Technical Report: Algorithm Implementation Details and Extended Numerical Results

Francesco Montorsi*[1], Fabrizio Pancaldi†[2], and Giorgio M. Vitetta‡[1]

[1]Department of Information Engineering, University of Modena and Reggio Emilia, Reggio Emilia, Italy

[2]Department of Science and Methods for Engineering, University of Modena and Reggio Emilia, Reggio Emilia, Italy

May 11, 2013

Contents

*francesco.montorsi@unimore.it

†fabrizio.pancaldi@unimore.it

‡giorgio.vitetta@unimore.it

1 Introduction

This Technical Report provides some details about 1) the algorithms discussed in [1] and 2) some numerical results not included in [1] for space limitations and regarding the performance provided by various localization algorithms.

2 Covering Generation Algorithm

All the map-aware algorithms discussed in [1] employ a *rectangular covering* as definition of the integration/search domain (see [1, Table I]). In this Section a simple, but effective, algorithm to generate rectangular coverings is illustrated.

As mentioned in [1], there is no one-to-one mapping between the support $\mathcal{R}$ of a (rectangular) map and its covering $\{\mathcal{R}_k\}$; however, various algorithms are already available in the technical literature to partition a generic polygon into a set of rectangles (e.g., see [2,3]) and, in particular, to generate a partition consisting of a *minimal* number of rectangles, i.e., to generate the so-called *minimal non-overlapping covering* (MNC) [4–6]. In typical localization systems the MNC can be computed offline, since $\mathcal{R}$ is usually time invariant. Moreover, building plans are usually characterized by rectangular features (walls, rooms, corridors, etc). For these reasons, instead of employing one of the optimized (but computationally intensive) algorithms proposed in [4–6], a simpler algorithm, summarized in Alg. 1, has been adopted in [1]. Note that this algorithm can handle maps containing "holes" (i.e., regions that are enclosed by the domain $\mathcal{R}$ but do not belong to it) and that the input data it requires can be easily extracted from the maps of typical building floors. It is based on the assumption that both the map and its "holes" can be approximated by the union of multiple rectangles, all having parallel sides.

Note that, in particular, Alg. 1 is fed by the list $\{(x_n, y_n)\}_{n=1}^{N_v}$ of the N_v vertices describing the map support $\mathcal{R}$ (and by an optional lists of vertices describing the "holes"). An example of map support $\mathcal{R}$ and of the notation that can be used to describe it is shown in Fig. 1. Alg. 1 operates by first creating the so called *dense non-overlapping covering* (DNC) (lines 1-15) and then, through iterative merge operations, by transforming it into the MNC (lines 16-27). Thanks to the assumption that all sides of $\mathcal{R}$ and its holes are parallel, the check accomplished at line 7 is simple (in fact, to understand if $\mathcal{R}_{i,j} \subset \mathcal{R}$, a simple check on the rectangle center can be employed).

Algorithm 1 Simplified MNC and DNC generation for rectangular maps from the vertex list (used by map-aware estimators)

```
Require: vertex list $\{(x_n, y_n)\}_{n=1}^{N_v}$ of the polygons modelling the outer edges of $\mathcal{R}$
Require: vertex lists $\left\{\mathbf{p}_n^{(m)} \in \mathbb{R}^2\right\}_{n=1}^{N_h^{(m)}}$, with $m = 1, \ldots, M_h$, of the polygons modelling the $M_h$
    “holes” of $\mathcal{R}$ (where $N_h^{(m)}$ denotes the number of vertices of the $m$-th “hole”).
Ensure: $\{\mathcal{R}_k^{\text{DNC}}\}$ and $\{\mathcal{R}_k^{\text{MNC}}\}$ contain a DNC and the approximated MNC for $\mathcal{R}$, respectively.
$\mathbf{x}_g \leftarrow$ sorted $\{x_n\}$ values
$\mathbf{y}_g \leftarrow$ sorted $\{y_n\}$ values
                                        ▷ $\mathbf{x}_g$ and $\mathbf{y}_g$ define a non-uniformly spaced grid on $\mathcal{R}$
$k \leftarrow 1$
for $i \in \{1, ..., |\mathbf{x}_g| - 1\}$, $j \in \{1, ..., |\mathbf{y}_g| - 1\}$ do
    $\mathcal{R}_{i,j} \leftarrow [\mathbf{x}_{g,i}; \mathbf{x}_{g,i+1}] \times [\mathbf{y}_{g,j}; \mathbf{y}_{g,j+1}]$
    if centre$\{\mathcal{R}_{i,j}\} \in$ polygons defined by $\{(x_n, y_n)\}_n$ and
            centre$\{\mathcal{R}_{i,j}\} \notin$ polygons defined by $\left\{\mathbf{p}_n^{(m)}\right\}_{n=1}^{N_h^{(m)}}$, with $m = 1, \ldots, M_h$ then
        $\mathcal{R}_k^{\text{DNC}} \leftarrow \mathcal{R}_{i,j}$
        $\mathcal{R}_k^{\text{MNC}} \leftarrow \mathcal{R}_{i,j}$
                        ▷ $\mathcal{R}_{i,j}$ is a rectangle which needs to be included in the DNC and the MNC
        $k \leftarrow k + 1$
                        ▷ $k$ counts the rectangles placed until now in the DNC and MNC
    end if
end for
                                        ▷ DNC is ready; now it is optimized to produce the MNC
$k \leftarrow 1$
while $k < |\{\mathcal{R}_k^{\text{MNC}}\}|$ do
    $\mathcal{I} \leftarrow$ a value of the set $\{i \neq k \mid \mathcal{R}_i^{\text{MNC}} \bigcup \mathcal{R}_k^{\text{MNC}}$ is a rectangle$\}$
    if $\mathcal{I} \equiv \emptyset$ then
        $k \leftarrow k + 1$
    else
                        ▷ $\mathcal{I}$ contains the index of a rectangle that can be merged with $\mathcal{R}_k^{\text{MNC}}$
        $\mathcal{R}_k^{\text{MNC}} \leftarrow \mathcal{R}_k^{\text{MNC}} \bigcup \left(\bigcup_{i \in \mathcal{I}} \mathcal{R}_i^{\text{MNC}}\right)$
                        ▷ the size of the MNC can be reduced, thanks to the merge operation
        remove $\{\mathcal{R}_i^{\text{MNC}}, i \in \mathcal{I}\}$ from the set $\{\mathcal{R}_k^{\text{MNC}}\}$
    end if
end while
```

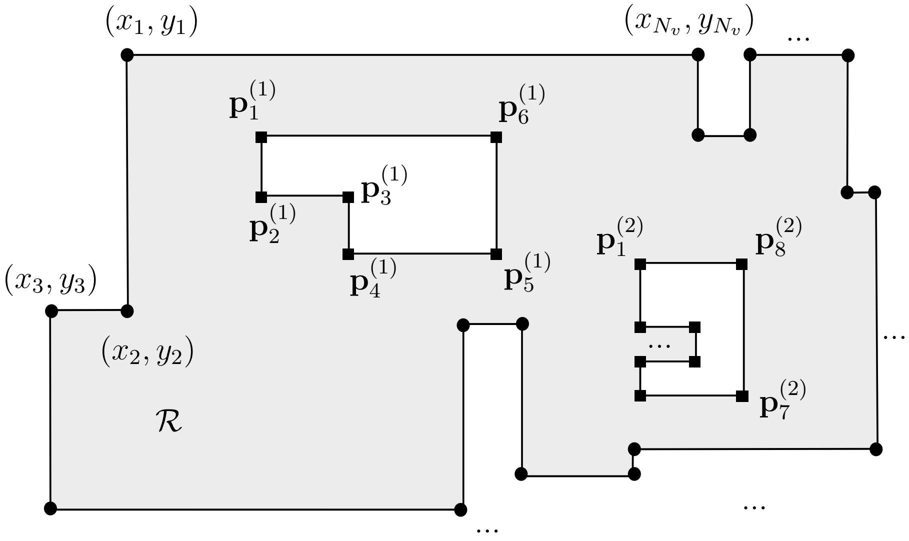

Figure 1: The support $\mathcal{R}$ of a map identified by $N_v = 18$ vertices suitable for processing with Alg. 1. Two "holes" are present in the support $\mathcal{R}$ and can be described by two polygons composed by $N_h^{(1)} = 6$ and $N_h^{(2)} = 8$ vertices, respectively.

Once the DNC is obtained, the MNC (which is identical to the DNC until line 18) is produced by searching, for each rectangle, all the neighbours that, once merged with the considered rectangle, will generate another rectangle (since, generally speaking, merging two rectangles does not produce another rectangle).

The MNC and DNC coverings generated for the same building map are thus related to the map support $\mathcal{R}$ as

$$\mathcal{R} \simeq \bigcup_{k=1}^{N_r^{\text{MNC}}} \mathcal{R}_k^{\text{MNC}} = \bigcup_{k=1}^{N_r^{\text{DNC}}} \mathcal{R}_k^{\text{DNC}} \quad (1)$$

where N_r^{MNC} (N_r^{DNC}) is the number of rectangles in the selected MNC (DNC) of $\mathcal{R}$.

For some examples of building maps and associated values obtained for N_r^{MNC} (N_r^{DNC}) the reader is referred to the numerical results section of [1].

3 The Map-Aware MMSE Cubature Implementation

In this section some details about the implementation of the *minimum mean square error* (MMSE) estimator based on cubature integration formulas and mentioned in [1] are provided. As shown in in [1], the MMSE estimator for the considered map-aware localization problem can be expressed as [1, Eq. (11)]

$$\hat{\mathbf{p}}_{\text{MMSE}}(\mathbf{z}) \simeq \frac{\begin{bmatrix} \iint_{\mathcal{R}^{(\mathcal{Z})}} x\, h(\mathbf{z},\mathbf{p}) d\mathbf{p} \\ \iint_{\mathcal{R}^{(\mathcal{Z})}} y\, h(\mathbf{z},\mathbf{p}) d\mathbf{p} \end{bmatrix}}{\iint_{\mathcal{R}^{(\mathcal{Z})}} h(\mathbf{z},\mathbf{p}) d\mathbf{p}} \tag{2}$$

where

$$h(\mathbf{z},\mathbf{p}) \triangleq \exp\left[q(\mathbf{p},\mathbf{z})\right] . \tag{3}$$

$$q(\mathbf{p},\mathbf{z}) \triangleq -\frac{1}{2}\sum_{i\in\mathcal{Z}}\left(\frac{z_i - d_i(\mathbf{p}) - \mu_{b,i}(\mathbf{p})}{\sigma_{n,i}(\mathbf{p})}\right)^2 \tag{4}$$

The integrations required by the MMSE estimator can be approximated using the so called *cubature formulas* [7–11]. In particular, a cubature formula allows to approximate the integral of the function $g : \mathbb{R}^N \to \mathbb{R}$ over a domain $\Omega \subset \mathbb{R}^N$ as

$$\int \cdots \int_{\Omega} g(\boldsymbol{\alpha}) d\boldsymbol{\alpha} = \sum_{j=1}^{N_n} w_j g\left(\boldsymbol{\alpha}^{(j)}\right) + R(g) \tag{5}$$

where $\boldsymbol{\alpha}^{(j)} \in \mathbb{R}^N$ and w_j are the j-th *node* and the j-th *weight*, respectively, N_n is the *order* of the cubature formula (i.e., the number of its nodes) and $R(g)$ is the remainder [8]. If $g(\cdot)$ is a polynomial function of degree d, cubature formulas having a minimal number of nodes $N_n(d)$ and ensuring that $R(g) = 0$ are available for different types of multidimensional domains Ω (e.g., see [7–11]); these formulas have been successfully employed in multidimensional filtering applications (e.g., see [12, 13]). However, if $g(\cdot)$ is not a polynomial function, known cubature formulas always lead to approximate results. In our work the results illustrated in [9], which provides "almost minimal" cubature formulas for polynomial functions of degree belonging to the interval 1-55 and for $\Omega = [-1; 1] \times [-1; 1]$, have been exploited. Such formulas can be easily adapted to multidimensional integration over the domain $\mathcal{R}^{(\mathcal{Z})}$ since, thanks to (1), any integral over $\mathcal{R}^{(\mathcal{Z})}$ can be expressed as the sum of N_r distinct integrals over rectangular domains. In fact, if $\mathcal{R}_k \triangleq [l_k; r_k] \times [b_k; t_k]$ denotes the k-th rectangle of the selected covering, it can be easily shown that

$$\iint_{\mathcal{R}_k} g(\mathbf{p})d\mathbf{p} = (\det \mathbf{L}_k) \iint_{\Omega} g(\mathbf{c}_k + \mathbf{L}_k \boldsymbol{\alpha})d\boldsymbol{\alpha} \tag{6}$$

where $\boldsymbol{\alpha} \triangleq \mathbf{L}_k^{-1}(\mathbf{p} - \mathbf{c}_k)$, $\mathbf{L}_k \triangleq \frac{1}{2}\,\mathrm{diag}\,\{r_k - l_k, t_k - b_k\}$ and $\mathbf{c}_k \triangleq \frac{1}{2}\,[l_k + r_k; b_k + t_k]^T$. Then, substituting (5) in the RHS of (6) yields the approximate integration formula

$$\iint_{\mathcal{R}_k} g(\mathbf{p})d\mathbf{p} \simeq (\det \mathbf{L}_k) \sum_{j=1}^{N_{n,k}} w_j g\left(\mathbf{c}_k + \mathbf{L}_k \boldsymbol{\alpha}^{(j)}\right) \tag{7}$$

Here, to ensure a proper accuracy in numerical integration, the number of nodes is heuristically selected as $N_{n,k} = N_n\left(\lfloor m\mathcal{A}_{\mathcal{R}_k} + d_{min} \rfloor\right)$, where $N_n(\cdot)$ is the function mapping the degree d of a polynomial into the number of integration nodes provided in [9], $\lfloor m\mathcal{A}_{\mathcal{R}_k} + d_{min} \rfloor$ is the "pseudo" polynomial degree associated with $g(\cdot)$, m and d_{min} are constants (their values are listed in Table [1, Table 1]) and $\mathcal{A}_{\mathcal{R}_k}$ is the area of $\mathcal{R}_k$. Note that the criterion proposed for the selection of $N_{n,k}$ ensures that high cubature orders are selected for large rectangles, so that the integration accuracy is enhanced in portions of the map where the integrand function may exhibit strong fluctuations (at the price, however, of an increased computational load). The cubature formula (7) can be exploited to evaluate the three integrals appearing in the RHS of (2). This leads to

$$\hat{\mathbf{p}}_{\mathrm{MMSE}_c}(\mathbf{z}) \simeq \frac{\begin{bmatrix} \displaystyle\sum_{k=1}^{N_r} \sum_{j=1}^{N_{n,k}} w_{k,j} \alpha_x^{(k,j)} h\left(\mathbf{z}, \boldsymbol{\alpha}^{(k,j)}\right) \\ \displaystyle\sum_{k=1}^{N_r} \sum_{j=1}^{N_{n,k}} w_{k,j} \alpha_y^{(k,j)} h\left(\mathbf{z}, \boldsymbol{\alpha}^{(k,j)}\right) \end{bmatrix}}{\displaystyle\sum_{k=1}^{N_r} \sum_{j=1}^{N_{n,k}} w_{k,j} h\left(\mathbf{z}, \boldsymbol{\alpha}^{(k,j)}\right)} \tag{8}$$

where $\boldsymbol{\alpha}^{(k,j)} \triangleq \mathbf{c}_k + \mathbf{L}_k \boldsymbol{\alpha}^{(j)} = [\alpha_x^{(k,j)}, \alpha_y^{(k,j)}]^T$ is the j-th node associated with $\mathcal{R}_k$ (see (1)) and $w_{k,j} \triangleq w_j \cdot \det \mathbf{L}_k$ is the corresponding weight.

Note that:

- all the sums appearing in the RHS of (8) require the evaluation of the same quantities[1] $\left\{h\left(\mathbf{z}, \boldsymbol{\alpha}^{(k,j)}\right)\right\}$;
- MMSE$_c$ requires the evaluation of (8) only and is non-iterative;
- its approximate complexity is $\mathcal{O}\left(\bar{N}_s \cdot N_{obs} \cdot N_{eval}\right)$ with $N_{eval} = \bar{N}_n \cdot N_r$, where $\bar{N}_n$ is the average number of nodes selected for integration inside each of the N_r rectangles.

In a practical implementation of (8) a final important issue needs to be considered; in fact, the quantities $\left\{h\left(\mathbf{z}, \boldsymbol{\alpha}^{(k,j)}\right)\right\}$ involve exponential operations (see (3)), so that their "dynamic range" can be very large (i.e., these quantities may assume very small values in some cases and very large values in other cases); this may lead to underflow and overflow problems depending on the hardware used to carry out these computations (in particular, the relevance of such problems depends on the number of bits used for *floating point operations* (FLOPs)). In our experience, a simple method to mitigate such a problem consists in

1. estimating the average $C \triangleq \mathbb{E}_{\mathbf{p},\mathbf{z}}\left\{q(\mathbf{p}, \mathbf{z})\right\}$, where $q(\mathbf{p}, \mathbf{z})$ (4) is the argument of the exponential operator of $h(\mathbf{p}, \mathbf{z})$ (3);
2. exploiting the simple analytical result

$$\begin{aligned}\hat{\mathbf{p}}_{\text{MMSE}}(\mathbf{z}) &\simeq \frac{\begin{bmatrix} \iint_{\mathcal{R}(\mathbf{z})} x\, h(\mathbf{z}, \mathbf{p}) d\mathbf{p} \\ \iint_{\mathcal{R}(\mathbf{z})} y\, h(\mathbf{z}, \mathbf{p}) d\mathbf{p} \end{bmatrix}}{\iint_{\mathcal{R}(\mathbf{z})} h(\mathbf{z}, \mathbf{p}) d\mathbf{p}} = \frac{\begin{bmatrix} \iint_{\mathcal{R}(\mathbf{z})} x \, \exp\left[q(\mathbf{z}, \mathbf{p})\right] d\mathbf{p} \\ \iint_{\mathcal{R}(\mathbf{z})} y \, \exp\left[q(\mathbf{z}, \mathbf{p})\right] d\mathbf{p} \end{bmatrix}}{\iint_{\mathcal{R}(\mathbf{z})} \exp\left[q(\mathbf{z}, \mathbf{p})\right] d\mathbf{p}} \\ &= \frac{\begin{bmatrix} \iint_{\mathcal{R}(\mathbf{z})} x \, \exp\left[q(\mathbf{z}, \mathbf{p}) - C\right] d\mathbf{p} \\ \iint_{\mathcal{R}(\mathbf{z})} y \, \exp\left[q(\mathbf{z}, \mathbf{p}) - C\right] d\mathbf{p} \end{bmatrix}}{\iint_{\mathcal{R}(\mathbf{z})} \exp\left[q(\mathbf{z}, \mathbf{p}) - C\right] d\mathbf{p}} \end{aligned} \tag{9}$$

Indeed the evaluation of the terms $\exp\left[q(\mathbf{z}, \mathbf{p}) - C\right]$ of (9) requires a much smaller number of bits in FLOPs with respect to the evaluation of $\exp\left[q(\mathbf{z}, \mathbf{p})\right]$, when C assumes large values.

[1] Reusing the quantities $\left\{h\left(\mathbf{z}, \boldsymbol{\alpha}^{(k,j)}\right)\right\}$ in the 3 integrals appearing in (8) reduces the overall computational complexity of MMSE estimation, since the evaluation of $h(\cdot, \cdot)$ involves complicated computations (see (3)).

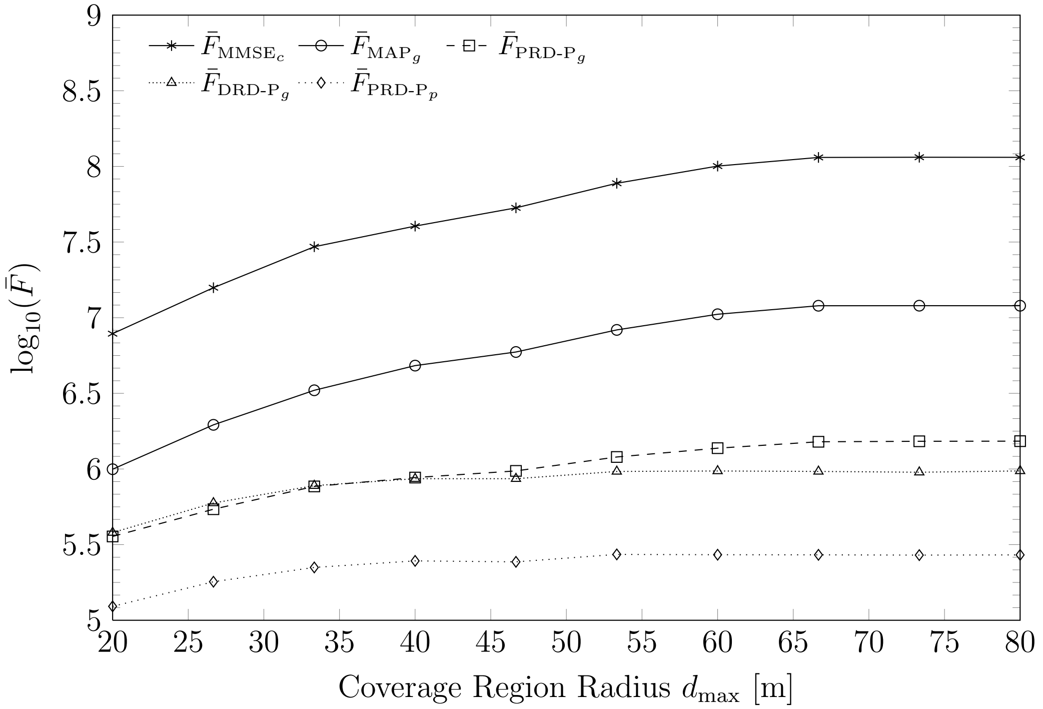

Figure 2: FLOP count $\bar{F}$ for various estimation algorithms versus $d_{\max}$. Map #4 is considered.

4 Additional Numerical Results

In this Section some numerical results related to the algorithms described in [1] are reported (they could not be included in [1] because of space limitations).

4.1 Impact of the Coverage Region Radius on Complexity

In the localization system described in [1] the radius $d_{\max}$ of the anchor coverage region (see [1, Sec. 2]) plays a fundamental role in determining both system costs[2] and the accuracy and complexity of most estimation algorithms; this is due to the fact that $d_{\max}$ determines the size of the domains $\mathcal{R}^{(\mathcal{Z})}$ and $\mathcal{P}^{(\mathcal{Z})}$. Fig. 2 allows us to assess the influence of $d_{\max}$ on the FLOP count for some

[2] Long-range anchors allow to perform accurate localization even if their spatial density is low and are thus useful to reduce system deployment costs.

implementations (in this case $\tilde{d}_{\max} = 0.9 d_{\max}$ is selected[3] when generating $\mathbf{z}$). From these results it can be inferred that:

1. $\bar{F}_{\text{MMSE}_c}$ and $\bar{F}_{\text{MAP}_g}$ increase quickly with $d_{\max}$ (e.g., $\bar{F}_{\text{MMSE}_c} \simeq 10^7 \to 10^8$), since the region $\mathcal{R}^{(\mathcal{Z})}$ becomes larger.

2. $\bar{F}_{\text{DRD-P}_g}$, $\bar{F}_{\text{PRD-P}_g}$ and $\bar{F}_{\text{PRD-P}_p}$ increase much more slowly than $\bar{F}_{\text{MMSE}_c}$ and $\bar{F}_{\text{MAP}_g}$ with $d_{\max}$ (e.g., $\frac{\partial \bar{F}_{\text{MMSE}_c}}{\partial d_{\max}} \simeq 7.3 \cdot 10^2 \frac{\partial \bar{F}_{\text{PRD-P}_p}}{\partial d_{\max}}$).

These results show that the computational saving provided by DRD and PRD estimators over MMSE and MAP estimators becomes very relevant when $d_{\max}$ is large (i.e., when long-range anchors are employed). On the contrary, when $d_{\max}$ is small (e.g., in a scenario of "connectivity-based" localization), $\mathcal{R}^{(\mathcal{Z})}$ is small and MMSE and MAP estimation algorithms can be directly employed (regardless of the size of $\mathcal{R}$).

4.2 Optimal γ Parameter for DRD-MMSE and DRD-MAP

In any localization system, a trade-off between complexity and accuracy has to be achieved. This is clearly evidenced by Fig. 3, where the γ parameter of DRD estimators is modified, so changing the portion of $\mathcal{R}^{(\mathcal{Z})}$ to be selected (see Paragraph [1, Paragraph 3.3]); this strongly influences the accuracy and complexity of DRD estimators. In particular, Fig. 3 shows that, as γ increases, DRD-E$_c$ and DRD-P$_g$ become more accurate, but also require a larger computational effort. Moreover, the RMSE characterizing such algorithms exhibits a threshold behaviour. In this case a good accuracy-complexity trade-off is obtained selecting $\gamma \simeq 0.6$, since this corresponds to the knee of the RMSE curve.

References

[1] F. Montorsi, F. Pancaldi, and G. M. Vitetta, "Reduced-Complexity Algorithms for Indoor Map-Aware Localization Systems," *Submitted to IEEE Trans. on Sig. Process.*, 2012.

[3] When the likelihoods [1, Eq. (6)] or [1, Eq. (7)] are employed, $d_{\max}$ should be larger than the maximum free-space anchor range, i.e. $\tilde{d}_{\max}$; more details can be found in [14].

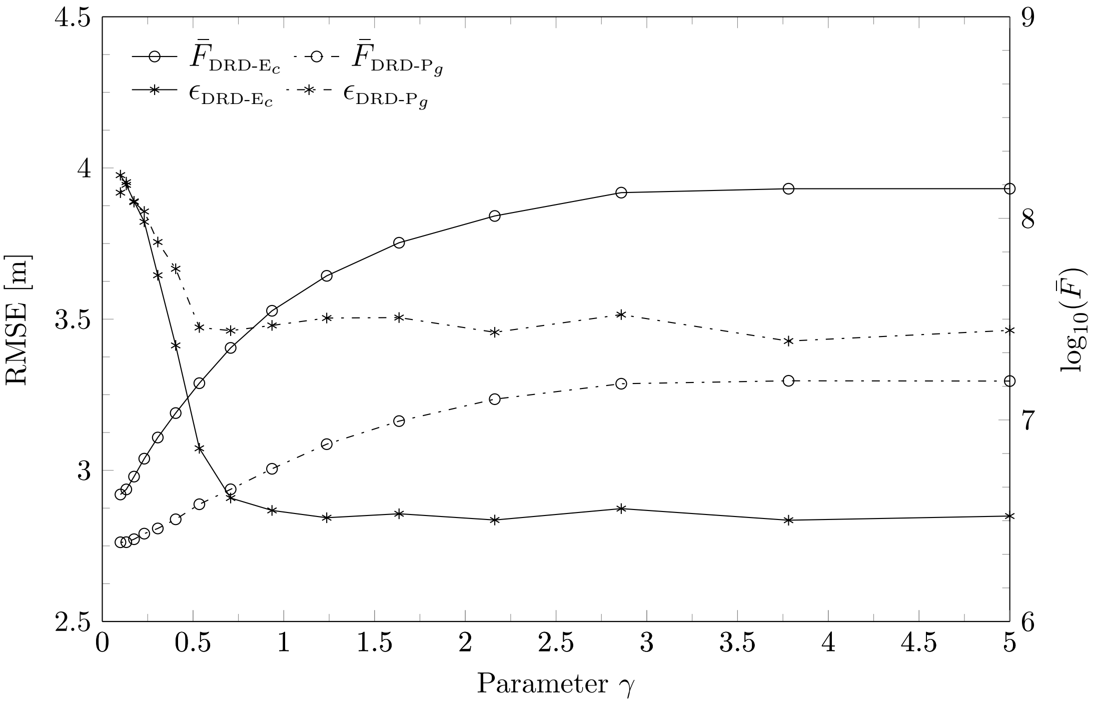

Figure 3: Accuracy and complexity characterizing DRD-MMSE and DRD-MAP estimators versus γ. Map #4 is considered.

[2] F. Cheng and I.-M. Lin, "Covering of polygons by rectangles," *Computer Aided Design*, vol. 21, no. 2, pp. 97–101, 1989.

[3] A. Hegedus, "Algorithms for covering polygons by rectangles," *Computer Aided Design*, vol. 14, no. 5, pp. 257–260, 1982.

[4] S. Chaiken, D. J. Kleitman, M. Saks, and J. Shearer, "Covering regions by rectangles," *SIAM J. Alg. Disc. Meth.*, vol. 2, no. 4, pp. 394–410, 1981.

[5] W. Liou, J.-M. Tan, and R. Lee, "Minimum rectangular partition problem for simple rectilinear polygons," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 9, no. 7, pp. 720–733, Jul. 1990.

[6] S.-Y. Wu and S. Sahni, "Fast Algorithms to Partition Simple Rectilinear Polygons," *VLSI Design*, vol. 1, no. 3, pp. 193–215, 1994.

[7] R. Cools, "Advances in multidimensional integration," *J. Computational and Appl. Math.*, vol. 149, no. 1, pp. 1–12, Dec. 2002.

[8] ——, "An Encyclopaedia of Cubature Formulas," *J. Complexity*, no. 19, pp. 445–453, 2003.

[9] M. Festa and A. Sommariva, "Computing almost minimal formulas on the square," *Submitted to J. Comput. Appl. Math.*, 2011.

[10] A. Haegemans and R. Piessens, "Construction of Cubature Formulas of Degree Seven and Nine Symmetric Planar Regions, Using Orthogonal Polynomials," *SIAM J. Numerical Anal.*, vol. 14, no. 3, pp. 492–508, 1977.

[11] H. Li, J. Sun, and Y. Xu, "Cubature Formula and Interpolation on the Cubic Domain," *Numer. Math. Theor. Meth. Appl.*, vol. 2, no. 2, pp. 119–152, 2009.

[12] I. Arasaratnam and S. Haykin, "Cubature Kalman Filters," *IEEE Trans. Autom. Control*, vol. 54, no. 6, pp. 1254–1269, Jun. 2009.

[13] I. Arasaratnam, S. Haykin, and T. R. Hurd, "Cubature Kalman Filtering for Continuous-Discrete Systems: Theory and Simulations," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 4977–4993, Oct. 2010.

[14] F. Montorsi, F. Pancaldi, and G. M. Vitetta, “Map-Aware Models for Indoor Wireless Localization Systems: An Experimental Study,” *Submitted to IEEE Trans. on Sig. Process.*, 2012.