MSKDE - Using Marching Squares to Quickly Make High Quality Crime Hotspot Maps

José Florencio de Queiroz Neto, Emanuele Santos, Creto Augusto Vidal
Department of Computing
Federal University of Ceará
Fortaleza, Brazil
Contact: {florencio, emanuele, cvidal}@lia.ufc.br

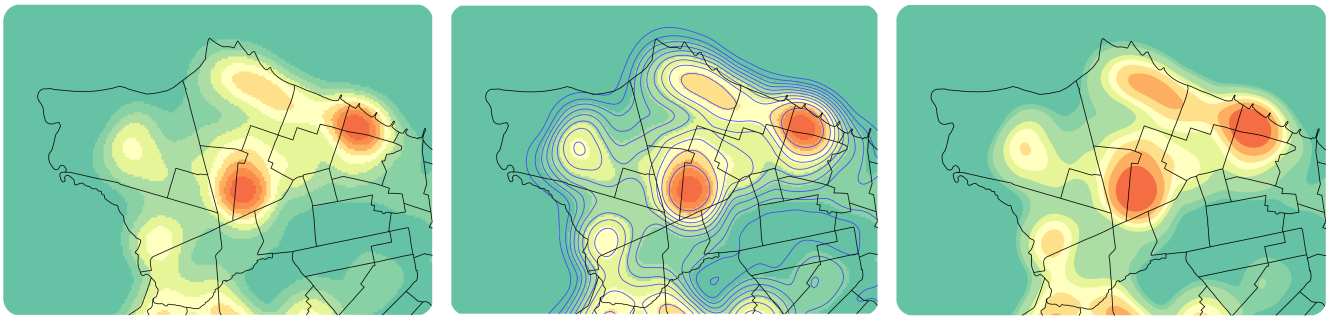

Figure 1. Marching Squares Kernel Density Estimation (MSKDE) is a technique to quickly create high quality, attractive, hotspot maps. On the left, a traditional crime hotspot map was created using KDE only. In the middle, contour lines, computed using Marching Squares, delineate a few predefined crime density levels. On the right, the original Kernel Density Estimation surface was removed and a sequential segmented colormap was used to finish the hotspot map. The hotspot map generated with MSKDE is less anomalous than the original KDE map and was generated much faster than traditional KDE hotspot maps with a similar level of accuracy, which require smaller cell sizes.

***Abstract*—In recent years, violence has considerably increased in the world. In a certain state of Brazil, for example, the homicide rate grew from 16 homicides per 100,000 inhabitants in 2000, to 48 homicides per 100,000 inhabitants in 2014. Police departments worldwide use various types of crime maps, which are generated with diverse techniques, in order to analyze and fight crime. Those types of maps enable decision makers to identify high-risk areas and to allocate resources more effectively. Hotspot maps, in particular, are crime maps often available in visual interactive systems for crime analysis. In order for hotspot maps to be really useful, they need to be very accurate - specially for resource allocation tasks - and to be processed very fast for quick analysis of different scenarios. In this paper, we propose MSKDE - Marching Squares Kernel Density Estimation, a solution for generating fast and accurate hotspot maps. We describe the technique and demonstrate its superior qualities through a careful comparison with the standard Kernel Density Estimation technique, which is widely used for generating hotspot maps.**

***Keywords*-Hotspot maps; Visualization.**

I. Introduction

Crime has become a central problem in many countries in the world. In Brazil, crime is a theme of growing interest. Violent crimes, in particular, have dramatically increased in some parts of Brazil in recent decades [1], challenging governments and the whole society [2].

Modern countries spend a lot of money to combat crime every year. For instance, USA, France, Germany and Brazil spend annually more than 1% of their GDP on public safety [1]. In reality, the total cost of crime is much higher than just the amount spent on public safety. In Brazil, for example, the total cost associated with violent crimes in 2013, including public safety, state prison maintenance and social costs, amounted to 100 billion dollars, corresponding to 5.4% of Brazil's GDP [1].

In the case of crime prevention, resources must be deployed at areas of higher crime likelihood, the so-called hotspot areas. Predicting those places is a basic problem of criminology, the scientific study of crime and criminals [3].

It is well known that crimes do not occur randomly and do not spread uniformly through space. In fact, the offenders usually search carefully the place and the time for their actions. Certain conditions make crime more frequent in some places than in others [4]–[8]. Therefore, the spatial distribution of crime likelihood is one of the main features in crime prediction.

Hotspot maps are widely recognized as effective tools used by the police [9]. Braga [10] investigated the effectiveness of that tool and attested that the police was able to achieve a significant reduction on crime and disorder. Boba Santos [11] states that the most important map for criminal analysis is the hotspot map based on Kernel Density Estimation (KDE) [12]. KDE is a technique for nonparametric estimation of a probability density function and has the ability to expose aggregated

and sparse regions, a useful feature for crime analysis.

The inherent difficulties when dealing with complex crimes, international terrorism, and increasingly sophisticated criminal organizations pushed governments to develop computational tools to deal with that special scenario. After the September 11 attack, visual interactive systems have gained prominence and have been increasingly adopted for dealing with complex and dynamic problems, such as criminal analysis.

In order to provide a high level of productivity and a good user experience, visual interactive systems must produce quick and high-quality responses to user inquiries [13]. Accuracy is also important for visual interactive systems, since responses that in one hand are quick but on the other hand are inaccurate can lead to swift wrong decisions. In contrast, accurate but slow systems are not capable of following the dynamics of the environment, and can hinder decision making, causing the response to arrive too late.

Unfortunately, speed and accuracy are not easily obtained simultaneously in crime hotspot map generation. Particularly using KDE, in order to produce a map at interactive speed, the cell size of the domain's discretization cannot be small, which ends up in a pixelated and not accurate hotspot map.

In order to obtain a hotspot map quickly without losing accuracy, we propose MSKDE (Marching Squares Kernel Density Estimation), a fast technique to transform a low resolution KDE hotspot map into a new hotspot map based on contour lines (see Figure 1). The outcome of MSKDE is a hotspot map with rounded and smooth boundaries, which can be generated quickly, with an accuracy similar to KDE hotspot maps made with smaller cell sizes. MSKDE maps, with their smoother boundaries, are a more natural representation of the real world's hotspots than the tiled looking original KDE maps. We will demonstrate in an experiment that MSKDE maps preserve accuracy at lower computational cost when compared with traditional KDE maps with similar levels of accuracy.

The remainder of this paper is organized as follows. In Section II, we review related work. In Section III, we present background knowledge about KDE and Marching Squares. In Section IV, we describe MSKDE. In Section V, we detail an experiment and a define a metric to compare MSKDE with the traditional KDE. In Section VI, we present and analyze the results of the experiment. Finally, in Section VII, we present conclusions of our work, and outline directions for future work.

II. Related work

Although the use of KDE for creating crime hotspot maps has been extensively studied in the last decades, only the aspects of the creation process and professional use were investigated and very little attention has been paid to map generation time, human perception and suitability for use in visual interactive systems.

Williamson et al. [14] present a technique for creating a crime map based on KDE where the bandwidth is determined by the mean of the distance of each crime to its k-nearest neighbors. The authors claim that their method is the first nonarbitrary way of determining the bandwidth, but a crime analyst still has to choose a suitable value for the parameter k.

The inclusion of a temporal weight in the KDE kernel function was investigated by Ratcliffe [15], Bowers et al. [16] and Mohler [17], when the authors tried to create an integrated framework with space and time dimensions. This strategy seems to be promising for creating specialized hotspot maps for regions with temporary crime density. Temporary and chronic crime density are concepts developed in [18].

Also, the construction of the KDE crime hotspot map is studied by Chainey et al. [19] from a statistical point of view, by Hart and Zandbergen [20] focusing on the best parameter values and best kernel function for the purpose of crime prediction, and by Chainey et al. [21] with a focus on the types of crime that can be better predicted. We believe that only a few possibilities in terms of cell size and bandwidth were tested because of their very laborious process and long execution time. A more automated process would benefit from our proposed solution, which enables testing a broader set of parameter configurations in less execution time.

In the visual analysis area, Malik et al. [22] present VALET, a visual interactive system for analysis of law enforcement spatiotemporal data, through maps built using Kernel Density Estimation. Newer versions of that system include improvements like correlation analysis and the event time as a new dimension [23], and a new multidimensional Gaussian kernel based on the k-nearest events [24]. VALET is a comprehensive system, but as far as we know, it does not include any technique for building maps quickly, except when using a large cell size. We also noticed that the kernel function used in VALET does not generate smooth surfaces and the map accuracy is not verified. Besides proposing a technique to build a hotspot map quickly, we also propose a metric that evaluates the accuracy of the generated map compared with a reference map. This metric is described in Section V.

III. Background

In this Section we describe the KDE technique and the Marching Squares algorithm, as they are both needed to build our solution.

A. Kernel Density Estimation

Kernel Density Estimation (KDE) is a technique for non-parametric estimation of a Probability Density Function (PDF). It was initially developed by Rosenblatt [12] for unidimensional random variables. However, it can be extended to more dimensions – in our case, we deal with two dimensions.

The idea behind KDE is that the random variable being studied has an inherent PDF that can be statistically estimated, based on sample data that is supposed to be generated by the real density function itself [25], [26].

In the case of two dimensions, in which we have events scattered on a region, KDE estimates the PDF by dividing the area into a regular grid of cells, and calculating for each cell, using a kernel function, a density value. The kernel function

calculates the density, summing contributions of all events that are located within a certain distance (bandwidth) from the cell center. The result is a two-dimensional scalar field that estimates the PDF, and it is the basis for a hotspot map. Figure 2 illustrates this process showing an example for a single cell in a two-dimensional field.

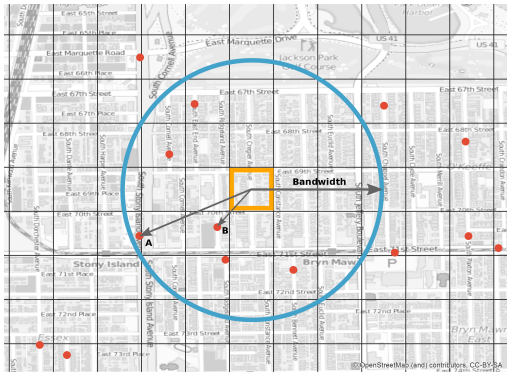

Figure 2. How the density in one cell (in orange) is calculated using KDE: for each event, represented by a point, inside the bandwidth circle, the algorithm computes its distance to the cell center divided by the bandwidth and sends it to the kernel function. Common kernel functions consider that nearer events (e.g. B) contribute more to the cell density than distant events (e.g. A). Finally, the density is computed as the sum of contributions of all events that are inside the bandwidth.

From the explanation above, it follows that KDE has three parameters that directly affect the resulting map: the cell size, the bandwidth and the kernel function.

- **Cell size** impacts map resolution, which affects appearance, accuracy and execution time. Small cells lead to more accurate, smooth and good-looking maps that are slow to compute. Large cells, in turn, lead to less accurate, pixelated maps that are fast to compute;
- **Bandwidth** impacts the degree of aggregation and cluster formation. Small bandwidths lead to spotty maps, because the events will only contribute to nearby cells. On the other hand, large bandwidths aggregate more points, forming big clusters;
- **Kernel function** defines how the events will contribute to each cell density. Common functions return a value inversely proportional to the distance between the event and the center of the cell, giving more weight to events that are nearer the cell. The most used and recommended kernel function is the quartic [1] [20], [27].

The KDE algorithm can be implemented in two ways, which result in different computational complexities (n is the number of cells, e is the total number of events, s is the cell size, b is the bandwidth):

1) Based on cells: each cell is visited and its density is calculated using all events that are within the bandwidth. In this case, the complexity is $O(n \cdot e)$.
2) Based on events: each event is visited and all cells within its bandwidth are updated with its calculated contribution. In this case, the complexity is $O(e \cdot \pi \cdot ceil^2(\frac{b}{s}))$.

[1] $f(u) = \frac{15}{16}(1-u^2)^2$ for $0 \leq u \leq 1$, and u is a normalized bandwidth.

Input: A 2D scalar field with **h** rows and **w** columns, a contour value **v**.
Output: A set of edges **E**.

```
Load the lookup table L with the 16 possible cases of how the value v can be related to the vertices of a cell (as in Figure 4)
for i = 1 to h − 1 do
    for j = 1 to w − 1 do
        Define a logical cell c and define c.values with the field values of vertices (i, j),(i + 1, j),(i, j + 1) and (i + 1, j + 1)
        index ← build_index(c.values, v)
        case ← lookup(L, index)
        if case == 5 or case == 10 then {Ambiguous cases}
            if average(c.values) < v then
                Contour follows along lowest vertices (solid lines in cases 5 and 10 in Figure 4)
            else
                Contour follows along highest vertices (dotted lines in cases 5 and 10 in Figure 4)
            end if
        end if
        Build cell geometry for the case (edges and vertices)
        Linearly interpolate the edge vertices along c.values
        Store the interpolated edges in the set E
    end for
end for
return E
```

Figure 3. Marching Squares Algorithm

To optimize our implementation, we compare n with $\pi \cdot ceil^2(\frac{b}{cs})$ and choose the way that produces the fastest computation.

B. Marching Squares

Marching Squares is a very efficient algorithm for generating contour lines in a regular grid representing a two-dimensional scalar field. It is the 2D version of the 3D marching cubes algorithm [28]. As its name implies, it works by *marching* the cells in the field, processing each one of them independently. The values of the scalar field on the cell's vertices (corners) are analyzed and the geometry of the cell is computed based on a pre-built lookup table (see Figure 4). The final position of the contour is obtained by linear interpolation along the cell's vertices. Notice in Figure 4 that cases 5 and 10 are ambiguous cases. For simplicity, the disambiguation in our implementation was obtained by averaging the values on the corners: if the average is greater than or equal to the contour value, the contour will pass between the two highest vertices (dotted lines), otherwise the contour will pass along the lowest vertices (solid lines). Other possible solutions for disambiguation are described in [29], [30]. The Marching Squares algorithm is detailed in Figure 3.

Figure 5 displays a small example of Marching Squares being calculated. Note that the interpolation phase (on the

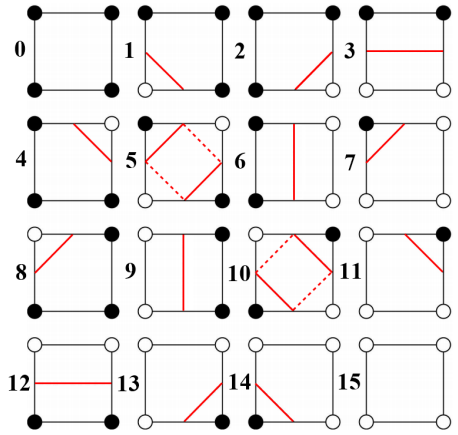

Figure 4. Marching Squares' 16 cases: Based on the cell's four vertex values, the contour can assume one of the 16 shown possibilities (0 to 15). Filled vertex dots mean that their values are greater than or equal to the contour value. Non-filled dots mean the opposite. In two cases (0 and 15), the contour line does not pass through the cell because the contour value is above or below all four vertex values. Special cases 5 and 10 are saddle points and hence, ambiguous. The disambiguation is solved using the average of the cell's four vertex values; if the average is greater than or equal to the contour value, the contour will cross the cell in each side of the ridge formed by connecting the two highest vertices (dotted lines), otherwise the contour will cross the cell in each side of the valley formed by connecting the lowest vertices (solid lines).

right) is separated from the cases' classification (in the middle) only for illustration purposes; in fact, they occur at the same time, to avoid other loop over the field.

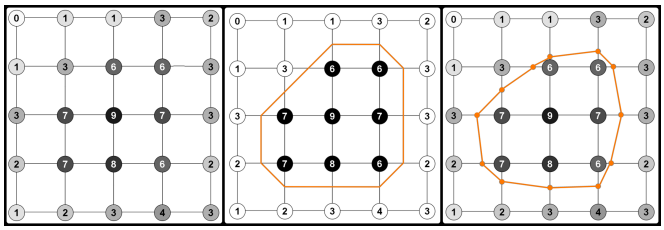

Figure 5. Marching Squares algorithm in action: on the left, a field where marching squares will be used to generate an isocontour representing value $v = 5$. In the middle, the result after "marching¨ all cells, each cell assumes one of the 16 cases in Figure 4 without any interpolation. On the right, the result of the interpolation phase, where the contour vertices are adjusted to cell vertex values.

IV. Our Approach: Marching Squares Kernel Density Estimation

Marching Squares Kernel Density Estimation (MSKDE) is a novel technique, presented in this paper, to generate high quality crime hotspot maps, based on a pre-built KDE hotspot map.

The basic idea behind MSKDE is that, as a crime hotspot map generated by KDE is a 2D field, we can apply Marching Squares to generate contour lines that are boundaries of heat levels, forming a new hotspot map.

A MSKDE hotspot map is generated according to the following steps:

Step 1 Select the spatial events for which we want to generate the hotspot map (Figure 6a);
Step 2 Select the KDE parameters: cell size, bandwidth, and kernel function;
Step 3 Generate the KDE hotspot map (Figure 6b);
Step 4 Select one or more contour values;
Step 5 For each selected contour value, apply the Marching Squares algorithm described in Figure 3 to the KDE map (Figure 6c);
Step 6 Apply a colormap to the regions defined by the contours (Figure 6d).

If the analysts do not want to select the contour values manually, they can use existing techniques for automatic selection. A usual choice is to equally divide the range of values of the field into a few segments and select the border values as contour values. Another option is to apply a one-dimensional clustering algorithm to the field values, forgetting the spatial components, partitioning them into groups, and again taking the border values as contours. In the example shown in Figure 6, the contour values were determined by one-dimensional k-means algorithm. Notice that a clustering algorithm is a good alternative to select contour values, because the border values will work as natural breaks. However, depending on the purpose of the map, percentiles could also be a good choice.

So far, when comparing the final result of MSKDE in Figure 6d with KDE in Figure 6b, it may seem that the only advantage of MSKDE is to draw smooth maps, but there are other advantages. In fact, the use of MSKDE in crime hotspot map generation is encouraged by three main aspects:

1) Research in crime hotspot map indicates interest in analysis of definite ranges of crime density (like the highest 5% or 10% of the field). That makes sense because crime hotpot maps are mainly used in predictive policing and resource allocation, where definite boundaries are easier to work with. MSKDE fulfills this requirement because the contour boundaries are definite delimiters of density levels.
2) Sometimes the pixelated appearance of KDE maps can lead to incorrect classifications of regions near boundaries. That could be solved by using a smaller cell size, but at the expense of increasing the execution time. MSKDE is better in that regard because the interpolation used to build the contours generates more realistic regions without a significant increase in execution time.
3) The round and smooth MSKDE delimiters bring a significant gain in accuracy. This accuracy can be converted to a gain in time, by applying MSKDE to low resolution KDE maps, speeding up the map generation and making possible to use better hotspot maps in interactive systems.

V. Experiment and Evaluation

In this section, we compare MSKDE with KDE in a typical task of the police department of a large city: generating a hotspot map to guide resource allocation and to define police patrol routes. The considered hotspot region is the highest 5%

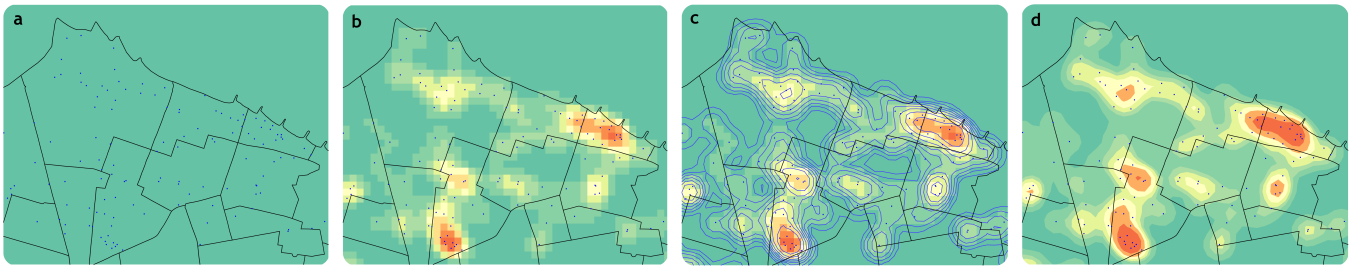

Figure 6. Overview of Marching Square Kernel Density Estimation: in (a) a set of spatial events is selected for the hotspot map. In (b) the KDE hotspot map is generated from the event set. In (c) we see a set of contour lines (in blue) generated after running Marching Squares. In (d) we see the final MSKDE map, after applying an appropriate colormap to regions defined by the contours.

of the field and the maps are compared with respect to two aspects: generation time and accuracy.

Our experiment consists in creating a set of MSKDE and KDE maps of the same region and events, in which we fixed the values of the bandwidth and of the kernel function and varied the cell size within a certain range to measure its effect on the execution time and on the appearance of the map.

For accuracy evaluation purposes, we compare the maps generated by both techniques with a reference hotspot map. This reference map represents the real crime probability density of the region. The ideal reference map would be a KDE map using a cell with infinitesimal size. However, since that is not practical, we use a KDE hotspot map with a very small cell size as reference map; in our case, 10 meters. All the data, parameter configuration and computer environment used are described below.

A. Data and parameters for the experiment

Our experiment was conducted with the following data and parameters:

- Spatial events: 2,916 homicide crimes occurred in a two-year period (2014 and 2015) in a large city in Brazil;
- KDE parameters:
  - Cell sizes: a range of 50 to 200 meters, with increments of 10 meters. This wide range enables more detailed comparisons between the techniques.
  - Bandwidth: 1000 meters. That value was chosen because, for applications such as resource allocation, a spotty small bandwidth map renders the decision-making process more difficult due to the excess of possibilities. Larger bandwidth values, such as 1000 meters, generate more continuous clusters, which is more appropriate to strategic tasks [31].
  - Kernel function: a quartic kernel function, following the trend of most studies in the field.
- Considered hotspot: the highest 5% of the field.
- Reference map for accuracy: a KDE hotspot map, generated from the same data, with the same bandwidth and kernel function, but with a cell size of 10 meters.[2]

[2]The generation of the KDE map for 2916 events and a 10-meter cell size took 28 minutes.

B. Computational Resources:

The computer and software used to run the experiments were:

- Computer: X64 compatible PC;
- Processor: Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz, 4 Cores;
- Memory: 16 GB of RAM;
- Operational System: Microsoft Windows 10 Pro X64;
- Software: Mathworks' Matlab v. 2015b.

C. Experiment

Our experiment consists of the following steps:

Step 1 Generate the KDE hotspot map with cell size of 10 meters, that is used as a reference in accuracy analysis.

Step 2 Classify the highest 5% cells of the KDE as hot and the remaining 95% as non-hot.

Step 3 Generate a MSKDE map and a KDE map for each cell size, in the range of 50 to 200 meters, with a 10-meter increment step, making a total of 32 maps.

Step 4 For each cell of the standard KDE reference map:

- get the coordinates of point located at the cell's center;
- find out whether the cell is classified as hot or non-hot;
- for each of the 32 generated maps:
  - increment a counter whenever the classificaiton of the cell in which that point falls is different from the classification of its cell in the standard KDE reference map.

After that experiment, we have, for each map, its generation time and the number of cells of the reference map that did not receive the same classificaiton in it. Those misclassified cells indicate the degree of malformation of the hotspot in a generated map. The hot cells that are misclassified as non-hot represent regions that will not be covered in the resource allocation, with, possibly, serious consequences. On the other hand, the non-hot cells that are misclassified as hot represent regions to which resources will be allocated unduly, indicating a waste of resources.

The map accuracy will be evaluated, indirectly, by evaluating the degree of anomaly of the map with respect to the

reference map. High anomaly indicates low accuracy, and vice versa.

To evaluate the anomaly between hotspot maps, we present the Hotspot Anomaly Index (HAI), an index that indicates how a hotspot map is anomalous with respect to a reference hotspot map. HAI is defined as 100 times the ratio of the number of cells in the reference map that are misclassified in a given map to the total number of cells classified as hot in the reference map. Note that we cannot evaluate a hotspot map only by its coverage of the hot cells of the reference map, because it would be easy to obtain a "100% accurate" map by just enlarging the hot area of the map under evaluation to cover its whole area. However, a hotspot map whose hot area covers the whole map is not really a hotspot map. For this reason, a hotspot map evaluation index must be with respect to the size of the hot area in the reference map.

The HAI formula is defined as

$$HAI = \frac{m}{n} \cdot 100, \tag{1}$$

where m is the number of cells in the reference map that are misclassified in the map being evaluated (considering the cell center), and n is the number of cells classified as hot in the reference map. The HAI index quantifies the anomaly of a hotspot as a percentage of the hot area in the reference map.

Figure **??** displays an example of HAI calculation.

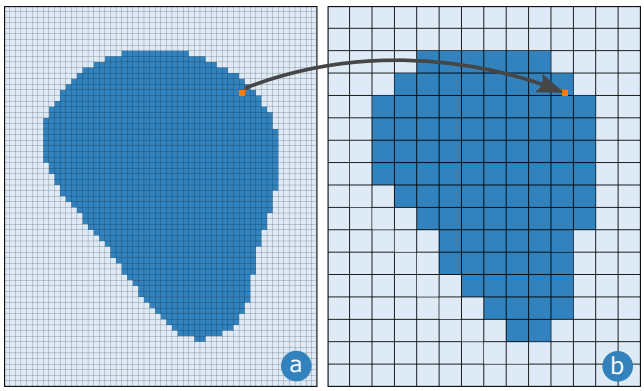

Figure 7. Hotspot Anomaly Index (HAI) calculation example: In **a**, we have a 68x56 reference hotspot map (3,808 cells) whose hot area, colored in blue, consists of 1,484 cells. In **b**, we have a 17x14 KDE hotspot map whose hot area consists of 94 cells. The HAI index of **b** with respect to **a** is calculated by counting how many of the 3,808 cells in **a** (e.g. the orange cell in **a** whose center is also marked as an orange spot in **b** pointed by the arrow) are classified differently in the map shown in **b**, dividing that count by the number of hot cells in the reference map in **a**, and multiplying that ratio by 100. In this example, 121 cells of **a** were misclassified in **b**, therefore, $HAI = \frac{121}{1484} \cdot 100 \approx 8.15\%$, which indicates that the hotspot map in **b** has an anomaly, corresponding, in size, to approximately 8.15% of the hot area of the reference map. Notice that for HAI calculation, the only information needed from map **b** is how this map classifies every cell of map **a**; the remaining information is taken from map **a**.

VI. Results and Discussion

The hotspot map used as reference was generated and it has 2802 rows and 2422 columns; with a total of 6,786,444 cells. The highest 5% values (338,423 cells), were classified as the hot part of the map.

All 16 KDE maps and 16 MSKDE maps were generated, with execution times presented in Table I.

Table I
Execution time: timings are expressed in seconds.

Cell Size	KDE	MSKDE
50	31.68	32.39
60	18.37	18.76
70	14.05	14.34
80	10.47	10.70
90	9.01	9.18
100	6.94	7.09
110	5.74	5.86
120	5.04	5.17
130	3.93	4.02
140	3.68	3.76
150	3.03	3.10
160	2.71	2.76
170	2.20	2.25
180	2.13	2.17
190	1.98	2.02
200	1.92	1.97

Figure 8 displays a chart with the execution time for each cell size and each map type.

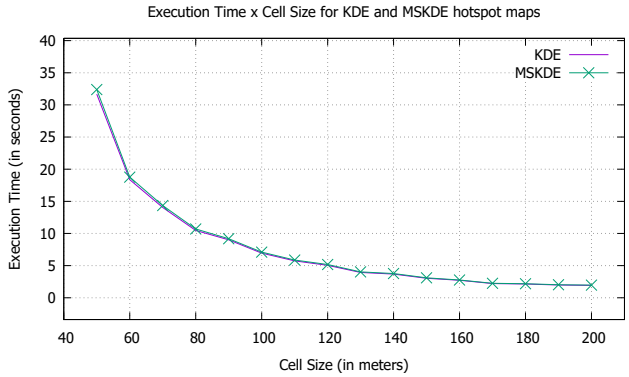

Figure 8. This chart shows the relation between execution time and cell size, for each kind of hotspot map.

Following the maps generation, the HAI indices for all 32 maps were calculated (see Table II).

Figure 9 displays a plot of the anomaly of the maps, for each analyzed cell size.

A. Discussion

1) Performance: On Table I, we report the execution times of MSKDE and KDE, for a range of cell sizes. We observe that, in average, the execution time of MSKDE is only 2.18% higher than that of KDE. This increase is not significant for most applications and is hardly noticeable in Figure 8.

The execution time, in both kinds of maps, has exponential decay as the cell size increases, and the time cost, when the cell size is small, is so high that its use in interactive systems is not recommended. For use in interactive systems, the cell size should have a minimum size that does not compromise the user's experience.

Table II
HOTSPOT ANOMALY INDEX: EXPRESSED IN PERCENTAGE OF THE HOT REGION IN THE REFERENCE HOTSPOT MAP.

Cell Size	KDE	MSKDE
50	3.32%	0.67%
60	4.04%	0.88%
70	4.84%	1.18%
80	5.56%	1.75%
90	6.43%	2.34%
100	7.15%	2.22%
110	7.87%	2.86%
120	8.49%	2.47%
130	9.36%	4.16%
140	10.70%	4.92%
150	11.08%	5.66%
160	11.99%	6.58%
170	13.01%	6.13%
180	13.59%	6.37%
190	14.76%	7.82%
200	15.81%	9.06%

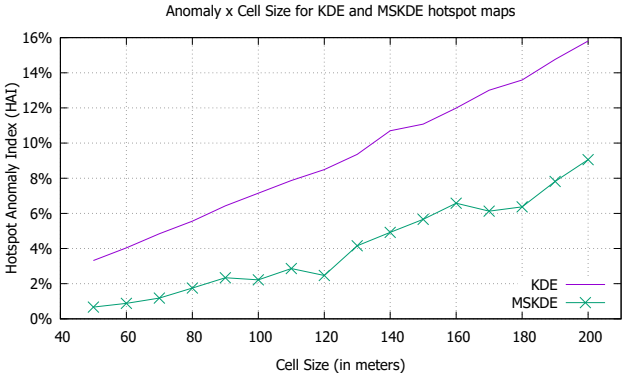

Figure 9. This chart shows the relation between the anomaly of the maps and their cell size, for each kind of hotspot map.

2) Accuracy: As observed in Table II and in Figure 9, for every cell size, MSKDE generates less anomalous (more accurate) maps than KDE.

Analyzing the chart in Figure 9, we can see that, for KDE and MSKDE maps with similar HAI, the cell size on MSKDE maps is significantly bigger than the cell size on KDE maps. This is very advantageous for MSKDE because maps with bigger cell sizes can be generated in less time (Table I and Figure 8).

For example, comparing a KDE map with cell size of 70 meters and a MSKDE map with cell size of 140 meters, both maps have similar HAI (4.84% and 4.92% respectively). However, while the KDE map is generated in 14.05 seconds, the MSKDE map is generated in only 3.76 seconds.

We compare the relationship between HAI and execution time for KDE and MSKDE in Figure 10. Notice that MSKDE outperforms KDE in every combination, whether providing a better execution time for similar anomaly or giving a more accurate map at the same execution time.

3) Limitations: In this work we have compared MSKDE and KDE in generating hotspots map with 5% of hot area and a bandwidth of 1000 meters. In our experiment, the MSKDE map with cell size of 50 meters is composed by 16 polygons,

Table III
RESULTS: CONSTRUCTION TIME (IN SECONDS) AND NUMBER OF POINT MISCLASSIFICATIONS (PM) FOR KDE AND MSKDE HOTSPOT MAPS.

KDE			MSKDE		
Cell Size	Time	HAI	Cell Size	Time	HAI
50	31.68	3.32%	120	5.17	2.47%
60	18.37	4.04%	130	4.02	4.16%
70	14.05	4.84%	140	3.76	4.92%
80	10.47	5.56%	150	3.10	5.66%
90	9.01	6.43%	160	2.76	6.58%
100	6.94	7.15%	170	2.25	6.13%
110	5.74	7.87%	180	2.17	6.37%
120	5.04	8.49%	190	2.02	7.82%
130	3.93	9.36%	200	1.97	9.06%

Figure 10. This figure shows the relationship between the Hotspot Anomaly Index (HAI) and the execution time for KDE and MSKDE maps. Notice that MSKDE outperforms KDE in every scenario, offering less anomaly at a fixed time and being faster for a fixed anomaly.

with a total of 2,362 edges.

Bandwidth is a parameter that has a major influence on the number of polygons, because as we saw in Section III-A, small bandwidths generate spotty maps, formed with more polygons. For example, if our experiment were done with a bandwidth of 400 meters, the MSKDE map would have been composed by 76 polygons, with a total of 4,502 edges. This larger number of polygons and edges can increase the execution time and delay operations such as coloring and rendering.

Other point that deserves attention is the additional cost incurred when it is necessary to plot a MSKDE map with more than one contour level. For example, in a case as shown in Figure 1 on the right, which has 9 contour levels, the part of the MSKDE process, responsible for calculating polygons, must run 9 times and the final map will have a lot of polygons and edges. This extra steps will add some cost to the final execution time.

VII. CONCLUSIONS AND FUTURE WORKS

In this paper, we introduced Marching Squares Kernel Density Estimation (MSKDE), a technique that quickly transforms low resolution KDE hotspot maps into a more attractive and accurate hotspot map. We demonstrated that MSKDE outperforms KDE not only by offering a less anomalous map in the same execution time, but also by generating a map with a predefined anomaly faster. Furthermore, we present the

Hotspot Anomaly Index (HAI), an index to evaluate the degree of anomaly between a hotsport map and a reference hotspot map.

Future works include the study of varying MSKDE parameters in order to prepare hotspot maps designed to predictive activities and incorporate the time event as a new dimension, forming an integrated spatiotemporal framework for hotspot map generation.

ACKNOWLEDGMENT

Our work has been funded by the MCTI/CNPQ/Universal grant number 459448/2014-5. Queiroz Neto has been supported by CAPES and FUNCAP. The authors would like to thank the Department of Public Security and Social Defense of the State of Ceara for providing the crime dataset.

REFERENCES

[1] F. B. de Segurança Pública, "Anuário brasileiro de segurança pública 2014," Tech. Rep., 2014, available at World Wide Web: "http://www.forumseguranca.org.br/produtos/anuario-brasileiro-de-seguranca-publica/8o-anuario-brasileiro-de-seguranca-publica".

[2] C. Izique. (2013) Crescimento da violência no país surpreende pesquisadores. [Online]. Available: http://exame.abril.com.br/brasil/noticias/violencia-democracia-e-direitos-humanos

[3] L. Siegel, *Criminology: Theories, patterns, and typologies*. Cengage Learning, 2010.

[4] L. W. Sherman, P. R. Gartin, and M. E. Buerger, "Hot spots of predatory crime: Routine activities and the criminology of place," *Criminology*, vol. 27, no. 1, pp. 27–55, 1989.

[5] J. E. Eck and D. Weisburd, "Crime places in crime theory," *Crime and Place*, vol. 4, pp. 1–33, 1995.

[6] L. W. Sherman, "Hot spots of crime and criminal careers of places," *Crime and Place*, vol. 4, pp. 35–52, 1995.

[7] S. Chainey and J. Ratcliffe, *GIS and crime mapping*. Wiley, 2005.

[8] J. Ratcliffe, "Crime mapping: spatial and temporal challenges," in *Handbook of quantitative criminology*, A. R. Piquero and D. Weisburd, Eds. Springer, 2010, ch. 2, pp. 5–24.

[9] J. H. Ratcliffe, "The hotspot matrix: A framework for the spatio-temporal targeting of crime reduction," *Police practice and research*, vol. 5, no. 1, pp. 5–23, 2004.

[10] A. A. Braga, "The effects of hot spots policing on crime," *The Annals of the American Academy of Political and Social Science*, vol. 578, pp. 104–125, November 2001.

[11] R. Boba Santos, *Crime Analysis With Crime Mapping*. SAGE Publications, 2012.

[12] M. Rosenblatt, "Remarks on Some Nonparametric Estimates of a Density Function," *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 832–837, 1956.

[13] J. Thomas and K. A. Cook, *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society Press, 2005.

[14] D. Williamson, S. McLafferty, V. Goldsmith, J. Mollenkop, and P. McGuire, "A better method to smooth crime incident data," *ESRI ArcUser Magazine*, 1999. [Online]. Available: http://www.esri.com/news/arcuser/0199/janmar99.html

[15] J. Ratcliffe, "Aoristic analysis: the spatial interpretation of unspecific temporal events," *International Journal of Geographical Information Science*, vol. 14, no. 7, pp. 669–679, 2000.

[16] K. J. Bowers, S. D. Johnson, and K. Pease, "Prospective hot-spotting the future of crime mapping?" *British Journal of Criminology*, vol. 44, no. 5, pp. 641–658, 2004.

[17] G. Mohler, "Marked point process hotspot maps for homicide and gun crime prediction in Chicago," *International Journal of Forecasting*, vol. 30, no. 3, pp. 491–497, 2014.

[18] W. Gorr and Y. J. Lee, "Longitudinal Study of Crime Hot Spots," Carnegie Mellon University, Heinz College Research, Tech. Rep., 2012.

[19] S. Chainey, S. Reid, and N. Stuart, "When is a Hotspot a Hotspot? A procedure for creating statistically robust hotspot maps of crime," in *Innovations in GIS 9: Socio-economic applications of geographic information science*, D. Kidner, G. Higgs, and S. White, Eds. Taylor and Francis, 2002.

[20] T. Hart and P. Zandbergen, "Kernel density estimation and hotspot mapping: Examining the influence of interpolation method, grid cell size, and bandwidth on crime forecasting," *Policing: An International Journal of Police Strategies & Management*, vol. 37, no. 2, pp. 305–323, 2014.

[21] S. Chainey, L. Tompson, and S. Uhlig, "The utility of hotspot mapping for predicting spatial patterns of crime," *Security Journal*, vol. 21, pp. 4–28, 2008.

[22] A. Malik, D. S. Ebert, R. Maciejewski, and T. F. Collins, "Visual Analytics Law Enforcement Toolkit," in *IEEE International Conference on Technologies for Homeland Security*. IEEE, 2010, pp. 222–228.

[23] A. Malik, R. Maciejewski, N. Elmqvist, Y. Jang, D. S. Ebert, and W. Huang, "A correlative analysis process in a visual analytics environment," in *IEEE Symposium on Visual Analytics Science and Technology 2012, October 14-19, Seatle, WA, USA*, 2012, pp. 33–42.

[24] A. Malik, R. Maciejewski, S. McCullough, D. S. Ebert, and S. Towers, "Proactive Spatiotemporal Resource Allocation and Predictive Visual Analytics for Community Policing and Law Enforcement," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1863–1872, December 2014.

[25] C. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. Springer, 2006.

[26] B. W. Silverman, *Density estimation for statistics and data analysis*. Chapman and Hall, 1986.

[27] N. Levine, *CrimeStat IV - A Spatial Statistics Program for the Analysis of Crime Incident Locations*. National Institute of Justice, 2013.

[28] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, Aug. 1987. [Online]. Available: http://doi.acm.org/10.1145/37402.37422

[29] G. M. Nielson and B. Hamann, "The asymptotic decider: Resolving the ambiguity in marching cubes," in *Proceedings of the 2Nd Conference on Visualization '91*, ser. VIS '91. Los Alamitos, CA, USA: IEEE Computer Society Press, 1991, pp. 83–91. [Online]. Available: http://dl.acm.org/citation.cfm?id=949607.949621

[30] A. Lopes and K. Brodlie, "Accuracy in contour drawing," in *Eurographics UK*, vol. 98. Citeseer, 1998, pp. 301–311.

[31] J. E. Eck, S. Chainey, J. G. Cameron, M. Leitner, and R. E. Wilson, *Mapping crime: Understanding hotspots*. National Institute of Justice, 2005.