Computing Unitary Covers for Arbitrary Triangles

Will van Noordt

April 6, 2019

The Problem

Consider a triangle $T = \{x_1, x_2, x_3\}$ with $x_1, x_2, x_3 \in \mathbb{C}$ (here, the space $\mathbb{C}$ is used for notational convenience), along with its interior. Define the *unitary cover* of T as the minimal union of unit squares $S_{h,k} = (h, h+1) \times (k, k+1)$ such that

$$T \subset \bigcup_{h,k \in \mathcal{R}(T)} S_{h,k},$$

with $\mathcal{R}(T)$ denoting the *unitary cover set* of T. A visualization of a unitary cover set can be found in Figure 1.

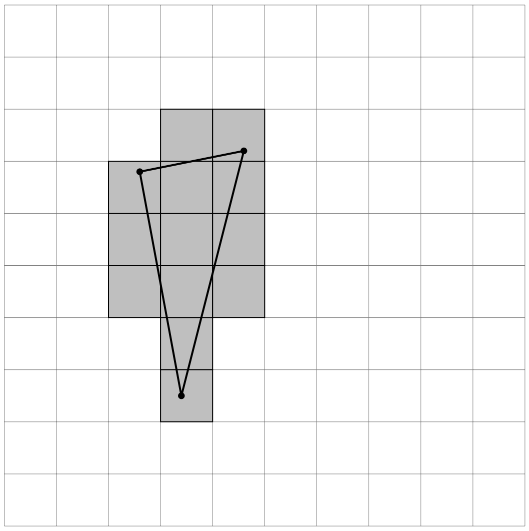

Figure 1: A unitary cover (grey) of an arbitrary triangle.

Method

It will be advantageous to reduce the computational domain of the problem for the sake of efficiency. Therefore, enforce the bounds

$$h_l = \min(\lfloor \mathrm{Re}(x_p) \rfloor) \le h < \max(\lceil \mathrm{Re}(x_p) \rceil) = h_u \qquad \text{and} \qquad k_l = \min(\lfloor \mathrm{Im}(x_p) \rfloor) \le k < \max(\lceil \mathrm{Im}(x_p) \rceil) = k_u. \tag{1}$$

Note that the strict right-hand inequality in (1) comes from the fact that all unit squares are considered from the "lower-left" corner: see definition of $S_{h,k}$. These bounds can be visualized in Figure 2.

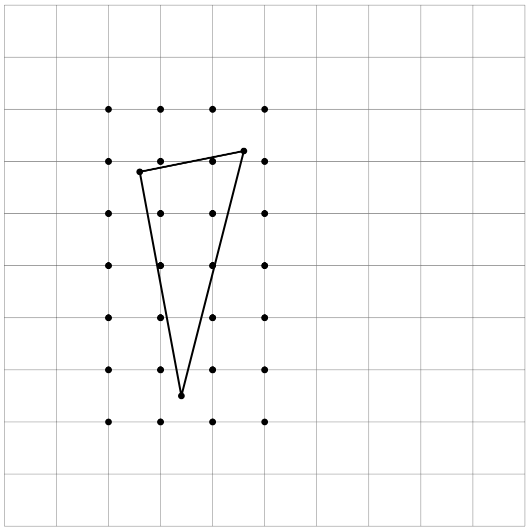
Figure 2: Computational bounds for this triangle.

Now, consider the following statement.

Claim: A unit square $S_{h,k}$ contains at least one point of a triangle T if any of the following are satisfied:

(i) $S_{h,k}$ contains a vertex of T,

(ii) One of the corner points of $S_{h,k}$ ("corner points" should have obvious meaning here) lies in T, or

(iii) For any 3 of the edges of $S_{h,k}$, there is a segment of the boundary of T (denoted here at ∂T) that intersects at least one of them.

Furthermore, (iv) if none of these properties are satisfied, then $(h,k) \notin \mathcal{R}(T)$.

Proof. The claims (i) and (ii) should be obvious, so only the proof of claims (iii) and (iv) are given. To prove claim (iii), first note that if there is no vertex of T in $S_{h,k}$, then the edge intersecting the boundary of $S_{h,k}$ intersects in two places (otherwise there would be a vertex of T in $S_{h,k}$). Therefore, it is sufficient to check only three of the edges of $S_{h,k}$: should it be the case that there is an intersection, then it is clear that this is the point of T in $S_{h,k}$. As for claim (iv), suppose by contradiction that $S_{h,k}$ contains a point x of T but does not satisfy (i), (ii), or (iii). It is obvious that $T \not\subset S_{h,k}$, or else property (i) would be satisfied. Since property (ii) is not satisfied, x is not on a corner point of $S_{h,k}$. So, by assumption, x is in the interior of $S_{h,k}$ and must lie on an edge t of T. This edge must pass through at least two sides of $S_{h,k}$, or else property (i) would be satisfied. So property (iii) must be satisfied. This is a contradiction. □

Now, the algorithm can be summarized as follows (where $(h,k) \to \mathcal{R}(T)$ denotes storing the indices in the unitary cover set and a--b denotes the segment from a to b in the complex plane):

```
$(\lfloor \mathrm{Re}(x_q) \rfloor, \lfloor \mathrm{Im}(x_q) \rfloor) \to \mathcal{R}(T)$
for $k_l \le k \le k_u$
   for $h_l \le h \le h_u$
      if $h + k\mathrm{i}$, $(h+1)+(k+1)\mathrm{i}$, $h+(k+1)\mathrm{i}$, or $(h+1)+k\mathrm{i} \in T$
         $(h,k) \to \mathcal{R}(T)$
      end
      else if one of $(h+k\mathrm{i})$--$(h+(k+1)\mathrm{i})$,$(h+k\mathrm{i})$--$((h+1)+k\mathrm{i})$,
      or$((h+1)+(k+1)\mathrm{i}$--$(h+(k+1)\mathrm{i})$ intersects $\partial T$
         $(h,k) \to \mathcal{R}(T)$
      end
   end
end
```

The results of this algorithm can be seen in Figure 3. A gallery of examples can also be found in the appendix.

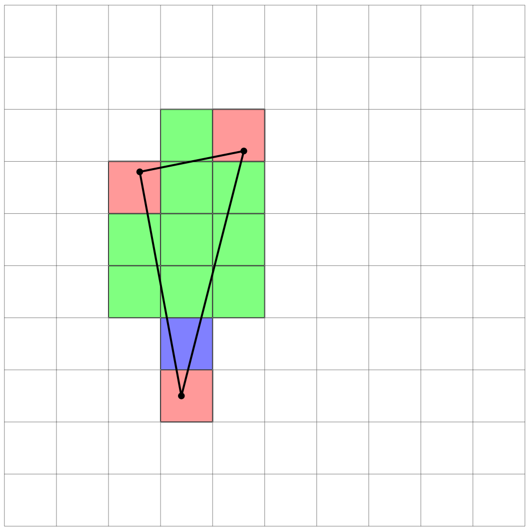

Figure 3: The unitary cover of T. The conditions are color-coded: condition (i) is red, (ii) is green, and (iii) is blue.

Remark

Determining whether or not a point is in a triangle is simple and is documented here:

`https://www.geeksforgeeks.org/check-whether-a-given-point-lies-inside-a-triangle-or-not/.`

The process for determining whether or not two segments intersect is also fairly simple, documented here:

`https://www.geeksforgeeks.org/check-if-two-given-line-segments-intersect/.`

This algorithm is not designed to be efficient on a large scale, but with some effort, it could be. This process has been documented because as far as the author is aware, there is no current documentation on this process as of the time of writing.

Appendix: Gallery