Abstract—In this paper, we examine the problem of navigating cluttered environments without explicit object detection and tracking. We introduce the dynamic risk density to map the congestion density and spatial flow of the environment to a cost function for the agent to determine risk when navigating that environment. We build upon our prior work, wherein the agent maps the density and motion of objects to an occupancy cost function, then navigate the environment over a specified risk level set. Here, the agent does not need to identify objects to compute the occupancy risk, and instead computes this cost function using the occupancy density and velocity fields around them. Simulations show how this dynamic risk density encodes movement information for the ego agent and closely models the object-based congestion cost. We implement our dynamic risk density on an autonomous wheelchair and show how it can be used for navigating unstructured, crowded and cluttered environments.

I. INTRODUCTION

Autonomous mobility comes in many forms and packages: cars, trucks, golf carts, wheelchairs, and delivery buggies, to name a few. As we integrate these vehicles into human-centric environments, autonomous systems must detect and adapt to the presence of moving agents and obstacles. The correct response depends on the level of congestion and the nature of the motion. This paper presents a solution to navigation in clutter with dynamic obstacles agnostic to the classification of the obstacle. We formalize a method of estimating the density and motion of the clutter with a congestion cost function. Using the density and velocity field of the environment, we generate the dynamic risk density function, which encodes the location and movement of dynamic obstacles into a single cost function. Using the dynamic risk density, we demonstrate navigating crowded hallways with an autonomous wheelchair that is able to avoid collisions with both static obstacles and pedestrians. Figure 1 shows our autonomous wheelchair platform and an illustration of our path planning algorithm.

This paper focuses on autonomous mobility platforms for individuals, such as autonomous wheelchairs, whose features are needed to navigate crowds, buildings, plazas, airports, and other cluttered and unstructured environments.

In recent years, sensing and perception technology has made significant advances in our ability to parse and understand an environment. However, these perception systems remain prohibitively expensive at the consumer scale. Furthermore, many of these approaches require advanced algorithms with significant computational overhead, or are limited in the environmental complexity they can handle. Here, we are interested in studying mobility solutions achievable with minimal sensor data and light computational loads. Our goal is to rapidly assess and translate risks for reliable, collision-free motion in diverse, cluttered, and unstructured environments. Our previous work proposed using “risk level sets” to navigate a cluttered environment. In [1], we introduced a cost function that quantified the level of congestion based on the positions and movement of other agents and obstacles in the system. This function mapped the environment clutter to a metric of risk for an ego agent planning a path through the environment. We also introduced varying risk thresholds for the agents, such that from the congestion cost function, the agents choose their allowable risk tolerance, which results in a range of conservative to aggressive behavior while still avoiding collisions. Here, we extend the results of our previous work by removing the ability to detect and track obstacles. We present the “object-less” version of our congestion cost function, which approximates the previous congestion cost using only the occupancy density of the environment and the velocity flow field around the agent. We show this closely matches the true congestion cost function in the calculation of the safe planning space.

We implement our dynamic risk density cost function on an autonomous wheelchair navigating busy pedestrian corridors. The wheelchair has a prior map of the environment, and given some initial starting location, must navigate to a goal location along a reference path. The map only includes static...
Our approach uses the density and velocity information to formulate our cost function. We build upon work in occupancy grid-based navigation [28], [29], [30], [31], which combines all available sensor data into an occupancy probability for navigation. By tracking features, it is possible to create spatio-temporal models of the flow patterns throughout the environment [32], [33]. Here, we use the changes in the densities to predict the velocity flow field of the environment, and then compute a weighted costmap for navigation. In [34], [35], the authors use information about the pedestrians in the environment to weight their presence in a costmap differently than other obstacles. Our approach does not directly detect or track obstacles, treating any occupied space as a region to avoid.

The remainder of the paper is organized as follows: Section II defines the congestion cost and problem formulation. Section III presents our dynamic risk density calculation when object detection is not available. We present simulations in Section IV comparing the performance of the object-based congestion cost and its more general form of the dynamic risk density. Section V details our experiments navigating cluttered pedestrian walkways with an autonomous wheelchair, and we state our conclusions in Section VI.

II. PROBLEM FORMULATION

In this section, we present our problem formulation and summarize the congestion cost from prior work [1]. From this congestion cost, we then present our approximation when only the local density and velocity flow field information is known. Consider an environment $Q \subset \mathbb{R}^N$, with points in $Q$ denoted $q$. Here, we consider all agents and obstacles as dynamic obstacles. For $n$ dynamic obstacles in the environment, we write the position of each obstacle as $x_i$, for $i \in \{1, ..., n\}$. For brevity, we write the position of all the agents as $x$, with dynamics $\dot{x}$. We define the ego agent with subscript $e$ and denote its position $x_e$.

The occupancy cost due to all dynamic obstacles in the environment is calculated

$$H(q, x, \dot{x}) = \sum_{i=1}^{n} \frac{\exp \left( -(q - x_i)^T \Omega(q - x_i) \right)}{1 + \exp \left( -\alpha \dot{x}_i^T (q - x_i) \right)}$$

(1)

where $\alpha$ is a scaling factor and $\Omega$ is the diagonal matrix of the inverse square of the standard deviation. In $\mathbb{R}^2$, $\Omega = \text{diag}\{\frac{1}{\sigma_x^2}, \frac{1}{\sigma_y^2}\}$ for $x, y$ axes. By construction, the congestion cost is the summation of a Gaussian peak centered at an obstacle’s location multiplied by a logistic function in the direction of its motion. By skewing the position by the velocity, the occupancy cost $H$ is greater in the direct path of the obstacle’s motion, and lower where obstacles are not actively moving. This particular formulation allows us to make robust claims about the collision avoidance of the ego agent under minimal assumptions of the other agents.

In our previous work, we assume that agents are “self-preserving,” which means all agents will avoid collisions when possible with other agents. This assumption implies that there are no adversarial agents in the environment, and that if the ego agent is capable of avoiding collisions with dynamic obstacles, those obstacles will also attempt to avoid collisions. Unlike other work, we do not assume the collision...
avoidance happens by any particular control policy, nor do we assume the control policies be reciprocal among agents.

Under this assumption, we can find the thresholds in $H$ that define safe and unsafe values, which then allow us to define the risk thresholds. By computing level sets of the cost function, we restrict the ego agent’s planning space to guarantee safe motion through the environment, and further, change the behavior of the agent based on these choices. Let $H_c$ be the value of the cost function when a collision occurs. We define $H_T < H_c$ to be the maximum risk threshold of the system that guarantees an agent can avoid collisions. We also define $H_P < H_T$ as the chosen planning threshold of the ego agent. For more aggressive maneuvers, an agent would choose a value of $H_P < H_T$, while a conservative agent would choose $H_P << H_T$.

For an ego agent planning a path through the environment $Q$, we define the risk level set, $L_p$, as all points in the environment that are below the chosen planning threshold $H_P$. The risk level set is written

$$L_p = \{ q \mid H(q-x_c, x_t) \leq H_P \}. \tag{2}$$

From the level set in (2), it can be shown that an ego agent that only chooses actions within that set will avoid collisions with all other agents [1]. We demonstrate this by proving that the cost function in (1) never exceeds the thresholds $H_c$, and that for any value of $H < H_c$, a collision cannot occur.

When it is feasible to detect and track obstacles, the congestion cost in (1) is effective for preventing collisions between an ego agent and dynamic obstacles within the environment. However, if the autonomous system is not capable of reliably tracking obstacles, using (1) may fail to capture all hazards. In the next section, we detail our dynamic risk density function, which provides an approximation of the congestion cost function without explicit obstacle detection.

III. DYNAMIC RISK DENSITY

When obstacle detection is unavailable, we can still compute an approximation of the congestion cost from sensor information. We introduce the approximation of this cost using the density and the velocity field in the environment. Simulations in Section IV show the fidelity of this approximation to the congestion cost presented in (1). For the experiments on the autonomous wheelchair presented in Section V, the local environment density is calculated from laser scanner information, and we present several methods for estimating the velocity field, detailed later in this section.

A. Dynamic Risk Density from Occupancy and Velocity Fields

Consider the congestion cost function presented in (1). To construct our approximation, we look at substitutions to replace the object locations and velocities with environment density and velocity fields.

Recall the numerator of (1) is constructed from Gaussian peaks about the object locations. In the approximation, we instead use the occupation density of the environment in its place. Let $\rho(q, t)$ define the density at point $q$ in the environment at time $t$. To compute the logistic function without the objects, we use both the density and the velocity field.

We define $\mathcal{V}(q, t)$ as the velocity field of the environment at time $t$. Overall, the dynamic risk density is computed

$$H_\rho(q, t, \rho, \mathcal{V}) = \frac{\rho(q, t)}{1 + \exp(\alpha \nabla \rho(q, t) \cdot \mathcal{V}(q, t))}, \tag{3}$$

where $\nabla \rho(q, t)$ is the gradient of the density, and $\alpha$ is a user-designed scaling factor. For brevity in future notation, we use the superscript $t$ to denote functions at a particular time, letting $\rho^t$ and $\mathcal{V}^t$ denote the density and velocity field, respectively. In Section V, we use (3) in conjunction with an RRT*-based controller to navigate a cluttered environment autonomously. We estimate the environment density directly from laser range scanners and occupancy grids. The velocity field may be computed in several ways: here, we present two distinct methods for determining the velocity field from the density.

B. Estimating the Velocity Field from Clustering

To estimate the velocity field from the density information, we applied a k-means clustering algorithm [36] to the occupancy grid, then track the movement of the centroids over time. Using the movement of the centroids, we approximate their velocity. To map velocity for all points in the environment, we calculate the Voronoi tessellation of the environment from the centroids and apply the velocity of the centroid to all points within its associated Voronoi cell. A summary of our approach is given in Algorithm 1.

Let $c^t_k$ denote the position of a centroid at time $t$, with $C^t = \{(c^t_1)^T, \ldots, (c^t_k)^T\}^T$ representing the positions of all $k$ centroids from the clustering algorithm. At each time $t$, the k-means clustering algorithm is seeded using the centroids $C^{t-\Delta t}$. We then use the Hungarian algorithm [37] to match the clusters over time and assign the indices from $C^{t-\Delta t}$ to points in $C^t$. From the matched pairs, we compute the velocity of a centroid as

$$v^t_k = \frac{(c^t_k - c^{t-\Delta t}_k)}{\Delta t}, \tag{4}$$

where $\Delta t$ is the time elapsed between computations. To compute the velocity field $\mathcal{V}^t$, we apply the centroid velocity $v^t_k$ to all points within the associated Voronoi cell $V^t_k$,

$$\mathcal{V}^t = \sum_{i=1}^{k} v^t_k \mathbf{1}_{V^t_k}, \tag{5}$$

where $V^t_k$ is the Voronoi cell about centroid $c^t_k$, computed

$$V^t_k = \{ q \mid \|q - c^t_k\| \leq \|q - c^j_k\|, \forall i, j \in k, i \neq j \}. \tag{6}$$

Using this approach, the clusters track areas of high occupancy, yielding a notion of features for the ego agent. In the experiments presented in Section V, we demonstrate this approach on our autonomous wheelchair, as well as a variation that incorporates a Kalman filter on the centroid velocity estimates to reduce the effect of noise.
Algorithm 1 Velocity Field Estimation from Clustering
1: Input: $\rho^t, C^t\rightarrow\Delta t$
2: Compute clusters $C^t$ from $\rho^t$
3: Match clusters $C^t$ to $C^{t-\Delta t}$
4: Compute cluster velocities $v_i^t$ (4)
5: Estimate velocity field $V^t$ from Voronoi tessellation (5)

C. Velocity Field Estimation from Density Flow

In contrast to the clustering approach with discrete estimates, here we present a method for estimating velocity field $V^t$ directly from occupancy density changes $\rho^t$ and $\rho^{t-\Delta t}$. Our approach is inspired by dense optical flow estimation with applications in computer vision. For a 2D+t dimensional case a voxel at location $(q = (x, y), t)$ with density $\rho(x, y, t)$, $\rho(x, y, t)$ will have moved by $\Delta x, \Delta y,$ and $\Delta t$ between the two measurements, and the following density constancy constraint can be formulated:

$$\rho^t(x, y) = \rho(x, y, t) = \rho(x + \Delta x, y + \Delta y, t + \Delta t)$$ (6)

Under the assumption of small movement the density can be approximated by first order Taylor expansion to

$$\rho(x + \Delta x, y + \Delta y, t + \Delta t) = 
\rho(x, y, t) + \frac{\partial \rho}{\partial x} \Delta x + \frac{\partial \rho}{\partial y} \Delta y + \frac{\partial \rho}{\partial t} \Delta t + \text{H.O.T.},$$

such that by employing the density constancy condition (6) and dividing by $\Delta t$ the partial differential equation

$$\frac{\partial \rho}{\partial x} \Delta x + \frac{\partial \rho}{\partial y} \Delta y + \frac{\partial \rho}{\partial t} \Delta t = 0,$$ (7)

relates the density gradient $\nabla \rho = \left( \frac{\partial \rho}{\partial x}, \frac{\partial \rho}{\partial y} \right)$ to the density flow $V^t = \left( \frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t} \right)$. We approximately solve (7) as described in [38] to estimate the density flow $V^t$ from two sequential densities $\rho^t, \rho^{t-\Delta t}$ based on local polynomial expansion similar to estimating dense optical flow in images.

D. Control Policy

While the risk level set defines the planning space, the user may implement their chosen control policy for navigating the environment. In Section V, we implement an RRT*-based controller on the wheelchair in which the congestion cost is used to update the local path to avoid obstacles. Algorithm 2 provides an overview of our chosen RRT*-based control policy used in our experiments.

Algorithm 2 Dynamic Risk Density Navigation
1: Update $\rho^t$ from sensors
2: Estimate velocity field $V^t$
3: Compute dynamic risk density (3)
4: Compute risk-bound path using RRT* along reference path

IV. SIMULATIONS

To benchmark the dynamic risk density, we ran simulations in Matlab and Python to compare the performance against the congestion cost function in (1) and formalized in [1]. The object-based cost function is computed directly from the object positions and velocities, while the dynamic risk density is computed using the density and velocity flow field. For this comparison, we generate the density field from the object positions and velocities, while the dynamic risk density is computed using the density and velocity flow field. From Figure 2, we see that the contours of both cost functions are quite similar. To further compare the variations between the two cost functions, Figure 3a tracks the values of the level set $L_p$ for the congestion cost (blue) and dynamic risk density (red).

Fig. 2: Contours of the (a) object-based cost function and (b) dynamic risk density.

Fig. 3: (a) From a simulation of $n = 10$ obstacles moving through the environment, we can compare the value of the congestion cost (blue) in (1) to the dynamic risk density (red dashed) in (3). The dynamic risk density yields a similar planning area to (1). (b) For varying numbers of obstacles, we ran 50 trials of randomized configurations and computed the planning area $L_p$ for the congestion cost (blue) and dynamic risk density (red).
Fig. 4: Contours of the dynamic risk density with the velocity field determined by applying the k-means cluster velocity to a Voronoi tessellation of the environment. Over time, we use the evolution of the clusters to determine the velocity field.

V. Experiments

To validate our dynamic risk density cost function, we performed experiments using an autonomous wheelchair developed at MIT. The hardware platform is inspired by the Singapore-MIT Alliance for Research and Technology (SMART) scooter [39] and golf-cart [40] platforms. The wheelchair uses the same autonomy stack developed for the MIT autonomous Prius platform [41] detailed below. For the experiments, we use laser scanners to construct occupancy grids in the environment that are used for motion planning. Figure 6 shows stills from the video included with the submission of our paper synced to snippets of the costmap.

A. Platform Overview

The experiments are performed on a wheelchair modified for drive-by-wire, which runs an autonomy stack composed of control, navigation, and sensor modules implemented in ROS [42]. The sensor modules interface with the laser scanners, encoders, and an IMU that equip the wheelchair. The control modules provide linear and angular velocity controllers, where the reference signals are given by the navigation modules, as well as emergency stop and human-input control signals. The navigation stack comprises localization, path planner, path following, and costmap modules. Localization is performed using the AMCL package [43], [44], which integrates odometry poses computed from the encoders and IMU data, and laser scan data registered into an a priori constructed map. The location of the wheelchair together with a reference path and costmaps are used by an RRT* planner [45] to generate obstacle-free paths in the environment. We use a pure-pursuit controller [46] for path following, which provides the steering control signal. For simplicity, we command constant linear velocity, unless an obstacle is too close to the wheelchair and all velocities are set to zero. Collision checking is performed using a discrete costmap-based approach, where grids are populated with laser scan measurements. The ROS costmap_2d package [35] computes snapshot costmaps that integrate static map information with obstacles generated from the laser scan data, and inflated radially for safety.

Our proposed dynamic risk density implementation interfaces with the vanilla costmap and path planning modules. It takes in the costmap stream, and generates discrete (grid) risk cost functions that take into account the underlying velocity fields of cluttered (occupied) space. The cost function is passed to the path planner module that computes obstacle-free paths at the current time, and potentially within a future time horizon. The dynamic risk density computation ran at 60Hz on a Dell Precision 5520 laptop with Core i7-7th gen processor and 32GB RAM, allowing the wheelchair to react quickly to the environment and operate smoothly. A visualization of the path planning is shown in Figure 5.

B. Performance Comparison

We conducted our experiments in the Stata Center at MIT. We ran trials along the a busy pedestrian corridor during weekday afternoons, and compared the performance of our navigation algorithm in Algorithm 2 using several methods of computing the velocity field. These variations were:

- **Snapshot** costmap setting \( V^t = 0 \) (baseline using only occupancy grid information).
- **Density flow** using (7) to solve for \( \nu^t \), akin to optical flow methods in image processing.
- **k-means** clustering of the occupancy grid, then estimating \( \nu^t \) using the Voronoi approach in Algorithm 1.
- **Filtered k-means** clustering with a Kalman filter on the centroid velocity estimates \( \nu^t \) before mapping to \( \nu^t \).

First, we computed the velocity field using the k-means clustering approach with the Voronoi-based velocity field, as described in Algorithm 1. Next, we ran a modified version of the clustering algorithm, this time with a Kalman filter on the generated cluster velocities \( \nu^t \). We also ran several trials with the velocity field generated from the density flow approach outlined in Section III. As a baseline, we compare our results to the snapshot costmap.

Overall, we found the snapshot costmap to be either too conservative in its obstacle padding, or too aggressive and collision-prone. For the two variations on clustering, filtering the velocity helped reduce noise in the estimation.
of static obstacles, but introduced a delay in responding to new dynamic obstacles. Both versions of our clustering algorithm were performed with $n = 50$ clusters over our costmap. As shown in Figure 6 and seen in our video supplement, the clustering approach is able to handle a congested hallway where people are entering and exiting a classroom, introducing new obstacles.

While the clustering approach provides a discrete approximation of points of interest in the environment, the density flow-based velocity field generates a high-resolution, fine-grained estimation of the velocity field. It yielded a better approximation of the environment boundaries, but was more susceptible to artifacts in the density map, and clipped objects within the costmap due to more discontinuities in velocity over the field. Figure 7 shows the four versions of our costmap from a single reference point.

VI. CONCLUSIONS

In this paper, we propose the use of a dynamic risk density function for navigating a cluttered environment without explicit object detection and tracking. The dynamic risk density is computed by taking the congestion density and spatial flow of an environment, and mapping it to a cost function that approximates risk for a navigating ego agent. We build upon prior work, which proposed the use of risk level sets to safely navigate a cluttered environment without collision. Simulations illustrate how the dynamic risk density provides a high-fidelity approximation of the congestion cost function. We also propose two methods of calculating the velocity field: first, using a k-means clustering-based approach, which provides discrete approximations of features; and second, a density flow approach that provides a fine-grained approximation of the velocity field. Experiments on an autonomous wheelchair platform demonstrate how the dynamic risk density can be used in cluttered environments, and we perform multiple trials comparing the different instantiations of our velocity field estimation as compared to a “snapshot” costmap.
[41] F. Naser, D. Dorhout, S. Proulx, S. D. Pendleton, H. Andersen, W. Schwarting, L. Paull, J. Alonso-Mora, M. H. Ang, S. Karaman, R. Tedrake, J. Leonard, and D. Rus, “A parallel autonomy research platform,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 933–940.

[42] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.

[43] D. Fox, “Adapting the sample size in particle filters through kld-sampling,” *The international Journal of robotics research*, vol. 22, no. 12, pp. 985–1003, 2003.

[44] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[45] S. Karaman and E. Frazzoli, “Sampling-based Algorithms for Optimal Motion Planning,” *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[46] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, “Real-time motion planning with applications to autonomous urban driving,” *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, September 2009.