Can Lead to Much Better Accuracy than the Traditional Statistical Ones: General Algorithm and Case Study", *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics SMC'2017*, Banff, Canada, October 5–7, 2017.

[6] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, Springer, Cham, Switzerland, 2016.

[1] `http://www.simon-rohou.fr/research/tubex-lib`

[7] C. H. Papadimitriou, *Computational Complexity*, Pearson, Boston, Massachusetts, 1993.

[8] S. G. Rabinovich, *Measurement Errors and Uncertainty: Theory and Practice*, Springer Verlag, Berlin, 2005.

[9] R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer Verlag, London, 2011.

[10] R. Voges, C.S. Wieghardt, and B. Wagner, "Finding Timestamp Offsets for a Multi-Sensor System Using Sensor Observations", *Photogrammetric Engineering & Remote Sensing*, 2018, Vol. 84, No. 6, pp. 357–366.

[11] R. Voges, and B. Wagner, "Timestamp Offset Calibration for an IMU-Camera System Under Interval Uncertainty", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, October 1–5, 2018.

[12] K. Worden, R. Osegueda, C. Ferregut, S. Nazarian, D. L. George, M. J. George, V. Kreinovich, O. Kosheleva, and S. Cabrera, "Interval Methods in Non-Destructive Testing of Material Structures", *Reliable Computing*, 2001, Vol. 7, No. 4, pp. 341–352.