Ex-Ante Assessment of Struck-by Safety Hazards in Construction Projects: A Motion-Planning Approach

Md Mahbubur Rahman, Triana Carmenate, Leonardo Bobadilla, Ali Mostafavi

***Abstract*— Struck-by accidents are one of the major causes of fatalities and injuries in construction projects. While the likelihood of struck-by hazards is significantly affected by the layout of a jobsite, sequence of activities, and movement patterns of equipment and workers, there is currently no existing methodology for ex-ante investigation of struck-by safety hazards during the construction *planning phase*. In this paper, we propose a preliminary methodology for evaluation of struck-by safety hazards using a motion planning approach. We solve the problem of finding collision states and obstacle free paths for working in the middle of moving machinery and in the presence of various obstacles. We present, propose, and implement efficient algorithms to create safe routes for workers that avoid *static* and *moving* obstacles. We present a detailed case study of a common construction task involving struck-by hazards. We discuss several avenues for future work that could transform into robust methodologies for safety planning in complex construction projects.**

I. Introduction

Struck-by accidents are one of the four most deadly hazards found on construction jobsites. Approximately 75% of struck-by fatalities involve heavy equipment such as trucks or cranes [1]. The complex, dynamic, and continuously changing nature of construction jobsites is one of the main drivers of struck-by hazards. Different research studies [8] have evaluated the nature of struck-by accidents in construction jobsites and have proposed solution concepts (e.g., radio frequency proximity warning system proposed in [20]) to prevent accidents. A key missing element is a methodology that enables predictive assessment of struck-by hazards based on an integrated evaluation of the construction site layout, movement patterns of workers and equipment, and the sequence of construction activities. The objective of this paper is to propose a vision for such a methodology based on motion planning algorithms.

Our working hypothesis in creating a methodology is that struck-by accidents can be partially predicted through efficient, robust, easily implementable algorithms.

Two coupled phenomena affect the level of safety hazards related to struck-by accidents in construction jobsites: (1) sequence of activities and jobsite layout, and (2) movement patterns of workers and equipment. In our previous work [3], we presented an initial formalization of the physical state space of construction sites along with simple filtering algorithms and prototypes of inexpensive sensing systems for real-time monitoring of struck by accidents in construction sites. Ex-ante evaluation of construction jobsites in terms of struck-by safety hazards is critical in enhancing jobsite layout and activity planning. Different studies such as ([18]) have developed optimization-based methodologies for safety assessment of construction site layouts. These studies have two main limitations: (1) lack of consideration of the impact of the layout of construction jobsites on the spatio-temporal motion trajectories related to the workers and equipment, and (2) lack of consideration related to the dynamic changes in the layout of construction sites at different stages of a project schedule.

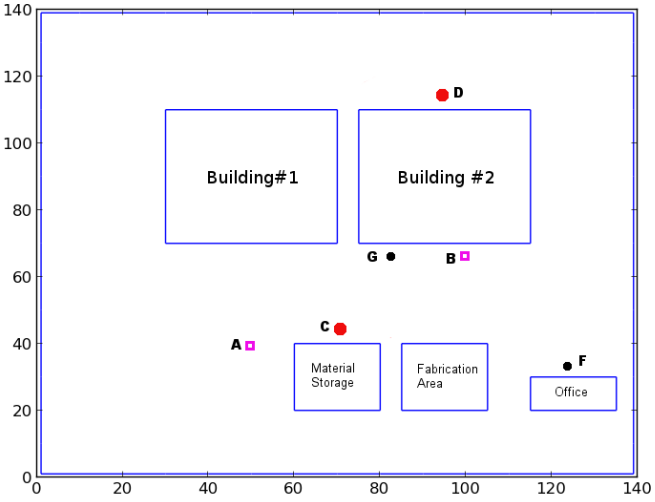

Fig. 1. An example of a construction site

Different studies (e.g. [17], [10]) have developed models based on discrete event simulation to analyze the dynamics of construction jobsites. These models were mainly developed for cost and schedule analysis and do not capture the safety aspects due to the interactions between moving crew and equipment.

Zhang et al. [21] and Hammad and Zhang [7] [22] proposed an approach for real-time motion planning and demonstrated its application in safety analysis related to crane operations. The proposed approach used a cell-based discrete event simulation model to capture the spatial interdependencies between resources. The limitation of this approach is that the physical state space of construction site and motion trajectories of the crew and the equipment are not modeled; therefore, the collision states as a result of the movement of workers and equipment cannot be captured.

M. Rahman, T. Carmenate, and L. Bobadilla are with the School of Computing and Information Sciences, Florida International University, Miami, FL, 33199, USA. A. Mostafavi is with the OHL School of Construction, Florida International University Miami, FL, 33199, USA. {mrahm025,tcarm002}@fiu.edu, bobadilla@cs.fiu.edu, almostaf@fiu.edu

Instead our approach to preemptively investigate collisions in construction projects uses techniques from *motion planning with dynamic obstacles*. Several motion planning algorithms have been proposed to solve the problem of a robot moving in an environment with moving obstacles. These approaches include *sampling-based* and *combinatorial* methods [4], [12], [13], [14], [15].

Our contribution includes a concrete formulation based on physical state spaces of the problem of finding obstacle free paths during the planning phase of a construction project. We propose a procedure, using a modified version of a well known motion planning algorithm [11], to calculate obstacle free paths with moving equipment.

II. Preliminaries

To follow Motion Planning formulations [4], [12], [13], [14], [15], we will define the construction site as a workspace, $\mathcal{W} = \mathbb{R}^2$. In this workspace there will be a collection of *static obstacles*, $\mathcal{O} \subset \mathcal{W}$, where each element in $\mathcal{O} = \{O_1, O_2, \cdots, O_n\}$ will be modeled as a polygon. These static obstacles represent places in the construction site that are obstructed for both workers and machinery.

Let us call $T \subset \mathbb{R}_{\geq 0}$ the time interval in which the construction project takes place. In our examples, the time interval will be a bounded set, $T = [0, t_f)$ where t_f represents the final time.

There will be a collection of k moving bodies, $\mathcal{B} = \{B^1, B^2, \cdots, B^k\}$, that represent the moving equipment in the workspace such as trucks, excavators, mixers and cranes. Each body will move with constant speed represented by a vector, $\mathcal{V} = \{v^1, v^2, \cdots, v^k\}$. We will model the moving bodies as circular regions that can move in any direction. A particular body, B^i, is represented by its center, (B^i_x, B^i_y), and its radius, r. $\mathcal{B}(t)$ denotes the position of the moving bodies at time, t.

There will be m workers $\mathcal{A}^1, \mathcal{A}^2, \cdots, \mathcal{A}^m$ present in the workspace. Each worker, $\mathcal{A}^j$, has an associated *configuration space*, $\mathcal{C}^j$, which denotes the position in the workspace. The configuration space for all workers is defined by $\mathcal{C}_{workers} = \mathcal{C}^1 \times \mathcal{C}^2 \cdots \times \mathcal{C}^m$. Let $\mathcal{A}(q)$ with $q \in \mathcal{C}_{workers}$ denote the positions of all the workers in $\mathcal{W}$. The workers' *physical state space* is defined as $X = \mathcal{C} \times T$. A state, $x \in X$, is expressed as $x = (q, t)$ and denotes the configuration of the bodies, q, at time, t.

We define the *obstacle region*, X_{obs}, in the *physical state space* as:

$$X_{obs} = \{(q, t) \in X | \mathcal{A}(q) \cap (\mathcal{B}(t) \cup \mathcal{O}) \neq \emptyset\}. \quad (1)$$

The moving bodies move in an obstacle free path, therefore, $\mathcal{O} \cap \mathcal{B}(t) = \emptyset$. The free space for the workers is denoted by $X_{free} = X \setminus X_{obs}$, which means the workers must avoid both the static obstacles and the moving bodies.

Let us denote the trajectory of the i^{th} moving body as $\gamma^i : [0, t_f] \rightarrow \mathcal{C}^{body}_{free}$ where $\mathcal{C}^{body}_{free}$ are the obstacle free configurations for the bodies. We model the trajectories of the bodies as piecewise linear functions of time, which means that the translation applied to bodies in $\mathcal{B}$ can be written as $(B^i_x + c_1 t, B^i_y + c_2 t)$ for some constants, $c_1, c_2 \in \mathbb{R}$.

Let $x_I \in X_{free}$ be the initial state of the worker and $x_G \subset X_{free}$ be the goal states. Let a trajectory of the j^{th} worker, $\mathcal{A}^j$, be $\tau_i : [0, 1] \rightarrow X_{free}$.

Based on our definitions we can describe the following two problems:

Problem 1: Finding Safe Trajectories for Workers
Given an initial configuration, x_I, a goal region ,x_G, the set of static obstacles, $\mathcal{O}$, and the motions of the moving obstacles in time, $\mathcal{B}(t)$, find an obstacle free trajectory, τ, such that $\tau(0) = x_I$ $\tau(1) \in X_{goal}$ that is as far away as possible from the static obstacles.

In some other scenarios, we would like to evaluate how safe a given trajectory is, which leads to Problem 2:

Problem 2: Safety Evaluation of a Trajectory
Given the set of static obstacles, $\mathcal{O}$, the motions of the moving obstacles in time, $\mathcal{B}(t)$, and a trajectory, τ, evaluate the trajectory's safety.

In the following section, we present our approach to solving problems 1 and 2.

III. Methods

A. Calculation of a safe roadmap for workers using a generalized Voronoi diagram

In this section, we will use the *generalized Voronoi diagram* (GVD) or *maximum-clearance roadmap* [16], [13], [19] to compute safe trajectories for workers. The generalized Voronoi diagram was chosen because it is a roadmap whose paths are as far away as possible from static obstacles(see Figure 2).

Recall that the set of static obstacles is given by $\mathcal{O} = \{O_1, O_2, \cdots, O_n\}$. We assume that the obstacles are *convex polygons*. If the obstacles are not convex, they can be approximated by surrounding them with a convex shape. For any point, $p \in \mathcal{W}$, let $d(p, O_i)$ denote the Euclidean distance from p to the nearest point in the obstacle O_i [9]. The midpoint or bisector between the two obstacles, O_i and O_j, is $b(O_i, O_j) = \{p | d(p, O_i) = d(p, O_j)\}$. We also define the dominance region of obstacle O_i over O_j as $Dom(O_i, O_j) = \{p | d(p, O_i) \leq d(p, O_j)\}$. The Voronoi region for the obstacle O_i is $V(O_i) = \bigcap_{i \neq j} Dom(O_i, O_j)$. The partition of space into $V(O_1), V(O_2), \cdots, V(O_n)$ is the generalized Voronoi diagram [9].

Algorithm 1 presents the pseudo-code of implemented procedure to find all possible safe paths using the algorithm for the Voronoi diagram. There are several exact procedures to calculate the GVD for a set of convex obstacles [16]. However, some of these approaches have complicated implementations, therefore, we use a solution based on calculating the Voronoi diagram of a set of points as discussed in [2].

The obstacle set, $\mathcal{O}$, contains both the static obstacles and the boundary region. In lines 1 and 2 of Algorithm 1, we obtain a set of points, P, containing the midpoints of all the polygonal obstacles and sample points from the boundary region. We apply any standard Voronoi diagram algorithm [6]

in line 3 ($GetVoronoi$) to P to generate the Voronoi diagram. Let L be the set of Voronoi edges.

Once the Voronoi edges are found, we remove the edges that pass through the obstacles. This can be done by checking if the voronoi edges and the obstacle edges intersect. Removing all the Voronoi edges (line 6) that intersect with the $obstacle\ line$ segments result in a set of line segments that approximate the generalized Voronoi diagram.

Algorithm 1 CalculateAllPaths($\mathcal{O}$,q_I,q_G)

Input: $\mathcal{O}$ {A set of static obstacles}
Output: $\mathcal{S}$ $[1..m]$ {A Matrix where each row is a list of line segments}

```
1:  P ← O.getEdges().midPoints
2:  P.Append(Lines_boundary.GetSamplePoints())
3:  L ← GetVoronoi(P)
4:  for l ∈ L do
5:      for l' ∈ O.getEdges() do
6:          if l.Intersect(l') == True then
7:              L.Remove(l)
8:          else
                continue
9:          end if
10:     end for
11: end for
12: G(V, E) = G(L.EndPoints, L)
13: E.Add(q_I, Nearest(E, q_I))
14: E.Add(q_G, Nearest(E, q_G))
15: S = GenAllPath(q_I, q_G, G)
```

Finally, we construct a weighted undirected graph, $G = (V, E)$, with weights given by $w : E \to \mathbb{R}_{\geq 0}$. In this graph, V is the set of vertices of the Voronoi diagram and an edge, E, is added for each Voronoi edge. The weight, $w(e)$, for $e \in E$ is given by the Euclidean distance between the vertices that compose the edge, e.

Let $x_I = (q_I, t_I)$ be the initial configuration of the worker and the goal configuration be $x_G = (q_G, t_G)$. q_I and q_G denote the initial and final point location on $\mathcal{W}$ respectively. We need to connect these two points on the roadmap given by the generalized Voronoi diagram. In Line 13 of Algoritm 1, this is achieved by drawing a perpendicular line from q_I to the nearest Voronoi line, $e \in E$. This introduces a new point on e which is the intersection of e and the normal line of q_I on e. This procedure is also applied to q_G. After connecting q_I and q_G to the graph, G, the number of vertices and edges of G are $|V| + 2$ and $|E| + 2$ respectively.

When calculating the roadmap for workers, there are several trade-offs and optimality criteria. For example, workers want to choose the fastest path but managers are concerned about the safety of the paths. Ideally, we would like to define the safest path without colliding with the moving bodies $\mathcal{B}(t)$. These will require us to consider all possible paths in graph G from q_I to q_G. The method $GenAllPath(q_I, q_G, G)$ in Line 15 generates all possible path using a variation of Breadth First Search (BFS) [5]. To find a shortest roadmap path from q_I to q_G we use the well known *Dijkstra's Shortest Path Algorithm* [2] on graph G. A path, τ, is composed of piecewise linear segments $\{l_1, l_2, \cdots, l_G\}$ where each l_k is a Voronoi edge.

B. Calculation of safetest roadmap avoiding moving bodies

The worker $\mathcal{A}^j$ must move along his path from $\tau(0)$ to $\tau(t_f)$ while the equipment $B^i(t)$ must move along its path over the time interval $T = [0, t_f]$. Additionally, we enforce two aspects of the workers' trajectories: 1) the speed of the workers is bounded and fixed and 2) workers will only use two actions for navigation: *STOP* and *MOVE*. The second aspect is motivated because, unlike mobile robots, workers will have difficulty precisely controlling their speed. Therefore, to avoid collision with moving bodies on a trajectory a worker must yield and let the moving equipment pass.

We will modify the *velocity tuning method* (see [11] and [13], Chapter 7 for details) to obtain a *plan* for the workers with a fixed speed, v_{worker}, and two actions, $STOP$ and $MOVE$. Let $U = \{STOP, MOVE\}$ the two allowable actions. We call a *plan* a mapping, $\pi : T \to U$.

Algorithm 2 presents our modified procedure to create a plan for the worker. Initially, at $t = 0$ both the worker and the moving body start at the initial point of their respective trajectory τ and γ_i. Moving bodies at different times in $T = [t_0, t_f)$ occupy different spaces on the worker's trajectory, τ. The solution to the problem of avoiding moving bodies lies in a *space-time* coordinate system. Let $S = [0, |\tau|]$, where $|\tau|$ is the length of the trajectory, τ. We define the time-space as $Y = S \times T$ in which each (s, t) indicates the position along the path, $s \in S$, and time, $t \in T$ [11], [13]. The space occupied by the moving body on the workers' path (obstacles in Y) can be calculated in this space-time coordinate system. The plan can be obtained by intersecting the swept area of the bodies with the given trajectory, τ, of the worker [11]. This area is called the *forbidden region*. Let a worker in a piecewise path segment, $l \in \tau$, be characterized parametrically by $\frac{x-x_0}{a} = \frac{y-y_0}{b} = \eta;\ \eta \in [0, 1]$.

We assume, without loss of generality, that the position of the circular moving body at $t = 0$ is (B^i_x, B^i_y), its velocity is $v^i = (v^i_x, v^i_y)$, and its radius is, r. Then, the equation of the area covered by the circular obstacle, as a function of time, is given by [11]: $[x-(B^i_x+v^i_x t)]^2+[y-(B^i_y+v^i_y t)]^2 = r^2$. Substituting x, y from the parametric line equation of the worker we get, $[x_0-B^i_x+a\eta-v^i_x t]^2+[y_0-B^i_y+a\eta-v^i_y t]^2 = r^2$. Let $\Delta x = x_0 - B^i_x$ and $\Delta y = y_0 - B^i_y$. The above equation then is, $\eta^2(a^2 + b^2) - 2(av^i_x + bv_y) + t^2((v^i_x)^2) + (v^i_y)^2)\eta(a\Delta x + b\Delta y) - 2t(v^i_x\Delta x + v^i_y\Delta y) + (\Delta x^2 + \Delta y^2 - r^2) = 0$. This equation [11] is of the form of a general conic equation, $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$, and its discriminant is $B^2 - 4AC = 4(av^i_x + bv^i_y)^2 - 4(a^2 + b^2)((v^i_x)^2 + (v^i_y)^2) = -4(av^i_y - bv^i_x)^2$, which is always negative, therefore the resulting conic after the area swept across the path by the moving equipment represented as a circle is an ellipse. The height of the ellipse indicates the amount of time the moving body is present on the trajectory of a worker. The width is the occupied space which is inaccessible by the worker during that time. We get an ellipse for each intersection between the moving body and the worker [13], [11]. .

The obstacle region in the space-time system is $X_{obs} = \{(s,t) \in X | \mathcal{A}(\tau(s)) \cap \mathcal{B}(t) \neq \emptyset\}$. X_{free} is defined as $X_{free} = X \backslash X_{obs}$. The task is to find the plan $\pi : [0,1] \rightarrow X_{free}$. For simplicity of path planning, we consider the ellipses to be quadrilaterals.

Algorithm 2 CalcVelocityProfile($v_{worker}, \mathcal{V}, \tau, \gamma^i$)

Input: $v_{worker}, \mathcal{V}, \tau, \gamma^i$ {v_{worker} : velocity of worker;$\mathcal{V}$: velocity of moving body;τ : path of worker;γ^i :path of body}

Output: $List, e$ {$List$:Line list in $s \times t$ space. e:Obstacle list.}

1: **for** $l \in \tau$ **do**
2: **for** $\sigma \in \gamma^i$ **do**
3: $obs_{s \times t} = FindObs(l, \sigma, v_{worker}, v^i, w_t, w_s, B^i_x, B^i_y)$
4: $e.Add(obs_{s \times t})$
5: **end for**
6: **for** $b \in e$ **do**
7: **if** $intersect(b.left, l)$ *or* $intersect(b.bottom, l)$ **then**
8: $(x, y) = intersection(b, l)$
9: $List.Add(Point(w_s, w_t), Point(x, y))$
10: $List.Add(Point(x, y), Point(x, b.upperLeftY))$
11: $List.Add(Point(x, b.upperLeftY),$
$Point(w_s, w_t + time(x, y, upperLeftY)))$
12: **end if**
13: **end for**
14: $w_s = w_s + l.length$
15: $w_t = w_t + \frac{l.length}{v_{worker}}$
16: **for** $\sigma \in \gamma^i$ **do**
17: **if** $\frac{\sigma.length}{v^i} \geq w_t$ **then**
18: $\sigma_{current} = \sigma$
19: **end if**
20: **end for**
21: $B^i_x = \sigma_{current}.x + v^i \times w_t$
22: $B^i_y = \sigma_{current}.y + v^i \times w_t$
23: $\gamma^i_{start} = \sigma_{current}$
24: **end for**

Once all the obstacle regions in the space-time system are calculated, we need to find a path avoiding all the space-time obstacles. This can be solved by using a combinatorial planning algorithm such as the *trapezoidal decomposition* algorithm which is commonly used for 2D path planning problems (see [6], Chapter 6 and [13], Chapter 7 for details). The worker starts in $(0,0)$ and moves along a line having a slope, $m = \frac{dt}{ds}$ and $m = \frac{1}{v_{worker}}$ (see Figure 3). Whenever the line intersects an obstacle region, the worker stops and the path π will go up vertically which means the time is increasing but the worker does not move ($\frac{ds}{dt} = \frac{0}{dt} = 0$). The line continues until it reaches the upper left corner of the obstacle which means that the obstacle is cleared and the worker is allowed to move (See Figure 3). This process breaks the workers' path segment into three sub segments every time an obstacle is faced in state space. If no moving body is faced by the worker, then the number of segments in space-time will be equal to the number of line segments in τ.

In the workspace, both the worker and moving body paths are piecewise linear. Recall that we denote the body position along γ^i in $\mathcal{W}$ as (B^i_x, B^i_y). Initially, the worker is in the $(0,0)$ position in $S \times T$ space denoted by $(w_s, w_t) = (0,0)$. In lines 1 and 2 of Algorithm 2, for each line segment of worker $l \in \tau$ we need to find out the intersections of all line segments, $\sigma \in \gamma^k$, of a moving body, B^k.

The method $FindObs$ in line 3 estimates the location, height, and width of the ellipse shaped obstacle in $S \times T$ which might be encountered by the worker. As the workers' configuration change is time monotonic, we must know how much time it takes to finish a path's linear segment before continuing to the next linear segment of the path τ. Lines 6 to 15 perform the velocity profile calculation for the current segment, l. In $S \times T$ space the path segment l starts at (w_s, w_t) with slope $\frac{dt}{ds}$. The segment can hit an obstacle from the left or from the bottom.

In line 7, we check for collisions. If any collision is found then the line is broken down into three pieces surrounding the left edge of the obstacle. The finishing point of current line l is the starting point of next line. So in line 14, (w_s, w_t) is updated. w_s is increased by the length of l. w_t, the time taken to finish the path, is increased by the time taken to complete the line avoiding the obstacles. However, at time w_t the moving obstacle has also been moved to a new position on γ^i. Before processing the next worker path segment, we need to know the position of the moving body and which sub-segments of γ^i will compose it to estimate the intersections between those sub segments and next worker path segment.

In lines 16 to 19, we search for the line $\sigma_{current} \in \gamma^i$ on which the moving body will be after time w_t. Then, in line 21 and 22 we update the exact position of the moving obstacle, (B^i_x, B^i_y), on this line. The process will finish when the worker reaches the goal configuration, x_G. In a space-time system, the finishing point is (s,t), where $s = |S|$ is the length of the trajectory, τ, and t denotes the time taken to finish the path after avoiding all the moving bodies.

IV. Results

A. Case Study : Excavation Activity

We will be using a $140ft \times 140ft$ construction jobsite as depicted in Figure 1. An excavation task is going on in the area labeled as *Building #2*. There are five static obstacles: *Building #1, Building #2, Material Storage, Fabrication Area and Office*. A dump truck has a trajectory from $A \rightarrow B, B \rightarrow A$. The material supply workers' path is from $D \rightarrow C, C \rightarrow D$. Inspection workers' will follow the path $F \rightarrow G, G \rightarrow F$.

We tested our model on a simulated environment developed using the Python Programming Language to calculate a safe roadmap for the workers. A generalized Voronoi diagram generated by the Algorithm 1 of the construction site is shown in Figure 2. The red line is the shortest path for the material supply worker between the points C *and* D, following the safe Voronoi edges. The shortest trajectory derived for the worker is, $\tau = [[71.00, 42.00, 82.50, 42.00], [82.50, 42.00, 82.50, 55.00], [82.50, 55.00, 72.50, 63.34], [72.50, 63.34, 72.50, 115.88], [72.50, 115.88, 80.00, 121.06], [80.00, 121.06, 90.00, 124.50],$

$[90.00, 124.50, 95.00, 124.50], [95.00, 124.50, 95.00, 115.00]]$. Among the four attributes in each tuple, the first two attributes denote the coordinate of one endpoint and the last two attributes denote the other endpoint of a line segment. The dump truck is moving back and forth at a speed of $1.8\ unit/s$ across the path, $\gamma^i = [[50, 40, 100, 65], [50, 40, 100, 65]]$. Let the radius of the area occupied by the truck be $15\ units$. We need to know when the dump truck is going to cross the path of the worker. Figure 3 is the space time system generated by Algorithm 2. For the first two lines of τ, the worker is allowed to move freely at a constant speed of $v_{worker} = 2\ units/s$. On the third line he has a possibility of collision with the truck. The duration of the collision is $24\ units - 15\ units = 9\ units$ and it occupies the space of $33\ units - 18\ units = 15\ units$. We advise the worker to stop, which is indicated by the vertical green line in Figure 3 from time 10 to $24 = 14\ units$. A vertical line indicates a stop in the space-time system because the term $ds = 0$ and $dt \neq 0$. The truck will come back to the opposite direction on the workers' path at time $43\ units$. This is represented by another rectangle centered at $(25, 43)$. This time the worker has already passed, so there will be no collision. The worker finishes when he reaches the point $(124, 76)$ in Figure 3. This means that the worker took $76\ units$ of time to complete a $124\ unit$ long path. If the worker would not have faced the obstacle, he would have taken $124/2 = 62\ units$ of time. We considered four other alternative paths presented in Figure 4 and 5. If we emphasize safety, the path of Figure 4(a) is a good candidate as it has no collision and takes $77\ units$ of time to finish the length of $155\ units$, which is longer than the shortest path, but also safer. $Safety\ score$, which we defined in *Problem 2*, can be assigned to all possible paths based on safety and distance. For example, if we sort all the trajectories primarily based on the number of collisions they have and secondly on their lengths, the path in Figure 4(a) has a high safety score.

The path in Figure 4(c) is also safe, as it has no collisions, but it is long. Similarly, the path in Figure 5(a) is collision free but very long, and the path in Figure 5(c) is the longest with a length $300\ units$ and with some collision risk.

V. Conclusions and Future Work

In this paper, we presented an easily implementable methodology to proactively detect hazardous conditions and collision states in a construction jobsite and to calculate safe trajectories that are as far as possible from static obstacles while avoiding moving bodies (i.e., equipment). The work presented in this paper is based on combinatorial methods for motion planning that are simplified to allow easy implementation. At this point, the motion planning component requires as input a description of the geometric layout of the construction site, the trajectories of the moving construction bodies, and their speed and initial time that have been defined manually. In the envisioned methodology (on-going research work), these values are automatically obtained from the discrete event model. In the next paragraphs, we will describe extensions and directions for future research.

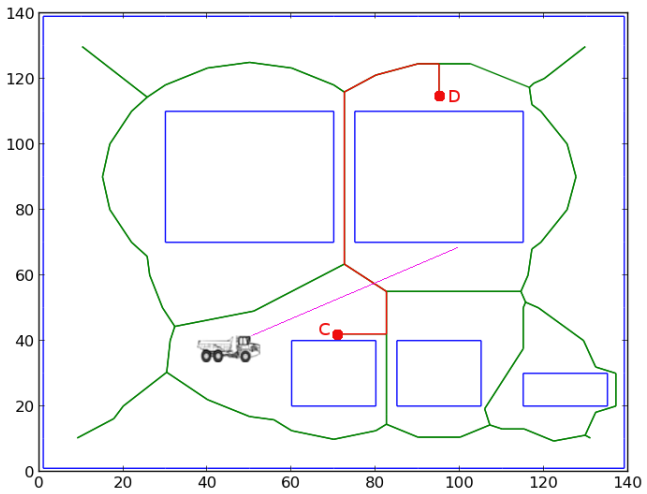

Fig. 2. Generalized voronoi diagram of the construction site

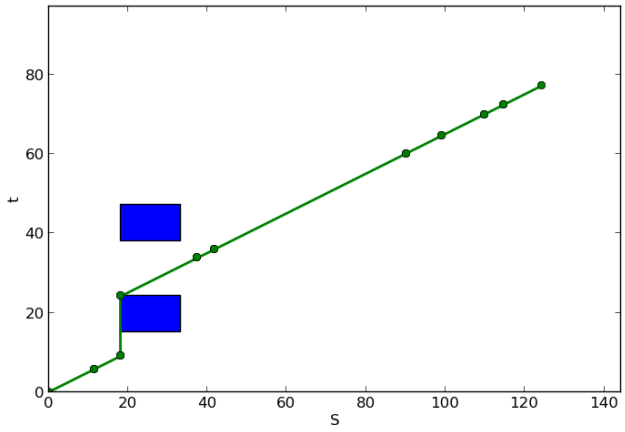

Fig. 3. Obstacles in $s \times t$ space. Vertical line means STOP. Diagonal lines mean MOVE.

The most immediate extension to our work is to remove assumptions about moving bodies and their motions $\mathcal{B}(t)$. We assumed that the bodies, represented by circular disks, move with constant speed in piecewise linear paths, composed of a set of lines. We would like to provide more accurate motions of machinery. First, we would like to incorporate bodies with accurate 3D models and with several degrees of freedom as found in common construction machinery such as cranes and excavators. Second, instead of assuming omnidirectional motion capabilities, we will incorporate dynamic models of the moving machinery that take into account constrains on acceleration and motion (trucks, for example, can not move sideways) [13], [15]. Third, we will integrate the approach presented in this paper with discrete event simulation to automatically model construction activities. Discrete event simulation is widely used in modeling of construction activities; however, the existing methodologies are mainly used for evaluation of project time and cost and are not adequate for investigating safety issues (due to lack of consideration

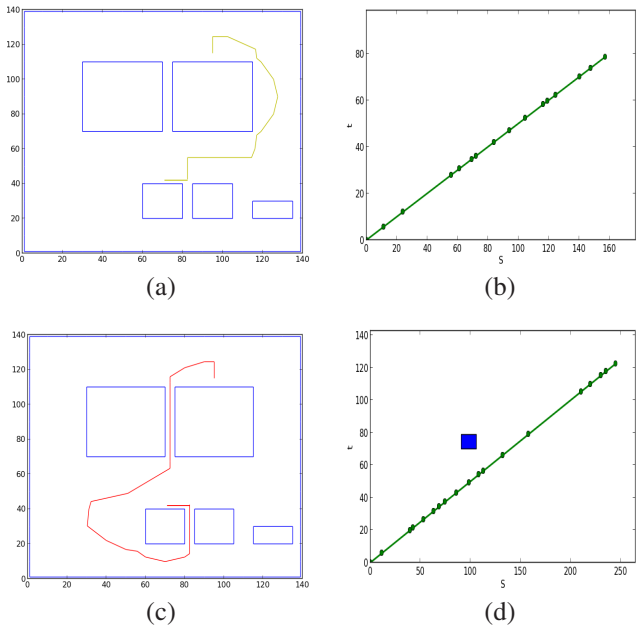

Fig. 4. (a) and (c) Two alternate paths that are not the shortest; (b) and (d) Corresponding velocity profile guidelines from the $s \times t$ graph. There are no collisions, but the paths are longer

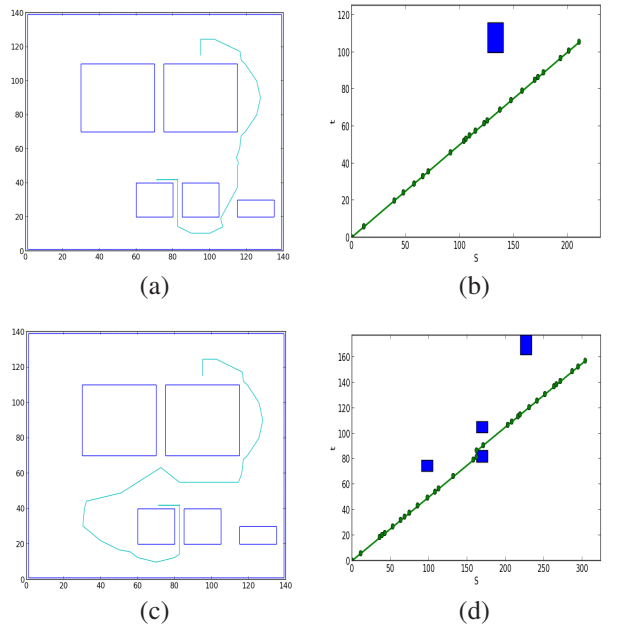

Fig. 5. (a) and (c) More alternative paths that are not shortest; (b) and (d) Corresponding $s \times t$ graph. d) is totally unacceptable as it traverse a long distance having collision chance too.

of spatiotemporal movements of workers and equipment).

The risk score of a path can be used by construction and safety managers to predict vulnerabilities before the execution of a project and modify plans accordingly. For instance, although increasing the number of moving vehicles can reduce the completion time of a construction task, it will increase the danger to the workers, which can be evaluated using our methodology. The interactions between multiple activities, equipment, and workers increase the likelihood of hazards. Since our algorithms have low complexity, they can handle larger problems and deal with these complex hazardous situations.

VI. Acknowldegements

We would like to thank Mr.Winston Newman and the loss prevention team at Balfour Beatty Construction Company for their valuable feedback on the ideas presented in the paper.

References

[1] https://www.osha.gov/SLTC/etools/construction/struckby/mainpage.html.
[2] Priyadarshi Bhattacharya and Marina L Gavrilova. Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path. *Robotics & Automation Magazine, IEEE*, 15(2):58–66, 2008.
[3] Leonardo Bobadilla, Ali Mostafavi, Carmenate Triana, and Sulabh Bista. Predictive assessment and proactive monitoring of struck-by safety hazards in construction sites: An information space approach. ICCCBE, 2014.
[4] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, 2005.
[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (2nd Ed.)*. MIT Press, Cambridge, MA, 2001.
[6] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications, 2nd Ed.* Springer-Verlag, Berlin, 2000.
[7] Amin Hammad and Cheng Zhang. Towards real-time simulation of construction activities considering spatio-temporal resolution requirements for improving safety and productivity. In *Simulation Conference (WSC), Proceedings of the 2011 Winter*, pages 3533–3544. IEEE, 2011.
[8] Jimmie Hinze, Xinyu Huang, and Lani Terry. The nature of struck-by accidents. *Journal of Construction Engineering and Management*, 131(2):262–268, 2005.
[9] Kenneth E Hoff III, John Keyser, Ming Lin, Dinesh Manocha, and Tim Culver. Fast computation of generalized voronoi diagrams using graphics hardware. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 277–286. ACM Press/Addison-Wesley Publishing Co., 1999.
[10] Vineet Rajendra Kamat. *VITASCOPE: Extensible and scalable 3D visualization of simulated construction operations*. PhD thesis, University Libraries, Virginia Polytechnic Institute and State University, 2003.
[11] Kamal Kant and Steven W Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research*, 5(3):72–89, 1986.
[12] J.-C. Latombe. *Robot Motion Planning*. Kluwer, Boston, MA, 1991.
[13] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Also available at http://planning.cs.uiuc.edu/.
[14] S. M. LaValle. Motion planning: The essentials. *IEEE Robotics and Automation Society Magazine*, 18(1):79–89, 2011.
[15] S. M. LaValle. Motion planning: Wild frontiers. *IEEE Robotics and Automation Society Magazine*, 18(2):108–118, 2011.
[16] D. T. Lee and R. L. Drysdale. Generalization of Voronoi diagrams in the plane. *SIAM Journal on Computing*, 10:73–87, 1981.
[17] Julio Cesar Martinez. *STROBOSCOPE: State and resource based simulation of construction processes*. University of Michigan, 1996.
[18] Xin Ning and Ka Chi Lam. Cost–safety trade-off in unequal-area construction site layout planning. *Automation in Construction*, 32:96–103, 2013.
[19] C. O'Dúnlaing and C. K. Yap. A retraction method for planning the motion of a disc. *Journal of Algorithms*, 6:104–111, 1982.
[20] Jochen Teizer, Ben S Allread, Clare E Fullerton, and Jimmie Hinze. Autonomous pro-active real-time construction worker and equipment operator proximity safety alert system. *Automation in Construction*, 19(5):630–640, 2010.
[21] C Zhang, H AlBahnassi, and A Hammad. Improving construction safety through real-time motion planning of cranes. In *Proceedings of International Conference on Computing in Civil and Building Engineering*, pages 105–115, 2010.
[22] Cheng Zhang, Amin Hammad, and Jamal Bentahar. Multi-agent-based approach for real-time collision avoidance and path re-planning on construction sites. In *28th Int. Symp. on Autom. & Robotics in Const., Seoul*, 2011.