AUTOMATED LABEL PLACEMENT ALGORITHM BASED ON EUROCONTROL'S HMI REQUIREMENTS FOR AIR TRAFFIC CONTROL SYSTEM

Paula Bezerra Rocha Garcia
Atech Negócios em Tecnologias S/A
Rua do Rócio 313, 2º andar, São Paulo/SP, Brasil
Phone: +55 11 3103-4600, E-mail: pgarcia@atech.com.br, Fax: +55 11 3103-4601

Rafael Leme Costa
Atech Negócios em Tecnologias S/A
Rua do Rócio 313, 2º andar, São Paulo/SP, Brasil
Phone: +55 11 3103-4600, E-mail: rlcosta@atech.com.br, Fax: +55 11 3103-4601

Eduardo Nunes Alvares Pereira
Atech Negócios em Tecnologias S/A
Rua do Rócio 313, 2º andar, São Paulo/SP, Brasil
Phone: +55 11 3103-4600, E-mail: epereira@atech.com.br, Fax: +55 11 3103-4601

ABSTRACT

Several algorithms for radar label placement have been developed in order to prevent and to minimize label overlapping automatically on air traffic control (ATC) displays. Automatic label placement is a NP-complete problem, thus its solution depends on an appropriate heuristic. This paper proposes a cluster-based approach in order to determine a local optimal solution by using the obstruction polygon theory, marking weights related to symbols in an occupancy grid and judging conflicts according to Eurocontrol's priority rules. Firstly, the model was validated and then it was implemented in an operational context as a functionality of Atech[1]'s ATC system.

Keywords: radar label anti-overlap, label conflict, label overlapping, label placement, ATC/HMI.

[1] Atech is a company from Embraer's group that develops ATC systems, defense systems and critical technologies solutions.

1. INTRODUCTION

The label-positioning problem has been studied since 1960s in cartography. This field was first concerned with two-dimensional static maps and evolved to three-dimensional dynamic scenarios. Applications for air traffic control, games, virtual reality and augmented reality demand complete and real-time solutions for the label overlap problem.

The air traffic control (ATC) is the focus of this paper. In this context, the controller has the main role as he/she is the responsible for providing maximum safety for airspace users. Besides this, he/she must allow an ordered, fluid and economic air traffic flow, also keeping a harmonious coexistence between civil and military aviation (Cardoso, 2010).

The essential element for the controller to perform his tasks is the Plan View Display, a system that consists of an airspace situation display with dimensions of 2000x2000 pixels. This display shows a two-dimensional scenario, composed by a map in the background and a variety of symbols. Among these symbols, there are the aircrafts' labels, which includes the most important information about the tracks.

The term *"track"* means a radar detection of an aircraft. When the number of tracks increases on the screen and the functionalities of zoom-in/zoom-out are used, these labels can overlap with other labels or with other symbols. This is prejudicial to the intelligibility of the controller on many levels of operation.

Figure 1 shows the label overlapping problem on an actual controller's display. It is a screenshot of the new DA/COM (air defense system) user interface, developed by Atech.

According to Dorbes (2000), a number of algorithms for radar label placement has been developed in order to prevent and to minimize label overlapping automatically. According to Azuma and Furmanski (2003), Peterson *et al.* (2009) and Reek (2010), the label overlapping problem is NP-complete. This means that an optimal solution might not be found and an appropriate heuristic solution needs to be tested.

EUROCONTROL – an international organization composed of Member States from the European Region that is involved in almost every aspect of air traffic management – defined a set of requirements for the radar label placement on user interface. Besides the automatic conflict resolution functionality, or automatic deconfliction, requirements to manually position labels need to be considered (Dorbes, 2000). The transition between auto mode and manual mode must always be available for the controller.

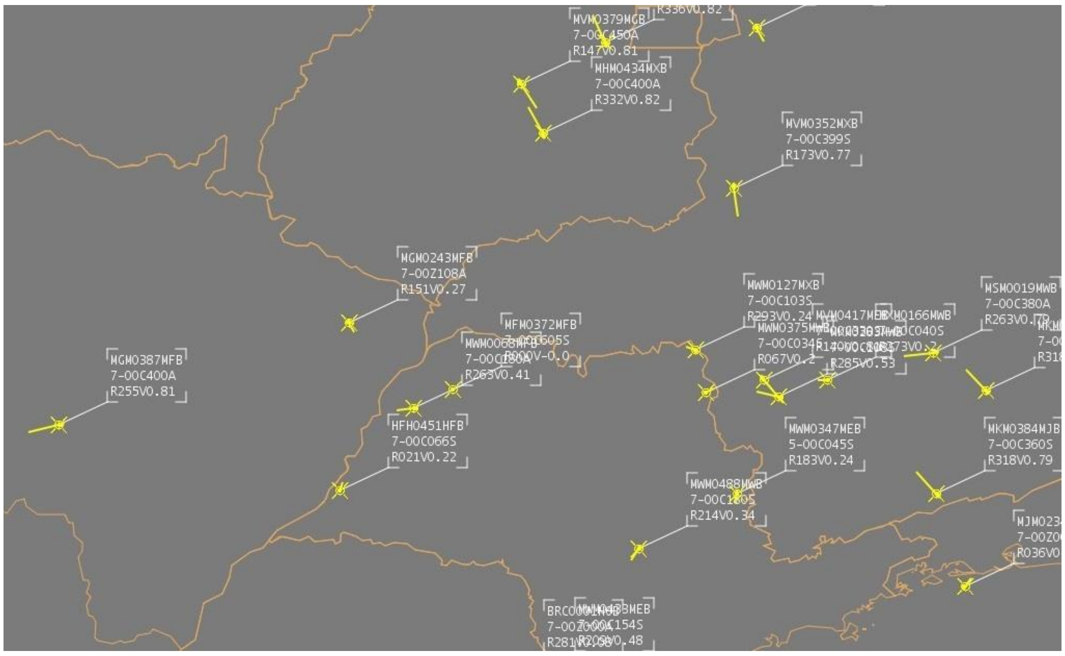

Figure 1- New DA/COM user interface by Atech.

Among the automatic solutions proposed in literature, there are algorithms based on models that explore local optimal solutions and models that seek the global optimal solution. Such solutions can be grouped in four main categories: models based on force (Azuma and Furmanski, 2003), models based on clusters (Duverger, 2005; Azuma and Furmanski, 2003), models based on navigation functions (Kakos and Kyriakopoulos, 2005) and models based on Probabilistic Roadmaps (Reek, 2010).

This paper presents the model adopted by Atech for an automated label placement algorithm, its validation and the results of its implementation and integration on an actual ATC control console.

Specifically, Section 2 is an overview of the problem's context and requirements. Section 3 reports on literature review, presenting four models. Section 4 explains the paper's methodology and proposed solution. Section 5 presents the model's implementation and validation, while Section 6 shows the results of the practical implementation. Section 7 provides a conclusion and outlines future work.

2. OVERVIEW

A representation of the symbols used in the airspace visualization displays, with their respective names, is shown in Figure 2.

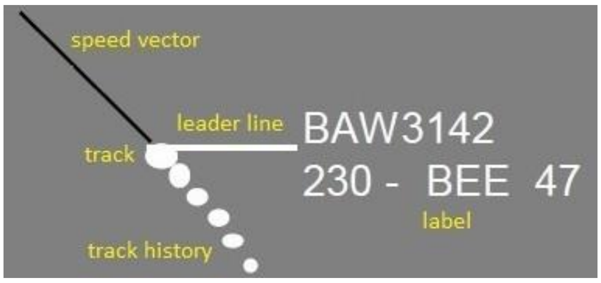

Figure 2 - Symbols on the airspace visualization display (adapted from Dorbes, 2000).

The *track*, represented by the largest circle, is the icon that shows the aircraft's current position. The *history track* is a record of previous radar locations of the aircraft. The *speed vector* represents the aircraft heading and the length of this vector is proportional to its instantaneous velocity. The *label* contains additional information about the aircraft. It has at least the aircraft's *callsign* (ID number), current flight altitude and speed, and a few other flight related information. When a track is selected, the label should appear with a different color. Finally, the *leader line* connects the label to the track and these three symbols move together on the display.

The display can also contain some symbols representing other important objects, such as airways, beacons, etc.

The priority of anti-overlap requirements established by EUROCONTROL for label placement algorithms is described below (Dorbes, 2000):

1. Callsign should never overlap;
2. The label of an aircraft should not overlap with the position symbol/track history of another aircraft;
3. Labels should not overlap;
4. The leader line of a label should not cross the leader line of another aircraft's label;
5. The leader line of a label should not cross the label of another aircraft;
6. The label of an aircraft should not overlap with the speed vector of another aircraft;
7. The leader line of a label should not interfere with the position symbol/track history of that aircraft;
8. The leader line of a label should not interfere with the position symbol/track history of another aircraft;
9. The leader line of a label should not interfere with the speed vector of that aircraft;
10. The leader line of a label should not interfere with the speed vector of another aircraft;
11. If inclusion of other features is needed: the label of an aircraft should not overlap with beacon symbols, airway symbols, DFL, etc.

Besides these requirements, EUROCONTROL recommends optimum and maximum values for label movement distance assignments, leader line length (minimum, maximum and default) and label rest position (90°, 135° – *default*, 225°, 270° from the reference).

3. LITERATURE REVIEW

The four most important models available in literature to the automatic label deconfliction are presented in this section and their advantages and disadvantages are evaluated.

3.1. Force-based model

The general idea is to model objects as electrically charged particles that attract and repel each other. Labels repel other labels and tracks attract labels (ensuring labels will not stray too far from their respective tracks).

According to Azuma and Furmanski (2003), this model has been used in graph visualization, pushing the graph's nodes away from each other to maximize legibility.

3.2. Cluster-based model

A cluster is a group of tracks that has overlapping symbols. The main models discussed in literature that consider the use of clusters in their algorithms are the models described by Duverger (2005) and Azuma and Furmanski (2003). These algorithms focus on the identification of overlapping label clusters and on the resolution of these conflicts locally.

The algorithm developed by Duverger (2005) is called *Mathematical Weighted Formula* (MWF). Reek (2010) explains and adopts this model in his thesis, so the general idea is: the area around the track is discretized in a grid of possible cells to place the labels and each of these cells receives a cost. The label positions are determined every time the track moves on display and the cell with the lowest cost is chosen to locate the label.

Figure 3 illustrates the algorithm. The label to be positioned is the one associated with the central track in the grid. The cells with higher probability of overlap receive higher cost (darker cells) and the cell that should be chosen is the one with the lowest cost among all (lighter cell).

The following properties compose the cost function: overlapping symbols, angle of the leader line, distance between label and track, and penalty for "jitter" caused by label movements. The final formula is an empirical weighted combination of those four costs.

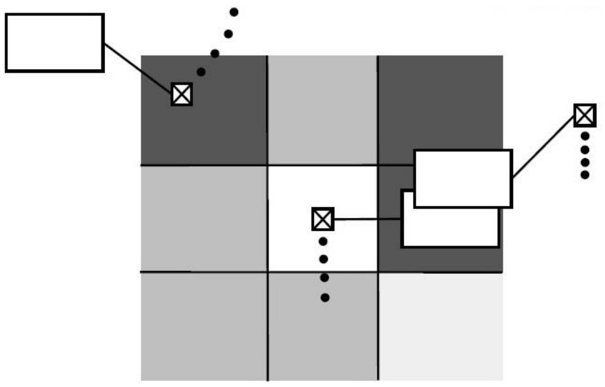
Figure 3 - MWF algorithm: grid around an aircraft with cells of different costs (Reek, 2010).

On the other hand, Azuma and Furmanski's algorithm (2003) identifies clusters and searches for solutions by repositioning all labels in the group simultaneously. The authors say that it avoids local minima, problem that affects other methods that move just a label at a time.

The model is simple: firstly, the algorithm searches for labels in conflict and groups them in clusters; secondly, each cluster is visited and all its labels are moved randomly; finally, the new configuration is compared to the previous positions using a cost function and, after a certain number of iterations, the best configuration is chosen.

Peterson *et al.* (2009) also implement the cluster-based method and emphasize that each cluster is analyzed according to descending size order and the new random positions are limited to a fixed number. Furthermore, the positions of the previous configuration are also included in the evaluation in order to avoid an inferior choice.

Regarding the cost function, Azuma and Furmanski (2003) and Peterson *et al.* (2009) define descending costs for label-label overlap, label-leader line overlap and intersection between two leader lines. The authors use 36 possible radial positions to place the label around a track. Figure 4 shows the results of the algorithm.

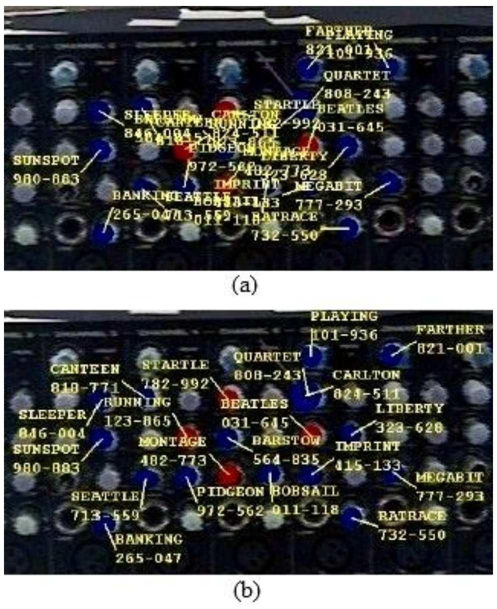

Figure 4 - Cluster-based algorithm by Azuma and Furmanski (2003): (a) initial positions of the labels randomly chosen; (b) new positions for the labels after applying the algorithm.

3.3. Model based on navigation functions

Navigation or motion planning problems in robotics consist of dividing a task into discrete movements that satisfy some constraints and optimize some aspect of the total movement. A common problem is to produce a continuous movement of a robot (movable object) to connect an initial configuration to a goal configuration, avoiding collision between the robot and the obstacles.

The label deconfliction can be interpreted as a problem of this kind. The robot is a label, it is considered punctual and it moves in a two-dimensional scenario. The configuration space is a plan and possible configurations are represented by parameters (x,y). A configuration describes, therefore, a robot's behavior. To avoid label overlap, it is necessary to avoid the robot's collision with tracks, leader lines, speed vectors and other labels.

There are several approaches to solve two-dimensional navigation problems (as grid-based algorithms, geometric algorithms and potential functions). Kakos and Kyriakopoulos (2005) argue in favor of a methodology that considers navigation functions, because they have the characteristic of having only one global minimum (which is defined to be the goal configuration). Rimon and Koditschek (1992) define the navigation functions as a class of artificial potential functions and provide mathematical details about them in their work.

Still on Kakos and Kyriakopoulos' (2005) model, to transform the label into a punctual robot it is necessary to subtract the volume of the label and add it to the volume of the obstacles. Each obstacle is decomposed into three rectangles. They are around the label, around the leader line and around the track symbol, as shown in Figure 5a.

For each rectangle, the Minkowski sum (refer to the appendix) is applied to the center of the robot, the central point within the label, in order to create a punctual movable robot, as in Figures 5b and 5c. The resulting form is then approximated by an ellipsoid to pose the problem as mathematically tractable and describable by a simple function.

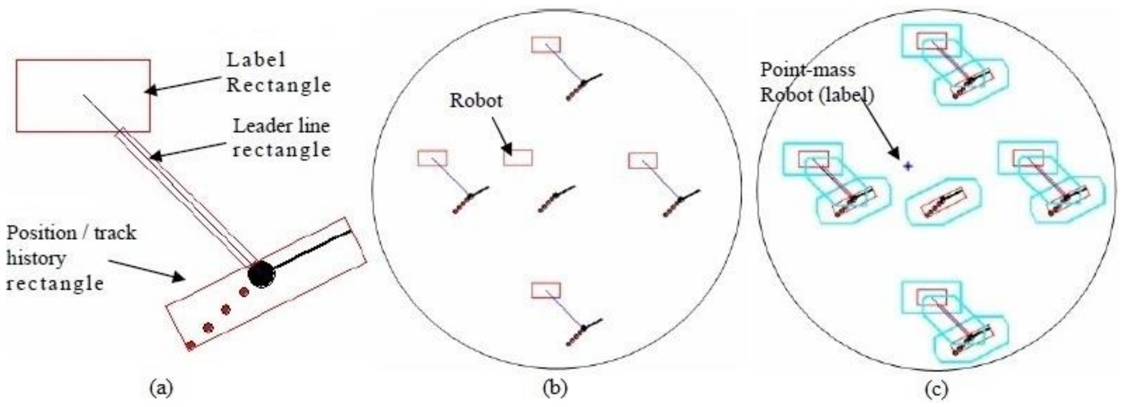

Figure 5 - (a) Decomposition of obstacles in three rectangles; (b) scenario with obstacles, a movable label and its configuration space; (c) Minkowski sum to create a punctual movable robot (Kakos and Kyriakopoulos, 2005)

Additionally, the model applies a coordinate transformation to make the scenario of interest spherical and represents it with a navigation function. Then the robot uses a gradient descent algorithm to find an optimal path to its destination.

3.4. Model based on Probabilistic Roadmap (PRM)

This model was presented by Reek (2010) and tries to obtain a global solution. Each label is treated as an independent robot that tries to find the best path to its goal in a space with obstacles (tracks, leader lines, etc.) and with other robots (labels).

Reek (2010) explains the general idea of PRM: create a graph with subgroups of all possible existing positions. The nodes of the graph represent the positions and the edges indicate a possible transition with its cost. Existing standard short-path algorithms can be applied easily and the chosen path is, therefore, the one with the lowest cost.

The roadmap construction is made by samples of random positions through which the robot can pass and the forbidden positions (for example, inside the obstacles) are discarded. The connections between the graph's nodes are made according to the real possibility of robot's movement and, usually, each node has a threshold of possible connections. Then, the edges costs are based in pre-established requirements, as the maximum distance between the positions, for example.

If a solution is obtained before the available time to complete the search, the graph can be extended with a prediction of future positions for the tracks and then the short path solution is obtained.

3.5. Evaluation of models

The force-based model is one of the oldest solutions to the label overlapping problem. Azuma and Furmanski (2003), who implemented the algorithm, mention that it does not work well to avoid overlaps and often causes distraction to the controller (oscillations are generated on the screen when groups of labels keep attracting or repelling each other repeatedly).

Regarding the cluster-based MWF model (Duverger, 2005), Reek (2010) says the overlaps are largely reduced and the labels movements occur in the same order as if done by human interaction. On the other hand, he calls attention to the problem of a cascade of movements and to an increase in overlaps duration. The first problem is because the implementation searches for a local optimum rather than a global optimum. Therefore, when a track moves, the algorithm searches for a better position to the respective label; it does not take into account the possibility to move other labels around, which could lead to a much better solution. The second problem is caused by the introduction of jitter cost, because there is always a relation between the overlap duration and the number of label movements; the solution is related to this cost fit, but probably only an in-depth study could fine tune the ideal parameters.

Peterson *et al.* (2009) evaluated the next cluster-based model from Azuma and Furmanski (2003). He stated that the algorithm shows good results in avoiding label overlap in real-time applications. On the other hand, the method has a long startup time: Azuma and Furmanski's (2003) results show that the initial label placement performance is poor, but the configurations improve after five or more iterations. Moreover, Peterson *et al.* (2009) guarantee a solution in every time interval, because the algorithm works with a fixed set of random positions to evaluate in each cluster. Obviously, this also means that the algorithm can fail in finding a satisfactory solution; a general label placement is always refined, however.

Concerning the navigation functions approach, it was difficult to find references that discuss or compare the performance of that method with others. However, at Kakos' website[2] a system that allows a collision-free movement using a navigation function methodology is presented. There, one is able to watch an online video with some minutes

[2] Kakos Bros Solution website <http://www.kakos.com.gr/page_1145700674781.html>.

of simulation in a scenario with ten simultaneous tracks and the algorithm presents satisfactory performance.

Reek (2010) also evaluates the model based on PRM and he lists as an advantage the human-like behavior of the algorithm. On the other hand, the problems he finds appear in a large number. The main PRM solution drawback is the difficulty in finding an optimal solution, because the algorithm is slow and the search space is too large. Additionally, the solution is global and all the labels are always taken into account. It implies an intense label movement on the screen, including labels which were not in conflict.

Thus, Reek (2010) concludes that the MWF algorithm is much better than the PRM to solve the label deconfliction. He reasons that: (i) MWF takes less memory and CPU power to complete the calculations; (ii) while PRM tries to solve a NP-hard problem approximately, MWF is a much simpler problem with a greater success rate to determine a good solution.

All these arguments contribute to the adoption of a local solution for the automatic label placement problem. It will guarantee a satisfactory result in each iteration and avoid problems with processing capacity.

4. PROPOSED APPROACH

Our methodology follows three steps: (i) model's construction; (ii) model's validation; and (iii) algorithm's implementation and integration.

The solution to label deconfliction adopted in this paper was based on the two cluster-based models and the approach employing navigation functions; all available in literature and discussed above. The first two models depend on empirical tests to determine the best costs of each cell and the third model provides the idea of classifying the scenario in free regions and in forbidden regions. The obstruction polygon, based in the Minkowski sum, represents one possible tool to implement that classification.

Therefore, to solve or minimize the label overlapping problem, this paper proposes a cluster-based model that uses the theory of obstruction polygon to define forbidden positions to place labels and evaluates costs per cells of a grid to choose the best new configuration. Concepts such as occupancy grid, analysis per quadrants, obstruction polygon and relaxation constitute the model.

The validation phase is a proof of concept developed using MATLAB software. MATLAB is a powerful environment to work with, making matrix computation and numerical calculations easy and allowing fast code debugging.

Finally, after verifying the model, the automated label placement algorithm is implemented in Java and integrated as a functionality within SAGITARIO, which is Atech's system for Air Traffic Control. From MATLAB's proof of concept implementation, the integration in other environments becomes an easier task, because much of the Matlab code can be reused in Java from an already implemented and tested backbone solution.

4.1. Occupancy grid

A grid with same dimensions of the visualization area is used to represent the scenario. This representation allows the construction of alternative settings in a short time (Elfes, 1989). The grid is composed of cells. Each cell of the grid represents a screen point, with (x,y)-coordinates, and is assigned a power-of-two weight value according to the symbol type which is used to mark it on the grid, as shown in Table 1.

Table 1 – Symbol's weights for the occupancy grid

Weight	Symbol type
0	Free cell
1	Track
2	Label
4	Leader line
8	Speed vector
16	Obstruction polygon (forbidden region)
32	Callsign

Labels and tracks, for example, were defined to have the same shape and the same size of their types; labels are always rectangles and tracks are always squares. The points that compose the whole area of the symbols are marked on the occupancy grid.

4.2. Analysis by quadrants

When a label is repositioned, it must remain at this configuration for a period adjusted by the controller in the system. Moreover, when the controller moves a label manually, the deconfliction algorithm also needs to keep the new label position for that same period of time. This feature helps to avoid oscillations on the screen and distractions.

A track is, therefore, classified as movable when it has not been moved since a predefined time by the automatic deconflicter or by the controller, otherwise it is classified as fixed.

The visualization area is divided into four quadrants to set the order in which the movable tracks will be relocated. These quadrants are arranged in descending order with respect to the numbers of movable tracks inside each one of them. When two quadrants contain the same number of track labels, quadrants are arranged according to the usual numbering of the Cartesian plane's quadrants.

4.3. Obstruction polygon

The obstruction polygon (refer to the appendix) was used to define the positions in which a label cannot be placed in order to avoid overlapping with other symbols. So all those positions are called “forbidden region” and they are marked in the occupancy grid.

As labels are always rectangles and tracks are always squares on this model, the obstruction polygons for those objects also do not change. It is sufficient to calculate those polygons only once, to store the obtained results and to copy and shift them to their desired positions. In the case of convex polygons, the obstruction polygon is simply the intercalated reordering of the edges of the two polygons under consideration. The reference point used for the calculation is the upper-left point of the symbols. Figures 6a and 6b show an example of how the obstruction polygon is constructed for the label-label conflict and for the label-track conflict, respectively.

Regarding leader lines and speed vectors, it is not possible to precalculate the obstruction polygons and just reuse them, because straight lines can assume any slope. In that case, the calculations are done on demand. The straight lines are transformed into rectangles and the above-mentioned rule for intercalated edges in convex polygons remains.

4.4. Relaxation

Depending on the situation, there is no solution for repositioning all labels and some overlapping must be allowed. In other words, this algorithm needs to consider certain relaxation for the solution.

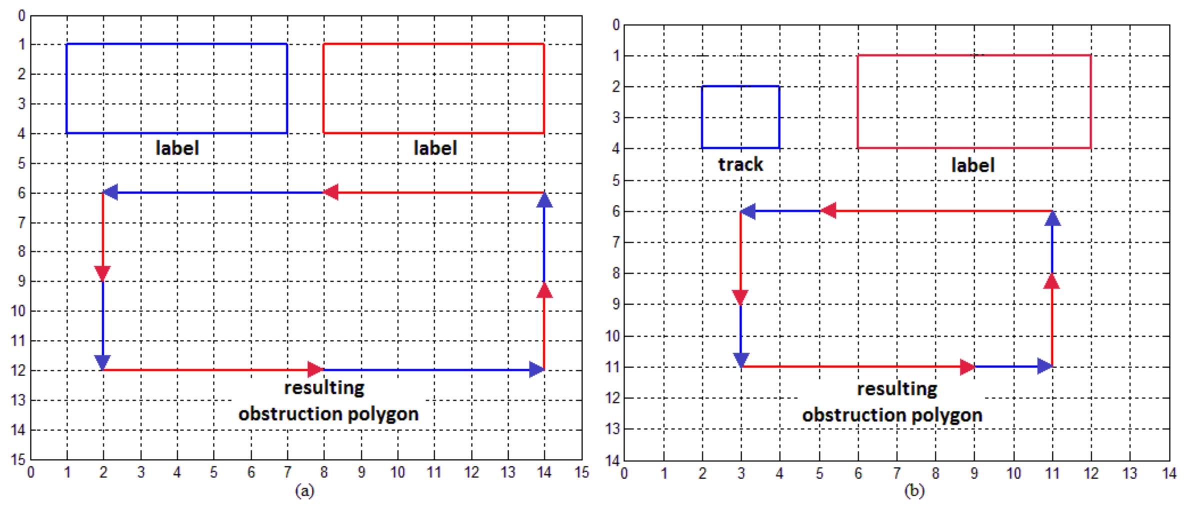

Figure 6 - Construction of obstruction polygon for: (a) label-label case; (b) label-track case.

When a new position for a label in conflict is calculated, as well as the new position of its respective leader line, the logical operation XOR (exclusive OR) is calculated between all the symbols and the occupancy grid. Then the result of that operation is converted to a penalty.

The possible overlaps are always related to a label, to a callsign or to a leader line with another symbol. Table 2 shows all types of possible overlaps, and the related power-of-two weights (described in Table 1) for the XOR operation. However, the XOR operation result is adapted in two cases: (i) when an element is in contact with a free position and (ii) when an element is in contact with an identical element. In the first case, the XOR operation result would be the value of the symbol itself, but the adapted result is defined as zero (because there is no problem in positioning a symbol in a free region). In the second case, the adapted XOR result is the value of the symbol itself (because a zero would indicate a free position erroneously).

The penalties for each overlap depend on a list of requirement proposed by Dorbes (2000). This values definition is empirical and, in principle, the values suggested by Reek (2010) will be used.

The details about the XOR operation, the adapted results and the penalties for all possible combination of symbols are shown in Table 2.

From the penalties described previously in the above table, a score is applied for each placement. This number is composed by the sum of the corresponding penalties for each type of overlap. For example, a new placement overlaps the leader line with the speed vector and the label with another label. The penalty in this case will be:

a. Leader line with speed vector: $4 \oplus 8 = 12$ → penalty = 5;
b. Label with label: $2 \oplus 2 = 0$ → 2 → penalty = 30;
c. Total score = 5 + 30 = 35.

Table 2 - Penalties for all possible combination of symbols

Type of overlap	XOR	Adapted result	Penalty
Callsign + X	$32 \oplus X$	32 – 40 or 48	150
Label + Track	$2 \oplus 1$	3	40
Label + Label	$2 \oplus 2$	2	30
Leader line + Leader line	$4 \oplus 4$	4	25
Leader line + Label	$4 \oplus 2$	6	20
Label + Speed vector	$2 \oplus 8$	10	15
Leader line + track	$4 \oplus 1$	5	10
Leader line + Speed vector	$4 \oplus 8$	12	5
Label + free cell or Leader line + free cell	$0 \oplus X$	0	0

5. MODEL’S IMPLEMENTATION AND VALIDATION

The automated label placement algorithm must receive a certain amount of tracks inside a rectangular visualization area, identify conflicting movable tracks and reposition them to solve (or to reduce) the overlapping problem in the given scenario.

The algorithm’s basic steps are: (i) divide the visualization area in an occupancy grid; (ii) group the movable labels in clusters per quadrant; (iii) map the forbidden regions to place a label using the position information of the fixed symbols (in other words, the obstacles); (iv) mark the forbidden region’s and fixed symbols’ weights in the cells of the occupancy grid; (v) for each movable label, vary leader line length and angle to place the label: if the new position does not cause overlapping, choose that configuration; otherwise, sum the penalties for all the possible configurations and choose the one with the lowest score; (v) go back to step (iii) and pick another movable label to reposition.

The proof of concept to validate the model was developed using MATLAB software (version R2014a), from MathWorks. MATLAB is a software dedicated for numerical computation and it provides a powerful environment for engineering and sciences in general.

Some assumptions were made to guide the solution's implementation. The origin of the Cartesian coordinate frame on the screen is located at the upper-left corner. So the y-axis is directed from top to bottom and x-axis coordinates increase from left to right. Besides, the coordinates are counted with natural numbers because the screen works with pixels.

The track is a circle and its reference point is its center, but on this implementation, it is defined as a square with side measuring twice the radius of the circle. The label is a rectangle and its reference point is the upper-left corner. It is also necessary to have its height and width.

The leader line and the speed vector are straight lines with an arbitrary slope. Both start from the center of the track (initial point). In case of the leader line, it ends at a vertex or at a midpoint of a label edge. In case of the speed vector, it can end at any point depending on the aircraft's speed.

It is also necessary to define a reference orientation to reposition labels (Dorbes, 2000), which can be relative to the speed vector (variable reference) or to the top of the screen/to the North (fixed reference), with positive values in the clockwise direction. Lastly, the user sets a default angle in relation to the defined reference (usually 135º).

The following simulation considers a scenario with 60x60 pixel dimension, 11 fixed tracks (in blue) and 7 movable tracks (in green). The minimum, default and maximum leader line lengths are 3, 10 and 20, respectively, with unitary increments. The label's extended height is twice label's default height. Finally, the default label rest position is 135º.

Figure 7, generated with MATLAB, shows the tracks distribution on the visualization area. The blue tracks are fixed and the green tracks are movable.

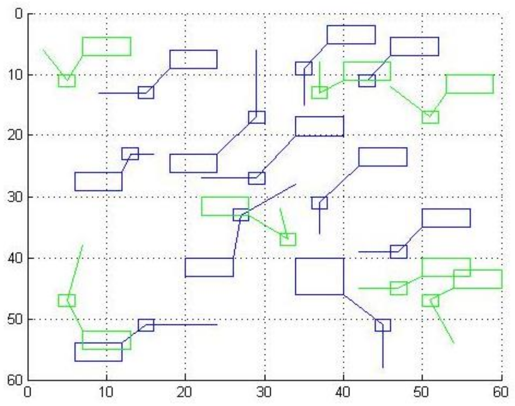

Figure 7 - Scenario for simulation with 11 fixed tracks (blue) and 7 movable tracks (green), generated with MATLAB software.

The repositioning results are shown in Figure 8a for reference relative to the top of the screen and in Figure 8b for reference relative to the speed vector. The repositioned labels belong to the red tracks. One can observe the different positions chosen by the algorithm when the reference changes. The upper-left red track shows this difference clearly: in the first case, it was rotated to the

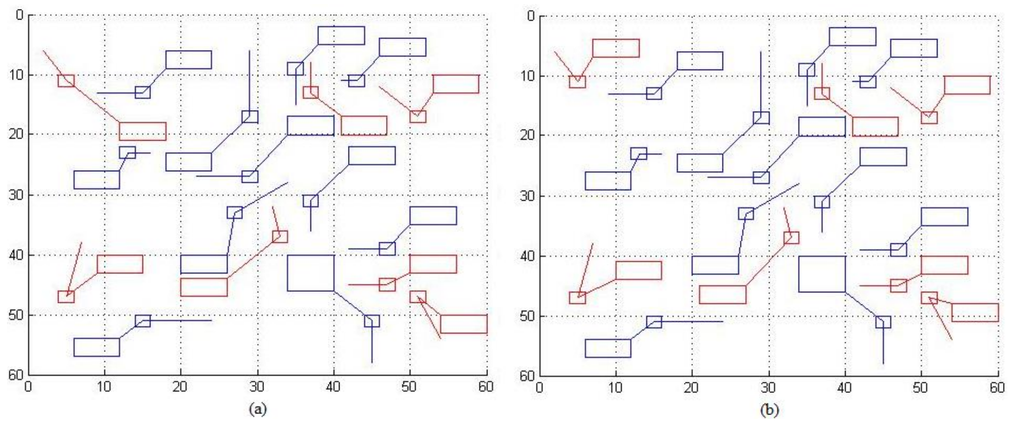

Figure 8 - Results obtained with MATLAB for simulation of scenario with 11 fixed tracks and 7 movable tracks: (a) top of the screen reference, (b) speed vector reference. Blue tracks are fixed and red tracks are new.

default position (angle of 135º and default leader line's length); in the last case, it remained at the same position because it was not in conflict and the default position would provoke overlap.

6. RESULTS

After validating the model, the automated label placement algorithm was implemented as a plugin of SAGITARIO. The algorithm was coded in Java and the label's default rest position was set as 135º related to the top of screen.

To illustrate what happens in the algorithm's background, Figure 9 shows an example of weight markings on the occupancy grid. Because the movable label in Figure 9a does not overlap with another symbol, the algorithm moved it to the default position and the resulting markings are illustrated in Figure 9b. Green colored cells, for example, represent the forbidden region with a weight of 16 for each cell and cells colored with yellow is the label weighted as 2 (see Table 1).

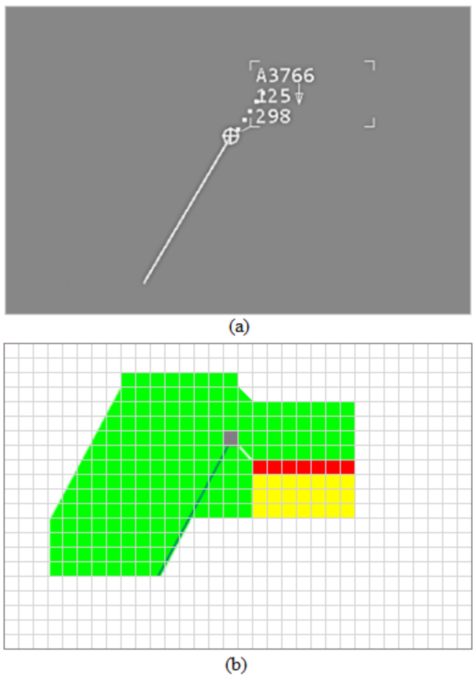

Figure 9 – Occupancy grid: (a) A track with a movable label; (b) Weights on the occupancy grid after repositioning the label (e.g. green is the forbidden region and red is the callsign).

An important requirement is the priority of manual movements over automatic deconflicter solution. Figure 10a shows a blue label that was selected by the controller. He/she dragged the label to a desired position, causing overlap with another label, as shown in Figure 10b. Consequently, the automatic deconflicter kept the controller's decision for the blue label and moved the other to solve overlap, as in Figure 10c.

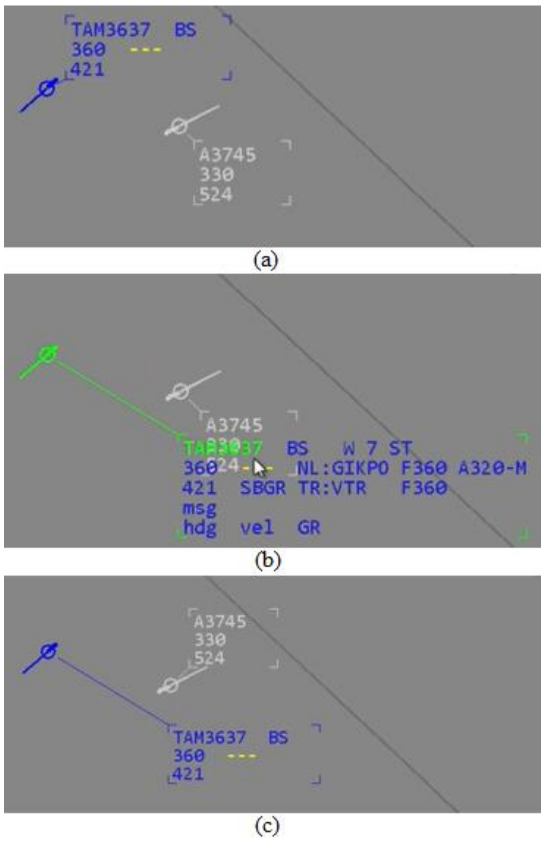

Figure 10 – Label manual placement: (a) The controller selects the label in blue; (b) He/she moves the label to a position that overlaps with the white label; (c) The white label escapes from the blue one automatically and the controller's choice remains unchanged for a configured period of time.

Figure 11a illustrates three labels in conflict, which includes callsign overlap. The algorithm is capable of solving that worst type of overlap successfully: the labels were repositioned respecting their callsign numbers, as shown in Figure 11b.

Expanding the scenario, Figure 12a shows a screenshot with 13 labels and all types of possible overlaps: label with label, leader line, speed vector and track, leader line with speed vector, and callsign. The solution is presented in Figure 12b and shows the algorithm's ability to solve all conflicts.

Finally, a scenario with no symbol overlaps is considered in Figure 13a. In that case, the algorithm should place the labels in the default position (135º related to the top of screen), since the new configuration does not cause overlap. Figure 13b shows the result.

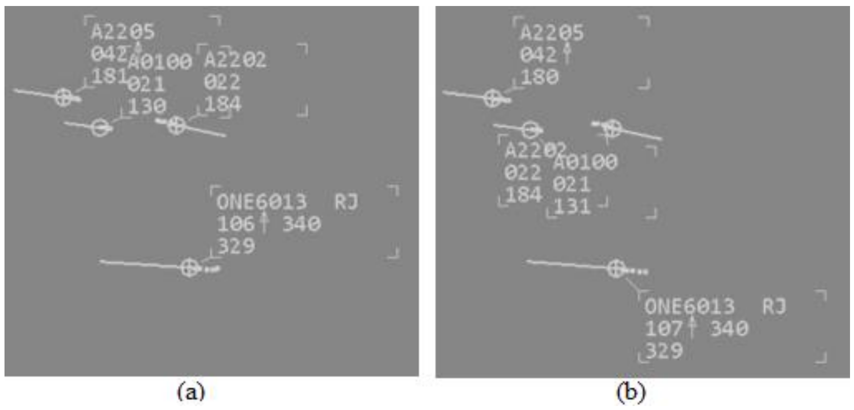

Figure 11 - Callsign overlap: (a) Scenario with three labels in conflict; (b) The automatic deconflicter repositions the labels in a way to avoid callsign overlap (see codes A2202 and A0100).

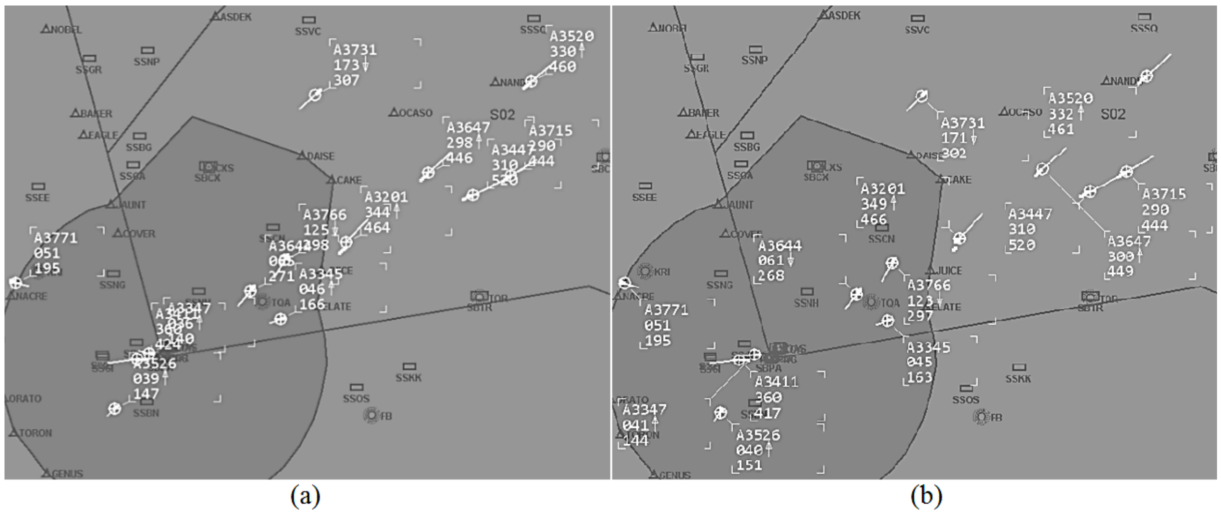

Figure 12 – (a) Scenario with 13 movable labels; (b) Automatic deconflicter's solution.

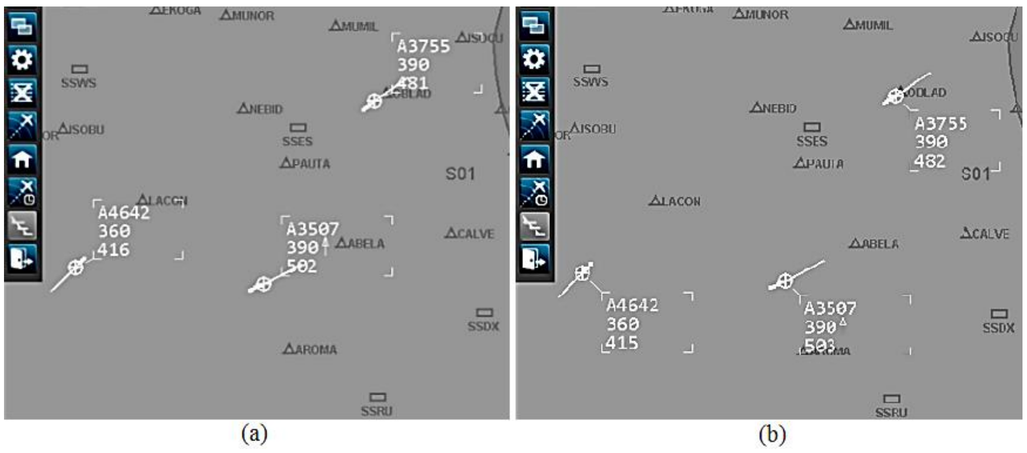

Figure 13 – (a) Scenario with 3 movable labels which are not in conflict; (b) The automatic deconfliction algorithm moves the labels to the default leader line length and label rest position (135º top of screen).

Figure 14 shows the algorithm's runtime as a function of the number of labels. The runtime is not linear and it depends on the number of labels and on the labels' arrangement on the visualization area. The first iteration tends to be longer than the following ones, because the occupancy grid needs to be initialized when the algorithm starts.

The controller usually works with a maximum of 20 tracks under his/her responsibility in a specified airspace's sector. Even when he/she positions the sector of interest in the center of the display, there are other neighboring tracks that also appear on the screen. In an Approach Control Unit (APP) context, there can be about 80 tracks on the screen at the same time and, in an Air Traffic Control Center (ACC), there can be about 160.

From those numbers and from the results presented in Figure 14, it is clear that the proposed algorithm is well suited for practical applications. Even if there were the possibility of a controller to work with a higher number of tracks simultaneously, the implementation would treat all labels and conflicts; however, operationally speaking, this situation is unreal. In Figure 15, it is represented a scenario with about 250 labels and a visualization area four times smaller than the professional 2000x2000 pixels display. Thus, the screenshot is equivalent, in terms of labels density, to a thousand labels in a professional display. One can observe that there is not enough useful area on the screen to reposition all the labels in a way to avoid overlaps.

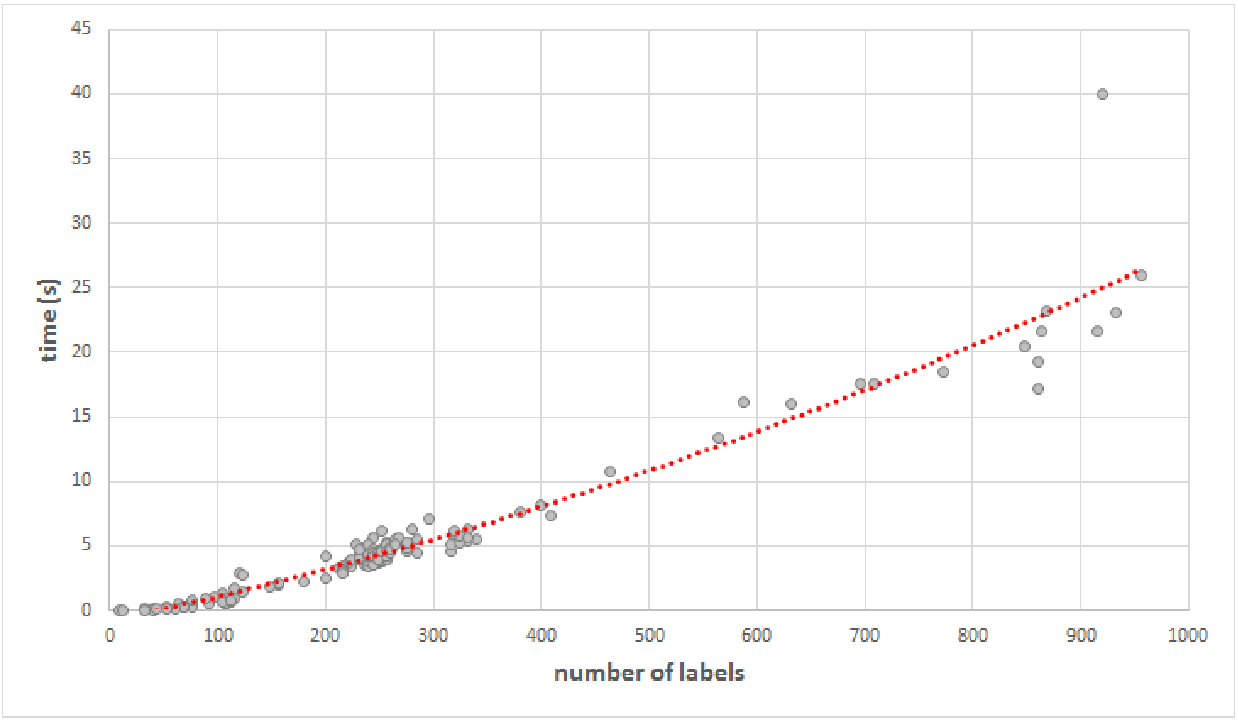

Figure 14 – Algorithm's runtime as a function of the number of labels on the screen.

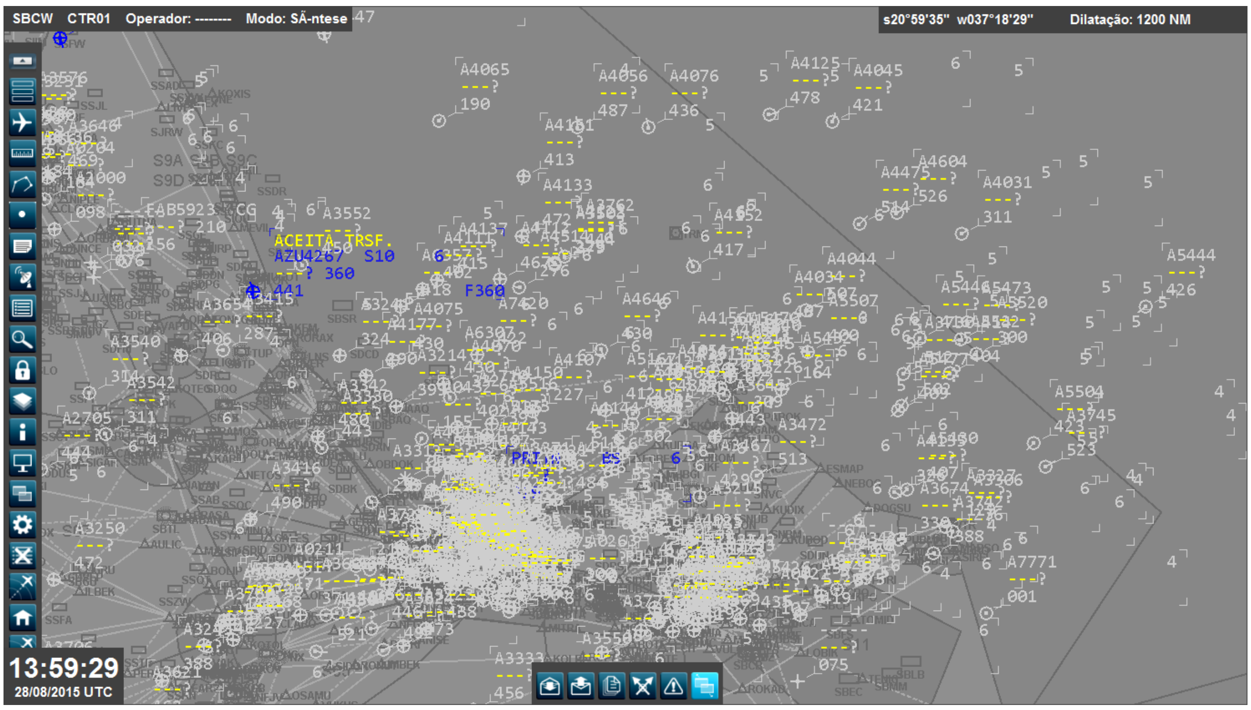

Figure 15 - Scenario with about 250 labels in a visualization area four times smaller in area than the professional 2000x2000 pixels display.

7. CONCLUSION

Since the label deconfliction problem is dynamic, the solution must be in real time and the controller's manual label setting should be maintained as much as possible, a local solution is more appropriate than the global one. This choice avoids problems like processing capacity and repeated oscillations on the screen that can distract the controller.

The algorithm successfully resolved the label-overlapping problem. It was able to identify and to judge all the eight kinds of overlapping between symbols. Because the proposed solution involves a heuristic, the runtime's prediction is not immediate. In the worst case, the algorithm will test all positions (varying the angle and the size of the leader line) and calculate all penalties, choosing the lowest value afterwards. In this situation, the runtime increases with the number of movable tracks. In addition, the total number of tracks on the screen also influences time performance because the occupancy grid needs to be initialized with the symbols and the forbidden regions.

Future directions for continuation of this work could consider the improvement of the algorithm in two aspects: penalty rules and parallel processing.

Regarding penalties, it would be interesting to test other penalty combinations for the symbols to evaluate the algorithm behavior. An idea is to use an exponential scale to analyze the label repositioning effects.

Finally, since the initial setup divides the scenario into four quadrants, a parallel processing of these four regions could help to minimize the algorithm's runtime, although the current response is already appropriate for real applications.

8. REFERENCES

AZUMA, R.; FURMANSKI, C. "Evaluating label placement for augmented reality view management". In: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality, 2003, 10 p.

CARDOSO, R. B. Dia internacional do controlador de tráfego aéreo. Available at: <http://www.decea.gov.br/?p=3905>. Published on 20 Oct. 2010. Accessed on 24 Sep. 2014.

DORBES, A. "Requirements for the implementation of automatic and manual label anti-overlap functions". EUROCONTROL Experimental Center, EEC Note n. 21/00, 2000.

DUVERGER, A. "Development of a mathematical weighted formula to eliminate the overlapping of aircraft labels on the ATC radar display". EUROCONTROL Experimental Center, EEC Note n. 19/05, 2005.

ELFES, A. “Using occupancy grids for mobile robot perception and navigation”. IEEE Computer, v.22, n.6, p.46-57, 1989.

KAKOS, S.; KYRIAKOPOULOS, K. J. “The navigation functions approach for the label anti-overlapping problem”. EUROCONTROL Experimental Centre innovative research activity report 2005. p. 307-319.

PETERSON, S. D. AXHOLT, M. COOPER, M.; ELLIS, S.R. “Visual clutter management in augmented reality: Effects of three label separation methods on spatial judgments”. In: IEEE SYMPOSIUM ON 3D USER INTERFACES. Lafayette, Louisiana, USA, 2009, p. 111-118.

REEK, S. “Real-Time Label Overlap Avoidance for Air Traffic Controllers using Probabilistic Roadmaps”. Dissertation (Master’s Degree) - Utretch, 76 p., 2010.

RIMON, E.; KODITSCHEK, D. E. “Exact robot navigation using artificial potential functions”. IEEE Transactions on Robotics and Automation, v.8, n.5, p.501-518, 1992.

SATO, A.K. “Proposta de algoritmo para a determinação da região livre de colisão e sua aplicação na solução de leiautes bidimensionais irregulares com recozimento simulado”. São Paulo, 89 p., 2011. Dissertation (Master’s Degree) – Escola Politécnica da Universidade de São Paulo.

APPENDIX: OBSTRUCTION POLYGON USING MINKOWSKI SUM

The obstruction polygon corresponds to translations applied to elements, which are matematically represented by a set of vectors. A reference point must be defined and it can be internal or external to the element, as shown in Figure 16 (Sato, 2011).

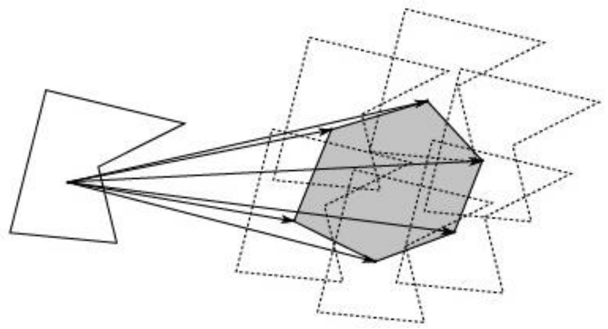

Figure 16 – Translations applied to an item with a reference point (Sato, 2011).

According to this operation, consider two elements positioned in space, a fixed P_i and a movable P_j. There are translations that overlap the movable one to the fixed one. The obstruction polygon is the region that represents the set of forbidden translations for movable element, ie., positions where this element intersects with the fixed one (Sato, 2011).

Figure 17 shows an example of an obstruction polygon.

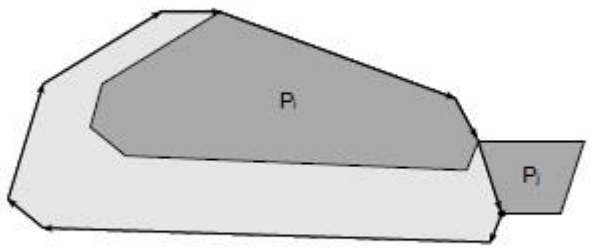

Figure 17 – Obstruction polygon: boundary defined by oriented edges (SATO, 2011).

The Minkowski sum between two polygons is an algorithm for creating the obstruction polygon and it is defined as the set of points:

$$\{O+\vec{v}+\vec{w} \mid O+\vec{v} \in P_i, O+\vec{w} \in P_j\}. \quad (1)$$