Saarland University, Saarbrücken, Germany

Klee's Measure Problem on Fat Boxes in Time $\mathcal{O}(n^{(d+2)/3})$

Karl Bringmann

s9kabrin@stud.uni-saarland.de

Abstract

The measure problem of Klee asks for the volume of the union of n axis-parallel boxes in a fixed dimension d. We give an $\mathcal{O}(n^{(d+2)/3})$ time algorithm for the special case of all boxes being cubes or, more generally, fat boxes. Previously, the fastest run-time was $n^{d/2}\,2^{\mathcal{O}(\log^* n)}$, achieved by the general case algorithm of Chan [SoCG 2008]. For the general problem our run-time would imply a complexity theoretic breakthrough for the k-clique problem and can thus be considered extremely unlikely.

1 Introduction

We consider a natural geometric problem: Computing the volume of the union of n axis-parallel boxes in $\mathbb{R}^d$, where d is considered to be constant. This problem was first stated by Klee [10], who gave an $\mathcal{O}(n \log n)$ algorithm for the case of $d = 1$ and asked if this was optimal, while leaving the question open how to extend this to arbitrary dimension $d > 1$. Bentley [2] gave an algorithm with run-time $\mathcal{O}(n^{d-1} \log n)$ for $d \geq 2$, which is optimal for $d = 2$. As Chan [6] noted, this is better than explicitly constructing the union of the given boxes, which has a worst-case combinatorial complexity of $\Theta(n^d)$. The next breakthrough was a paper by Overmars and Yap [12] giving an algorithm with run-time $\mathcal{O}(n^{d/2} \log n)$ for $d \geq 2$, which was the fastest known until Chan [6] slightly improved their approach to get a run-time of $n^{d/2}\, 2^{\mathcal{O}(\log^* n)}$. Interestingly, there is also an $\mathcal{O}\big(\frac{dn}{\varepsilon^2}\big)$ Monte Carlo $(1+\varepsilon)$-approximation algorithm for Klee's measure problem (KMP) [4].

So far, the only known lower bound is $\Omega(n \log n)$ for any d [8]. Additionally, there are a few known hardness results for KMP: Suzuki and Ibaraki [13] showed that it is #P-hard for $d \to \infty$, i.e., if we do not consider d to be constant. Furthermore, Chan [6] showed W[1]-hardness of KMP, meaning that no algorithm of run-time $f(d)\, n^{\mathcal{O}(1)}$ is likely to exist for any function f depending only on d. His proof relies on a reduction of KMP to the k-clique problem, i.e., checking whether a graph with n vertices contains a clique of size k. Since the best-known algorithm for the latter problem has a run-time of $\mathcal{O}(n^{\omega\lfloor k/3\rfloor+(k \bmod 3)})$ [11], where $\omega = 2.376..$ is the exponent occuring in fast matrix multiplication, he concludes that any $\mathcal{O}(n^{0.396d})$ algorithm for KMP, and even a combinatorial $\mathcal{O}(n^{(\frac{1}{2}-\varepsilon)d})$ algorithm, would require a breakthrough in the k-clique problem.

This raises the question whether there are easier special cases of KMP. To develop algorithms for such special cases is also one of the open problems posed by Jeff Erickson [7] on his website. The two special cases that got attention so far, are the case of all boxes being cubes (or fat boxes), or unit cubes (fat boxes of roughly equal size), respectively:

Klee's measure problem on cubes (C-KMP) can be solved in time $\mathcal{O}(n^{4/3} \log n)$ for $d = 3$, using an algorithm of Agarwal, Kaplan, and Sharir [1]. For higher dimensions, Chan's KMP algorithm is the fastest known. One may generalize the case of cubes to the one of α-fat boxes for some constant $\alpha \geq 1$, where we say that a box is α-fat, if its maximal side length is at most a factor of α larger than its minimal side length. A simple reduction shows that both problems can be solved in the same asymptotic run-time: Represent a fat box by a union of possibly overlapping cubes. You need at most $\lceil\alpha\rceil^{d-1}$ cubes for such a representation. Then the volume of the union of the cubes is the same as the volume of the union of the fat boxes, and we increased the number of boxes only by a constant factor. Hence, all algorithms for C-KMP also solve the case of fat boxes.

We speak of Klee's measure problem on unit cubes (UC-KMP) if all cubes have the same side length. As the union of n unit cubes has combinatorial complexity $\Theta(n^{\lfloor d/2\rfloor})$ [3], this special case can be solved in $\mathcal{O}(n^{\lfloor d/2\rfloor}\text{polylog } n)$ [9], which improves upon general KMP algorithms in odd dimensions. Furthermore, an elaborate algorithm runs in $\mathcal{O}(n^{\lceil d/2\rceil-1+\frac{1}{\lceil d/2\rceil}}\text{polylog } n)$ [5], which improves in even dimensions. This case can be generalized to fat boxes of roughly equal size, analogously to cubes and fat boxes. Note that all of these algorithms have run-time $\Omega(n^{d/2-1})$ in the worst case.

The Result: In this paper we are improving upon the algorithms for both of these special cases (in high dimensions) by giving a better algorithm for C-KMP. As customary, we assume d to be constant.

Theorem 1. *Given a set M of n axis-parallel cubes in $\mathbb{R}^d$, $d \geq 2$, we can compute the volume of the union of these cubes, i.e, solve C-KMP, in time $\mathcal{O}(n^{(d+2)/3})$.*

Note that this is faster than any known algorithm for C-KMP in dimensions $d \geq 4$ (at least by a factor of $2^{\mathcal{O}(\log^*(n))}$) and faster than any known algorithm for UC-KMP in dimensions $d > 8$.

More importantly, we reduce the exponent from $\frac{d}{2}+\mathcal{O}(1)$ to $\frac{d}{3}+\mathcal{O}(1)$. As this is notably faster than the bound $n^{0.396d}$, finding such a good algorithm for the general case would imply a breakthrough in more classical areas (k-clique).

The space requirement of our algorithm, as we will present it in this paper, is also $\mathcal{O}(n^{(d+2)/3})$ and cannot be bounded by an asymptotically smaller function. However, one may use the very same trick as Overmars and Yap [12] who show in their Section 5 how to reduce the amount of storage to $\mathcal{O}(n)$.

Our result uses some ideas from Overmars and Yap [12] as well as from Agarwal et al. [1]. However, it is no generalization of the latter, which works only for $d = 3$. Instead, we switch our view to another base case. The contribution of this paper can be seen as the identification of the this simpler base case, and the idea how to solve it. The differences to existing algorithms will be highlighted again in Section 2.

In the following section we give a structural overview of our algorithm for C-KMP. Sections 3 to 5 give the details of the algorithm in 3 steps. In the last part we give concluding remarks.

2 Structural Overview

In this section we give a rough overview of the algorithm. We start by defining some notation and the problem: In dimension $d \in \mathbb{N}$ we define a *box* to be a set $[x_1, y_1] \times \ldots \times [x_d, y_d] \subseteq \mathbb{R}^d$, $x_i \leq y_i$ for all i. A *cube* is a box with $y_i - x_i = y_j - x_j$ for all dimensions i, j, i.e., for $1 \leq i, j \leq d$. We refer to x_i (y_i) as the lower (upper) i-th coordinate of the box or cube. For a set $U \subseteq \mathbb{R}^d$ we define $\mathrm{Vol}(U)$ as the Lebesgue measure of U. If we want to point out the dimension we are working in, we use Vol^d instead of Vol. Moreover, for a box C we define $\mathrm{Vol}_C(U) := \mathrm{Vol}\big(C \cap U\big)$, the volume of U inside C. For a set of boxes M, we define $\mathcal{U}(M) := \bigcup_{B\in M} B$. Then $\mathrm{Vol}(\mathcal{U}(M))$ is the volume of the union of the boxes in M.

The problem C-KMP now is to compute $\mathrm{Vol}(\mathcal{U}(M))$ for a given set M of n cubes. Using a bounding box $\mathcal{BB}$ of all the boxes in M, this quantity is the same as $\mathrm{Vol}_{\mathcal{BB}}(\mathcal{U}(M))$.

Representation of sets of boxes: Similar to a construction of Overmars and Yap [12], we will represent a set of boxes M in the following way: We have sorted lists L_i of reals, $1 \leq i \leq d$, containing the i-th (upper and lower) coordinates of the boxes in M. Thus, every box occurs $2d$ times in the d lists. Furthermore, each occurence of a box will have a pointer to the next occurence, such that all occurrences of a box are in a cyclic structure of $2d$ pointers.

Such a representation allows us to do several things in a fast way: For example, given a pointer to one occurence of a box, we can delete this box from the set M in time $\mathcal{O}(1)$, as we can iterate through all occurences of the box in constant time. Also, we may iterate through all boxes sorted by lower i-th coordinate or sorted by upper i-th coordinate or similarly. More importantly, we will at some points in this paper be in the position that we iterate through some L_i and define a subset M' of M by deciding during the iteration whether the current box belongs to M' or not. Then a representation of M' can be computed in time $\mathcal{O}(|M|)$ by marking elements of M' in all lists L_i and computing L'_i by iterating through L_i and taking marked elements.

This representation will save us a logarithmic factor in run-time. Computing this representation when given any standard representation of sets of boxes can be done in time $\mathcal{O}(n \log n)$. This does not increase the overall run-time for $d \geq 2$.

2.1 Space partitioning

We compute the volume $\mathrm{Vol}_{\mathcal{BB}}(\mathcal{U}(M))$ by splitting up the bounding box: We define a *partitioning* of a box B to be a finite set P of boxes with $\bigcup_{R\in P} R = B$ and that for two different boxes $R, R' \in P$ the intersection is a null set, i.e., $\mathrm{Vol}(R \cap R') = 0$. We refer to the boxes in a partitioning as *regions*.

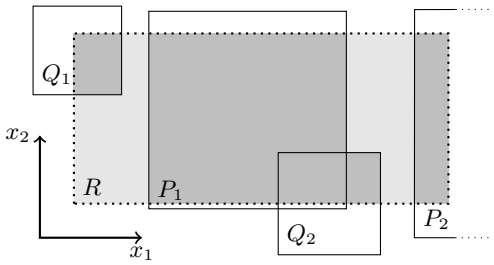


Figure 1: An example of piles and quasi-piles of a region R. P_1 and P_2 are 1-piles, Q_1 and Q_2 are $(1,2)$-quasi-piles. Note that in 2D all cubes partially covering R are piles or quasi-piles. This does not hold in higher dimensions.

Now, if we have a partitioning P of the bounding box $\mathcal{BB}$, we can compute the quantity of interest as $\mathrm{Vol}(\mathcal{U}(M)) = \mathrm{Vol}_{\mathcal{BB}}(\mathcal{U}(M)) = \sum_{R\in P} \mathrm{Vol}_R(\mathcal{U}(M))$.

We want such a partitioning to have several nice properties and we need some definitions to state them: We say that a box B *partially covers* a region R, if $\mathrm{Vol}(R) > \mathrm{Vol}_R(B) > 0$. A region is said to be *fully covered* by a box B, if $R \subseteq B$. Furthermore, for a box $B = [x_1, y_1] \times \ldots \times [x_d, y_d]$ we call $B_i := [x_i, y_i]$ the *i-interval* and $|B_i| = y_i - x_i$ the *i-th side length* of B. We call a box B an *i-pile* for region R and a dimension i, if we have $R_k \subseteq B_k$ for all dimensions $k \neq i$, and $\emptyset \subsetneq R_k \cap B_k \subsetneq R_k$ for $k = i$. We call B an *(i,j)-quasi-pile*, $i \neq j$, if we have $R_k \subseteq B_k$ for all dimensions $k \neq i, k \neq j$, and $\emptyset \subsetneq R_k \cap B_k \subsetneq R_k$ for $k = i$ and $k = j$. We call it a pile, if it is an i-pile for some i, similarly for quasi-pile. Note that any (quasi-)pile of a region R partially covers R. Intuitively, a box is a pile (quasi-pile), if it covers the region R in all but one (two) dimensions, see Figure 1 for an example.

Now, we can state the two properties we want from a partitioning P of the bounding box:

(P1) Every region $R \in P$ is partially covered by $\mathcal{O}(n^{2/3})$ cubes in M, of which $\mathcal{O}(n^{1/3})$ are quasi-piles for R and all other are piles for R.

(P2) The number of regions in P is $\mathcal{O}(n^{d/3})$.

Note that property (P1) does not state anything about the number of fully covering boxes of R, as we explicitly defined piles and quasi-piles not to be fully covering.

In Section 3 we will give a construction for such a partitioning. It returns the set of regions together with their sets of partially covering cubes and a fully covering box, if there exists one, in time $\mathcal{O}(n^{(d+2)/3})$ for $d \geq 2$. As it does not matter whether we have one or all fully covering boxes for computing $\mathrm{Vol}_R(\mathcal{U}(M))$, we can assume that each region has at most one fully covering box in the remainder.

The partitioning we build is very similar to the one of Overmars and Yap [12], just with a different parameter, and the same as a generalized Agarwal et al. [1] would do. The only difference is that we solve a static problem, while the mentioned algorithms do a sweep first and solve a dynamic problem. This is why they need to handle a partition *tree*, where one can insert and delete boxes, while we only need a partitioning, without any tree build on it. The reason for us solving the static problem is that the dynamic problem seems too hard to be solved fast enough to get a better algorithm, i.e., I do not know how to solve it.

2.2 Solving the base case

We are left with the following *base case*: Computing $\mathrm{Vol}_R(\mathcal{U}(M))$ for a region R and a set M of cubes containing at most one fully covering box for R, $\mathcal{O}(n^{1/3})$ quasi-piles, and $\mathcal{O}(n^{2/3})$ piles. If M

contains a fully covering box, computing this volume is trivial. Otherwise, we will split up this base case into a few easier problems. For this, we refer to the points $(z_1, \ldots, z_d)$ with $z_i \in \{x_i, y_i\}$ for all dimensions i as the *vertices* of a region $R = [x_1, y_1] \times \ldots \times [x_d, y_d]$. We call a (quasi-)pile of R *vertex-containing*, if it contains a vertex of R.

The motivation of splitting up the base case is that we found a way to solve the following *reduced base case* efficiently: We are given a region R in $\mathbb{R}^{d'}$ and a set M' of m boxes partially covering R, which are all vertex-containing piles or vertex-containing quasi-piles. Compute $\mathrm{Vol}_R^{d'}(\mathcal{U}(M'))$.

First of all, we found a way to solve the base case (on $\mathcal{O}(n^{1/3})$ quasi-piles and $\mathcal{O}(n^{2/3})$ piles) by solving $\mathcal{O}(n^{1/3})$ reduced base cases (on $\mathcal{O}(n^{1/3})$ boxes) using additional time $\mathcal{O}(n^{2/3})$. This can be seen as a Turing reduction from the base case to the reduced base case. This reduction is where we use the fact that we are facing cubes, not general boxes. In Section 4 this step will be described in detail.

Secondly, we found a way to solve the reduced base case in time $\mathcal{O}(m)$. Here, we use the standard technique of a space sweep to get a dynamic problem with dynamic piles and static quasi-piles. Then we solve this dynamic problem by considering a few smaller dynamic problems. The elaborate details of this step can be found in Section 5.

Note that this leaves us with an $\mathcal{O}(n^{(d+2)/3})$ algorithm for C-KMP.

For differentiating between our approach and [12] or [1], one has to point out that our base case is a static problem, while the mentioned algorithms solve a dynamic one. Furthermore, the identification and solution of our *reduced base case* is completely novel.

2.3 Subsumption

Summing up, we see that the algorithm consists of 3 steps:

- We construct a partitioning P of a bounding box $\mathcal{BB}$ with properties (P1) and (P2). Having this, we compute the volume $\mathrm{Vol}_R(M)$ inside every region $R \in P$ and sum up.
- We transform the base case of computing $\mathrm{Vol}_R(M)$ into easier reduced base cases.
- We solve the reduced base case.

These three steps will be described in the following 3 sections.

3 Construction of the space partitioning

In this section we describe how to compute a partitioning of the bounding box of M with properties (P1) and (P2) in time $\mathcal{O}(n^{(d+2)/3})$. Our construction will, additionally, compute for each region R the set of partially covering cubes of R in M and a fully covering cube, if there exists one.

The Construction: Say the bounding box of M is $\mathcal{BB} = [x_1, y_1] \times \ldots \times [x_d, y_d]$. We will proceed in d levels, starting with $\mathcal{BB}$ on level 1: On a level $1 \leq \ell \leq d$ we are given a box $B = [x'_1, y'_1] \times \ldots \times [x'_{\ell-1}, y'_{\ell-1}] \times [x_\ell, y_\ell] \times \ldots \times [x_d, y_d]$ and the set $M_{B,\ell} \subseteq M$ of cubes partially covering B. For a cube C in $M_{B,\ell}$ let v be the number of dimensions k, $1 \leq k < \ell$ such that C has a k-th coordinate in the open interval (x'_k, y'_k). We split $M_{B,\ell}$ into three subsets: $M^i_{B,\ell}$ contains all cubes with $v = i$ for $i \in \{0, 1, 2\}$. We will show in a moment, that no cube has $v > 2$. Now, we split the interval $[x_\ell, y_\ell]$ into $\mathcal{O}(n^{1/3})$ intervals, each of whose interior contains at most $n^{2/3}$ ℓ-th coordinates of cubes in $M^0_{B,\ell}$, at most $n^{1/3}$ ℓ-th coordinates of cubes in $M^1_{B,\ell}$ and no ℓ-th coordinate of a cube in $M^2_{B,\ell}$. This splits the box B into $\mathcal{O}(n^{1/3})$ smaller boxes (children), on which we recurse to level $\ell + 1$ after computing their sets of partially covering cubes $M_{B',\ell+1}$.

All the boxes we end up with on level $d + 1$ will be the regions of our partitioning. For each such region R, we are given $M_{R,d+1}$, the set of partially covering cubes of R in M, and split it into three subsets, $M^i_{R,d+1}$, $i \in \{0, 1, 2\}$, just as we did on lower levels.

Correctness: We need to prove two facts to show that this construction works: The first is that no cube will ever have $v > 2$ (on any level $1 \leq \ell \leq d+1$). Any such cube would have $v = 2$ on some lower level $\ell' < \ell$, so it would be in $M^2_{B,\ell'}$ on that level. But then we split so that no coordinate of the cube is in the interior of a new interval, so its v-value cannot grow.

The second fact is that we need at most $\mathcal{O}(n^{1/3})$ splits to assure that each interior of a new interval contains at most $n^{2/3}$ ℓ-th coordinates of cubes in $M^0_{B,\ell}$, at most $n^{1/3}$ ℓ-th coordinates of cubes in $M^1_{B,\ell}$ and no ℓ-th coordinate of a cube in $M^2_{B,\ell}$. To prove this, it suffices to show the upper bounds $|M^1_{B,\ell}| = \mathcal{O}(n^{2/3})$ and $|M^2_{B,\ell}| = \mathcal{O}(n^{1/3})$ for $1 \leq \ell \leq d+1$, as we can assure that each interval includes $|M^i_{B,\ell}|/\mathcal{O}(n^{1/3})$ coordinates of cubes in $M^i_{B,\ell}$ for each i by splitting at $\mathcal{O}(n^{1/3})$ positions. These inequalities are trivially true on the first level, as those sets are empty there. Also, they are maintained during the split at a level ℓ: For each child B' of the box B (corresponding to an interval $[x'_\ell, y'_\ell]$) we have $|M^1_{B',\ell+1}| \leq |M^1_{B,\ell}| + n^{2/3}$, as new such cubes may only come from a box in $M^0_{B,\ell}$, whose ℓ-th coordinate is contained in (x'_ℓ, y'_ℓ), and this number is bounded from above by $n^{2/3}$ by our construction. As the number of levels is d, a constant, the upper bound holds in all levels. We get the inequality for $|M^2_{B,\ell}|$ similarly.

It is now very easy to show the following lemma:

Lemma 2. *The above construction yields a partitioning with properties (P1) and (P2).*

Proof. Observe that the set of quasi-piles of R is exactly $M^2_{R,d+1}$ and the set of piles of R is $M^1_{R,d+1}$. Moreover, the way we constructed $M^0_{R,d+1}$ no cube in it can *partially* cover R, since every cube in this set has no k-th coordinate contained in the interior of the interval R_k for any k, so this set is empty. Since we proved $|M^2_{R,d+1}| = \mathcal{O}(n^{1/3})$ and $|M^1_{R,d+1}| = \mathcal{O}(n^{2/3})$, this shows that each region is partially covered by $\mathcal{O}(n^{2/3})$ piles, $\mathcal{O}(n^{1/3})$ quasi-piles, and no other boxes in M, which is property (P1).

For (P2) note that we split the box into $\mathcal{O}(n^{1/3})$ smaller boxes at each of the d levels. Hence, we create $\mathcal{O}(n^{d/3})$ regions. □

Implementation Details: Now we have a construction of a partitioning with the properties we wanted. However, it is not so clear how to implement this with the aforementioned run-time of $\mathcal{O}(n^{(d+2)/3})$, for the intuitive upper bound being $\mathcal{O}(n^{(d+3)/3} \log n)$. We use here, that we represent sets of cubes by sorted lists of coordinates in every dimension. Having these lists, we can split the set $M_{B,\ell}$ into $M^i_{B,\ell}$ and find the split of the current interval in time $\mathcal{O}(n)$ (this would not be possible without a sorted order). After we split the box into $\mathcal{O}(n^{1/3})$ children, we compute the set of partially covering boxes for each child and build sorted lists of coordinates of these boxes from the sorted lists we got for $M_{B,\ell}$. This can be done in $\mathcal{O}(n)$ per child. For each level $1 \leq \ell < d$, this run-time is admissible, as we get a run-time of $\mathcal{O}(n \cdot n^{(d-1)/3}) = \mathcal{O}(n^{(d+2)/3})$.

On the last level $\ell = d$, finding the positions for splits can be done in $\mathcal{O}(n)$, too, which is admissible, as there are $\mathcal{O}(n^{(d-1)/3})$ problem instances on level d. However, computing the sets of partially covering cubes for the children has to be done in (amortized) time $\mathcal{O}(n^{2/3})$ per child. This is easy for cubes that become i-piles or (i,j)-quasi-piles with $i, j < d$ in the children: Any such (quasi-)pile is partially covering one child if and only if it partially covers all children. Thus, a list of these (quasi-)piles can be constructed in time $\mathcal{O}(n)$, i.e., $\mathcal{O}(n^{2/3})$ per child. For cubes that become d-piles or (i,d)-quasi-piles, $i < d$, we use that any such (quasi-)pile is partially covering at most 2 children, since each of its two d-th coordinates may lie in the interior of another child. If we go through the children and the sorted list of d-th coordinates of these cubes "in parallel", we can also find the children a cube partially covers in (amortized) constant time. Hence, also these (quasi-)piles can be computed in $\mathcal{O}(n^{2/3})$ per child. Since any partially covering cube of a child is a pile or quasi-pile by property (P1), we are done with these two cases. After this, we have to split the set $M_{R,d+1}$ into the sets $M^i_{R,d+1}$. Note that this can be done in $\mathcal{O}(n^{2/3})$ trivially, as we

proved above that $|M^0_{R,d+1}| = 0, |M^1_{R,d+1}| = \mathcal{O}(n^{2/3})$ and $|M^2_{R,d+1}| = \mathcal{O}(n^{1/3})$, so that the overall number of cubes in $M_{R,d+1}$ is small enough to allow linear time procedures.

So far we lack the computation of a fully covering box for each fully covered region. We will compute those boxes on level d, too. For this, we go through the sorted list of d-th coordinates of *all* cubes in M. At the same time, we go through the children in the same sorted order. For a cube we can tell in constant time, whether it fully covers the current child. If so, we save it as a fully covering box for this child and go on in the list of children (and check the current cube with the next child). If not, we go on in the list of cubes or of children, respectively. This way, we compute a fully covering box for any child in time linear in the number of cubes in M and the number of children, i.e., in $\mathcal{O}(n)$, which is $\mathcal{O}(n^{2/3})$ per child.

Thus, we have seen a construction of a partitioning with properties (P1) and (P2) that computes for each region its set of partially covering boxes and one fully covering box, if there is one.

4 Simplifying the base case

In this section we show how to simplify the base case to the reduced base case. Recall that the base case is the problem of computing $\mathrm{Vol}^d_R(\mathcal{U}(M))$ for a region R and a set of cubes M containing $\mathcal{O}(n^{1/3})$ quasi-piles and $\mathcal{O}(n^{2/3})$ piles (and no other boxes). The reduced base case is the problem of computing $\mathrm{Vol}^{d'}_R(\mathcal{U}(M'))$ of a set of boxes M' consisting of vertex-containing piles and vertex-containing quasi-piles (in some dimension $1 \leq d' \leq d$). We describe how to solve the base case by solving $\mathcal{O}(n^{1/3})$ reduced base cases on $\mathcal{O}(n^{1/3})$ boxes using additional run-time of $\mathcal{O}(n^{2/3})$. This is the point where we use that we are dealing with cubes rather than general boxes.

Dealing with most piles: Without loss of generality we may assume that the side lengths of region R are in sorted order $|R_1| \leq |R_2| \leq \ldots \leq |R_d|$. An i-pile B of R is vertex-containing iff the interval B_i contains an endpoint of the interval R_i. Thus, any i-pile B of R with $i < d$ is already vertex-containing, since otherwise we would have $B_i \subsetneq R_i$, thus $|B_i| < |R_i| \leq |R_d|$, so that we cannot have $R_d \subseteq B_d$, as B is a cube, but this contradicts B being a pile.

Dealing with these vertex-containing i-piles, $i < d$, in the computation of $\mathrm{Vol}_R(\mathcal{U}(M))$ is easy: Out of the i-piles B with B_i containing the lower i-th coordinate of R we take the pile P_1 with largest interval $B_i \cap R_i$. Similarly, out of the i-piles B with B_i containing the upper i-th coordinate of R we take the pile P_2 with the largest interval $B_i \cap R_i$. Then we have for any i-pile B: $B \cap R \subseteq (P_1 \cup P_2) \cap R$. Thus, we can remove all i-piles other than P_1, P_2 from M and still get the same volume $\mathrm{Vol}_R(\mathcal{U}(M))$. We reduced the number of i-piles, $i < d$, to a constant this way.

Sweep along dimension d: Now we sweep along dimension d. Let $R_d = [y_1, y_2]$ and let $z_1 \leq \ldots \leq z_m$ be the d-th coordinates of boxes in M that lie in R_d. Let $z_0 := y_1, z_{m+1} := y_2$. We sweep a horizontal plane along dimension d from y_1 to y_2 stopping at each z_i, $0 \leq i \leq m+1$. Let $\Pi(t)$ be the plane satisfying $x_d = t$. For each $0 \leq i \leq m$ the cross-section $\Pi(z) \cap R \cap \mathcal{U}(M)$ is the same for all $z \in (z_i, z_{i+1})$. Let v_i denote the $(d-1)$-dimensional volume of this cross-section. Then $\mathrm{Vol}_R(\mathcal{U}(M)) = \sum_{i=0}^{m} v_i(z_{i+1} - z_i)$. In every interval (z_i, z_{i+1}) where a d-pile is active, i.e., where there exists a d-pile B of R with $(z_i, z_{i+1}) \subseteq B_d$, the volume v_i is trivially equal to $\prod_{j \neq d} |R_j|$. Not looking at these trivial v_i, the sequence of v_i's changes only if a box in M, which is no d-pile, is inserted or deleted. The number of those boxes is bounded by $\mathcal{O}(n^{1/3})$, as we reduced the number of i-piles, $i < d$, to a constant. Thus, we have to compute $\mathcal{O}(n^{1/3})$ v_i's, each with $\mathcal{O}(n^{1/3})$ partially covering boxes and no fully covering ones. We spent time $\mathcal{O}(n^{2/3})$ so far, since all this can be implemented in linear time with the coordinates already sorted. Note that we are not trying to compute v_i dynamically by updating v_{i-1}, but consider them as static problems.

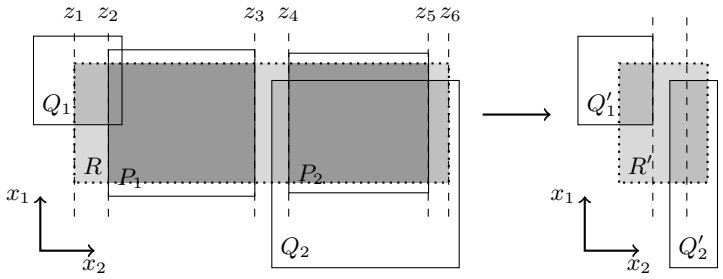


Figure 2: We want to cut out the 2-piles P_1 and P_2. This is done by cutting out the intervals (z_2, z_3) and (z_4, z_5). On the right hand side you see the result, where dashed lines denote that we deleted an interval there.

Most boxes are vertex-containing: We are left with $\mathcal{O}(n^{1/3})$ $(d-1)$-dimensional base cases, but not in its full generality. In fact, every (quasi-)pile is vertex-containing in these instances, except the $(d-1)$-piles. We use the fact that we are facing cubes to prove this: As in the second paragraph of this section we show that every i-pile, $i < d-1$, is already vertex-containing. This holds in general for the base case. A non-vertex-containing (i,j)-quasi-pile B, $i < j$, has side length $|B_i| < |R_i| \leq |R_j| \leq |R_d|$. But then B cannot stem from a pile or quasi-pile of the original region R, as B_k does not include R_k for $k = i, j, d$.

Cutting out $(d-1)$-piles: What do we do with the remaining $(d-1)$-piles that are non-vertex-containing? We simply cut them out of dimension $d-1$: Let $R_{d-1} = [y_1, y_2]$ and let $y_1 = z_0 \leq \ldots \leq z_{2m} = y_2$ be a partitioning of R_{d-1} such that each $R_1 \times \ldots \times R_{d-2} \times (z_i, z_{i+1})$ is fully covered by some $(d-1)$-pile for even i but neither fully nor partially covered by a $(d-1)$-pile for odd i, and $m = \mathcal{O}(n^{1/3})$. In this partitioning we want to cut out the intervals (z_{2i}, z_{2i+1}), as depicted in Figure 2. Formally, one can write this using a compression function κ defined as follows: For an $x \in (y_1, y_2]$ we find the i with $x \in (z_i, z_{i+1}]$. If i is odd, i.e., we do not cut out (z_i, z_{i+1}), we define $\kappa(x) := x - \sum_{j=0}^{(i-1)/2}(z_{2j+1} - z_{2j})$, if i is even we define $\kappa(x) := z_i - \sum_{j=0}^{(i-2)/2}(z_{2j+1} - z_{2j})$. Moreover, we set $\kappa(x) := x$ for $x \leq y_1$, and $\kappa(x) := x - Z$ for $x > y_2$, where $Z := \sum_{j=0}^{m-1}(z_{2j+1} - z_{2j})$ is the total length of cut intervals. For a box $B = [x_1, x'_1] \times \ldots \times [x_{d-1}, x'_{d-1}]$ we define $\kappa(B) := [x_1, x'_1] \times \ldots \times [x_{d-2}, x'_{d-2}] \times [\kappa(x_{d-1}), \kappa(x'_{d-1})]$, for a set of boxes M we define $\kappa(M) := \{\kappa(B) \mid B \in M\}$. Computing $\mathrm{Vol}^{d-1}_{\kappa(R)}(\mathcal{U}(\kappa(M)))$ is then the compressed problem on the right hand side in Figure 2. We now have $\mathrm{Vol}^{d-1}_{R}(\mathcal{U}(M)) = \mathrm{Vol}^{d-1}_{\kappa(R)}(\mathcal{U}(\kappa(M))) + Z \cdot \prod_{j=1}^{d-2} |R_j|$. Observe that we can compute $\kappa(R), \kappa(M)$ and Z in time $\mathcal{O}(n^{1/3})$, as we have sorted lists of coordinates.

In the problem $\mathrm{Vol}^{d-1}_{\kappa(R)}(\mathcal{U}(\kappa(M)))$ all partially covering boxes of $\kappa(R)$ are vertex-containing piles or vertex-containing quasi-piles, as we cut out the $(d-1)$-piles and all other boxes where vertex-containing before. Note that the function κ may destroy the property of being a cube, but we did not require this for the reduced base case. It follows that the problem of computing $\mathrm{Vol}^{d-1}_{\kappa(R)}(\mathcal{U}(\kappa(M)))$ is a reduced base case. This finishes the reduction.

5 Solving the reduced base case

In the last step we are left with solving the reduced base case: Compute $\mathrm{Vol}^{d}_{R}(\mathcal{U}(M))$ of a set of boxes M consisting of vertex-containing piles and vertex-containing quasi-piles for some $d \in \mathbb{N}$. It will be shown how to solve this problem in time $\mathcal{O}(m)$ for $m = |M|$, d constant. We may assume without loss of generality that each box in M is contained in R. This simplifies the representation of sets of piles and sets of quasi-piles.

5.1 The static problem

We will call the reduced base case the *static problem*. The reason for this is that we solve it by sweeping along dimension d, which leaves us with a dynamic problem. As described in more detail in the previous section, we sweep a horizontal plane along dimension d from y_1 to y_2 for $R_d = [y_1, y_2]$, stopping at each d-th coordinate z_i of a box in M. Between two such stops the $(d-1)$-dimensional volume v_i stays the same, so that we get $\mathrm{Vol}_R(\mathcal{U}(M))$ as $\sum_i v_i(z_{i+1} - z_i)$. In the dynamic problem of computing those v_i all (j,k)-quasi-piles and j-piles of R, $j,k < d$, are contributing to all v_i, so they are static, and all (j,d)-quasi-piles become dynamic j-piles, that may get inserted or deleted. Observe that each (j,d)-quasi-piles is either already present in the initialization of the dynamic problem and may be deleted, or may be inserted but never deleted, as we are facing vertex-covering (quasi-)piles. A d-pile of R becomes a fully covering box in the $(d-1)$-dimensional problem. As computing v_i is trivial if a d-pile B is active, i.e., if $(z_i, z_{i+1}) \subseteq B_d$, we may assume that there are no fully covering boxes in the dynamic problem.

With this sweep approach we get

$$T_{stat}(m,d) \leq T_{dyn}(m, d-1) + \mathcal{O}(m), \tag{1}$$

where $T_{stat}(m,d)$ denotes the run-time of our algorithm for solving the static problem on m boxes in d dimensions, and $T_{dyn}(m,d)$ denotes the run-time for solving the dynamic problem in d dimensions, where the number of boxes we initialize the problem with plus the number of updates is bounded from above by m.

5.2 The dynamic problem

More precisely, the *dynamic problem* is the following: On initialization we get a region $R \subseteq \mathbb{R}^{d'}$, a set of vertex-containing quasi-piles Q and a set of vertex-containing piles P^0 and return $\mathrm{Vol}_R^{d'}(\mathcal{U}(Q \cup P^0))$. There are two types of updates: Firstly, a vertex-containing pile may be inserted, call the set of these inserted piles P^1. Secondly, a pile from P^0 may be deleted. After each update we have to return the current volume $\mathrm{Vol}_R^{d'}(\mathcal{U}(Q \cup P^0 \cup P^1))$.

In the remainder of this section we will describe how to solve the dynamic problem. We need to define some notions first. Beside the dimensions $1, \ldots, d'$ we will speak of the $2d'$ *directions* $+1, -1, \ldots, +d', -d'$. Intuitively, $\pm i$ points to $\pm\infty$ along dimension i. We define the *opposite direction* $\bar{r}$ as $-i$ if $r = +i$ and $+i$ if $r = -i$. We say that two directions r, r' are *perpendicular*, $r \perp r'$, if $r' \neq r$ and $r' \neq \bar{r}$. We can canonically assign to each vertex-containing pile a direction: An i-pile B gets assigned to direction $+i$, if the interval B_i contains the upper i-th coordinate of R, and it gets assigned to $-i$, if B_i contains the lower i-th coordinate of R. It will be more convenient to speak of *r-piles* in the remainder, for directions r, not i-piles, for dimensions i. We can do the same for quasi-piles: An (i,j)-quasi-piles Q gets assigned the directions $r = \pm i$, $r' = \pm j$, depending on whether Q_i contains the upper or lower i-th coordinate of R (and the same for j), so that we can speak of an *(r,r')-quasi-pile*. As shorthands we write $P = P^1 \cup P^0$ and P_r^0 for the r-piles in P^0, similarly for P_r^1 and P_r. We will store only the sets Q, P_r^0 and P_r^1 explicitly. Iterating through all boxes can then be accomplished by iterating through these $4d' + 1$ sets "in parallel".

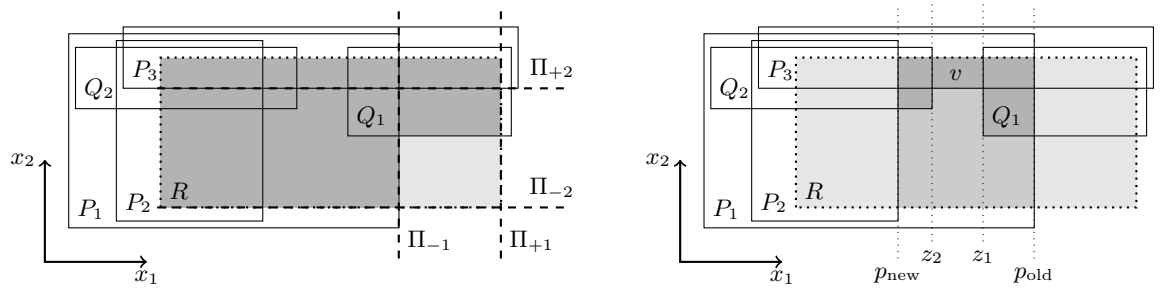
For a direction $r = +i$ we define the *rim* to be the plane Π_r defined by the equation $x_i = \min\{y_i \mid (y_1, \ldots, y_{d'}) \in \mathcal{U}(P_r)\}$, or $x_i = u_i$, if the set P_r is empty and u_i is the upper i-th coordinate of R. For a direction $r = -i$ we replace min by max and u_i by ℓ_i, the lower i-th coordinate of R. This is the plane that splits R into the part covered by r-piles and the part not influenced by r-piles. We take any r-pile that has non-empty intersection with Π_r (normally, this pile will be uniquely determined) and call it the *rim-defining* pile of direction r. Figure 3(a) shows an example with the rims indicated. The rim-defining piles of directions -1 and $+2$ are P_1 and P_3, respectively, the other two directions do not have a rim-defining pile.

Initialization: We are given the sets Q and P^0 and have to return $\mathrm{Vol}_R^{d'}(\mathcal{U}(Q \cup P^0))$, so we simply solve this as a static problem. This costs time $T_{stat}(m, d') + \mathcal{O}(m)$.

Moreover, we initialize dynamic problems D_r for each direction $r = \pm i$ with the $(d'-1)$-dimensional region $\prod_{j \neq i} R_j$ and all boxes in $Q \cup P \setminus (P_r \cup P_{\bar{r}})$ intersecting Π_r, the rim in this direction. During the algorithm the following invariant I_r will hold for all directions r after the initialization and each update: The dynamic problem D_r contains exactly the boxes in $Q \cup P \setminus (P_r \cup P_{\bar{r}})$ that intersect Π_r. We will show that initialization and updates of the instance D_r are of the form we defined above for the dynamic problem, if we re-initialize D_r at the right point.

Insertions and Deletions: When we have to delete an r-pile B, we delete it from P_r^0 (assuming that we are given a pointer to B in P_r^0 this can be done in $\mathcal{O}(1)$). It may be that it was the rim-defining pile in its direction. Then the volume $\mathrm{Vol}_R^{d'}(Q \cup P)$ changes. Figure 3(b) shows this situation, where the old rim was at p_{old} and the new will be at p_{new}. We may do a differential volume update, as we already have computed $\mathrm{Vol}_R^{d'}(Q \cup P)$ before the deletion. For this we need the volume v overlapped by boxes in $Q \cup P$ in the subregion from p_{old} to p_{new} of R. This volume can be computed by a sweep: We sweep a plane Π parallel to Π_r from p_{old} to p_{new} stopping at all coordinates of boxes in Q. We need to compute the $(d'-1)$-dimensional volume of the cross-section $\Pi \cap R \cap \mathcal{U}(Q \cup P \setminus (P_r \cup P_{\bar{r}}))$. This is exactly what the dynamic problem D_r gives us, so we insert or delete the boxes in Q we stop at into or from problem D_r. This way, we can do the sweep by utilizing the lower dimensional dynamic problem D_r. At the end of this sweep the invariant I_r is fulfilled again.

Additionally, we have to update the dynamic problems in the other directions: We delete the r-pile B from $D_{r'}$ for any direction $r' \perp r$. After this, invariant $I_{r'}$ holds again.



(a) An example with rims indicated

(b) The deletion of P_1 demands a sweep

Figure 3: You see an example of the dynamic problem with piles P_1, P_2, P_3 and quasi-piles Q_1, Q_2. In part (b) pile P_1 gets deleted, so that we have to compute the volume v for a differential volume update. We do a sweep from p_{old} to p_{new} stopping at z_1 and z_2 to delete Q_1 and insert Q_2 into D_{-1}. We also have to delete P_1 from D_{+2} and D_{-2}.

An insertion of a pile works analogously. Here, we need that we only insert a pile B, if it is not already contained in some other pile B' in P^1, i.e., if $B \cap R \not\subseteq B' \cap R$. If we do this, then any newly inserted r-pile is "the largest" in P_r^1, so that we can insert it in time $\mathcal{O}(1)$ into P_r^1.

D_r is really a *dynamic problem*: Note that the sequence of insertions and deletions of r-piles that actually change the volume inside R has a very simple form: It may start with some deletions. Eventually, an insertion changes the volume inside R, implying that the rim-defining pile will be one of P_r^1. From this point on no box in P_r^0 can be rim-defining anymore, so that no deletions will change the volume in R. Hence, this sequence of insertions and deletions starts with a row of deletions followed by a row of insertions. This also shows that the rim is monotonically decreasing up to some point, and is monotonically increasing afterwards. Observe that during

each of these two phases the updates to the dynamic problem D_r are of the demanded form: Each pile in $P \setminus (P_r \cup P_{\bar{r}})$ always intersects Π_r, and they are either there at the initialization, so also at the initialization of D_r, and may get deleted, or may get inserted but never deleted. Each (r', r'')-quasi-pile in Q, $r \perp r', r''$, always intersects Π_r, so always is in D_r. An (r', r)-quasi-pile with $r' \perp r$ may get inserted into D_r when sweeping Π_r during the deletion of an r-pile. But then it is never deleted from D_r in a deletion of an r-pile anymore. Similar statements hold for insertions and $(r', \bar{r})$-quasi-piles.

We use a trick now to make all dynamic problems of the demanded form: At the point we switch to the second phase in direction r (which is easy to recognize) we re-initialize problem D_r. Without this re-initialization D_r would not necessarily be of the demanded form, as a quasi-pile might get inserted into D_r during the deletion of a pile and get deleted from D_r again during the insertion of a pile. With this trick, the dynamic problems are of the demanded form and we now have at most $4d'$ of them: For all of the $2d'$ directions we may initialize the problem twice.

A Detail: So far during a sweep of an insertion or deletion of an r-pile we considered only the boxes in Q and $P \setminus (P_r \cup P_{\bar{r}})$ for computing the differential volume update. However, the piles in $P_{\bar{r}}$ may also influence this volume: If the two rims of r and $\bar{r}$ cross each other we get $\mathcal{U}(P_r \cup P_{\bar{r}}) = R$, but this is not taken into account by our differential volume update so far (and we would probably return a volume greater than $\mathrm{Vol}(R)$). This detail has a simple fix: During each sweep we have another stop at the point where the rim crosses its opposite rim, if this happens at all. If this cross is of the form, that $\mathcal{U}(P_r \cup P_{\bar{r}}) = R$ afterwards, then we note this and return at the end of the update not the volume we computed, but $\mathrm{Vol}(R)$. If the cross is of the form, that $\mathcal{U}(P_{r'} \cup P_{\bar{r'}}) \subsetneq R$ for each direction r' afterwards, then we reset the current volume to $\mathrm{Vol}(R)$ at this point. This simple fix can be seen to work trivially. It ends our description of the solution of the dynamic problem.

Run-time: To compute the required run-time we note that the number of boxes in each of the $4d'$ dynamic problems is bounded by m and all sweeps (and the rest) can be implemeted in linear time, as we have sorted lists of coordinates. Hence, we get

$$T_{\mathrm{dyn}}(m, d') \leq T_{\mathrm{stat}}(m, d') + 4d' \cdot T_{\mathrm{dyn}}(m, d' - 1) + \mathcal{O}(m).$$

Using the bound (1) we get

$$T_{\mathrm{dyn}}(m, d') \leq (4d' + 1) \cdot T_{\mathrm{dyn}}(m, d' - 1) + \mathcal{O}(m).$$

As d' is constant, this solves to $T_{\mathrm{dyn}}(m, d') = \mathcal{O}(m)$. Here, we used the observation that the dynamic problem can be solved in linear time in dimension 1: After each update we have to know the rim-defining $+1$-pile and the rim-defining -1-pile to compute the current volume inside the region. But this is simple since we are given sorted lists of coordinates.

6 Conclusion

In this paper we reduced the exponent β in the run-time of $\mathcal{O}(n^\beta)$ for Klee's measure problem on cubes from $\beta = \frac{d}{2} + \mathcal{O}(1)$ to $\beta = \frac{d}{3} + \mathcal{O}(1)$.

However, it is not clear so far, whether an algorithm for C-KMP needs a run-time of $n^{\Theta(d)}$. Proving W[1]-hardness of C-KMP would give such a result, conditioned on certain complexity theoretic assumptions, but this remains open. On the other hand, there may also be a "fast" algorithm for this problem, waiting to be found.

References

[1] P. K. Agarwal, H. Kaplan, and M. Sharir. Computing the volume of the union of cubes. In *Proc. 23rd annual Symposium on Computational Geometry (SoCG '07)*, pp. 294–301, 2007.

[2] J. L. Bentley. Algorithms for Klee's rectangle problems, 1977. Department of Computer Science, Carnegie Mellon University, Unpublished notes.

[3] J.-D. Boissonnat, M. Sharir, B. Tagansky, and M. Yvinec. Voronoi diagrams in higher dimensions under certain polyhedral distance functions. In *Proceedings of the eleventh annual symposium on Computational geometry (SoCG '95)*, pp. 79–88, 1995.

[4] K. Bringmann and T. Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. In *Proc. 19th International Symposium on Algorithms and Computation (ISAAC '08)*, Vol. 5369 of *Lecture Notes in Computer Science*, pp. 436–447. Springer, 2008.

[5] T. M. Chan. Semi-online maintenance of geometric optima and measures. *SIAM J. Comput.*, 32:700–716, 2003.

[6] T. M. Chan. A (slightly) faster algorithm for Klee's measure problem. *Computational Geometry: Theory and Applications*, 2009+. To appear, preliminary version appeared in *Proc. 24th ACM Symposium on Computational Geometry (SoCG '08)*.

[7] J. Erickson. Klee's measure problem, 1998. http://compgeom.cs.uiuc.edu/∼jeffe/open/klee.html.

[8] M. L. Fredman and B. W. Weide. On the complexity of computing the measure of $\bigcup[a_i, b_i]$. *Commun. ACM*, 21:540–544, 1978.

[9] H. Kaplan, N. Rubin, M. Sharir, and E. Verbin. Counting colors in boxes. In *Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '07)*, pp. 785–794, 2007.

[10] V. Klee. Can the measure of $\bigcup[a_i, b_i]$ be computed in less than $O(n \log n)$ steps? *American Mathematical Monthly*, 84:284–285, 1977.

[11] J. Nešetřil and S. Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26:415–419, 1985.

[12] M. H. Overmars and C.-K. Yap. New upper bounds in klee's measure problem. *SIAM J. Comput.*, 20:1034–1045, 1991.

[13] S. Suzuki and T. Ibaraki. An average running time analysis of a backtracking algorithm to calculate the measure of the union of hyperrectangles in d dimensions. In *Proc. 16th Canadian Conference on Computational Geometry (CCCG '04)*, pp. 196–199, 2004.