FlowBot: Flow-based Modeling for Robot Navigation

Daniel Dugas, Kuanqi Cai, Olov Andersson, Nicholas Lawrance, Roland Siegwart and Jen Jen Chung

***Abstract*— Autonomous navigation among people is a complex problem that also exhibits considerable variation depending on the type of environment and people involved. Here we consider navigation among crowds that exhibit flow-like behavior like people moving through a train station. We propose a novel pseudo-fluid model of crowd flow for such problems. These have an intuitive physical interpretation and do not require much tuning. We further formalize an observation model to infer flow properties from discrete sensor observations, including support for partial observability, and pair it with a flow-aware planner. We demonstrate the potential of the approach in simulated navigation scenarios. We achieve state of the art results on the CrowdBot navigation benchmark, and also compare favorably against a standard ROS planner on a partially observable environment, demonstrating that the flow-aware planner successfully estimates and plans around counter-flows in the crowd in real time. We conclude that flow-based planning shows great promise for crowded environments that may exhibit such flow-like behavior.**

I. Introduction

Autonomous navigation among people is both an ubiquitous and complex problem. One thing which emerges from the many efforts to operate robots in various public spaces is that they present a variety of situations, which can require switching between different specific planning methods [1], [2].

One specific scenario which is often observed in, for example, train stations and other public spaces which serve a mass-transit function, is the presence of medium to high density homogeneous masses of people moving together. These flows can be either clearly separated or turbulent. In the field of crowd-modeling, analogies to fluid flow are often employed when describing such dynamics [3]–[5]. Generally speaking, this is on the opposite end of the spectrum from individual or group modeling, which is often dominant in navigation planning work [6].

Examples of navigation planning in crowd flow environments include [7], which proposes a group-surfing planner that computes and matches the local flow by explicitly detecting moving groups of pedestrians. Henry et al. [8] propose an approach which is shown to avoid areas of counter-flow. However, rather than model crowd flow-robot interaction explicitly, their approach uses inverse reinforcement learning to imitate human navigation behavior.

Prior work by [9] described a deep-learning-based controller to regulate pedestrian flows through robot interaction.

This work was supported by the EU H2020 project CROWDBOT under grant nr. 779942 and a Wallenberg Foundation and WASP Postdoctoral Scholarship.

The authors are with the Autonomous Systems Lab, ETH Zürich, Zürich 8092, Switzerland. {dugasd; nandersson; lawrancn; rsiegwart; chungj}@ethz.ch, kayle.ckq@gmail.com

Fig. 1. Left: flow planner being tested in a simulated environment (CrowdBot-Challenge). Right: example of a corresponding real-life situation.

The controller takes a pedestrian flow estimate as input, and may only move the robot along a pre-determined trajectory. Fan et al. [10] similarly used deep learning methods to explore whether crowd-flow modeling can be used to help with mapping and localization. This crowd-flow map is also used to influence trajectory planning. To model the flow, K-means clustering is used to compute static velocity patterns in a discrete grid.

While data-driven methods have enjoyed some success on this problem, they can be opaque and require considerable data for training. In this work we instead propose an *interpretable* crowd-model based on pseudo-fluid simulation, and pair it with a planner for flow-aware crowd navigation. Our contributions are:

- We propose and design a navigation planner for robots in crowded environments based on pseudo-fluid models. These have a physical interpretation and require little tuning.
- We validate that this model is able to estimate crowd flow in simulated scenarios, and use this estimate to plan trajectories which avoid regions of counterflow.
- We show that adding flow awareness to classic local planning approaches results in improved success rates, efficiency and safety in the CrowdBot Challenge navigation benchmarks. We show that the planner can work in realistic conditions, including partial crowd observations, in real time.

II. Crowd Flow Modeling Considerations

There is a range of possible ways to formalize the problem of modeling a crowd as a fluid. Depending on assumed flow properties, the equations for approximating fluid dynamics contain many differential terms for physical quantities or phenomena which are not relevant to actual crowds and make them difficult to apply in practice.

Here we want to find an interpretable but simplified model sufficient for crowd navigation. To do so, we make a few observations based on empirical data and previous work:

1) Walking against a flow of people is harder than going with the flow. At the macroscopic level this is similar to

the effect of *viscosity* in fluids, which models the forces acting against objects moving within the fluid, as well as those forces within the fluid itself that affect its flow.

2) In denser crowds it is harder to deviate from the crowd velocity without causing collisions. This suggests a relationship between *viscosity* and *density*.

3) Obstacles tend to slow down crowd movement. The local slow-down can affect nearby flow, analogous to the *shear stress* equations in fluids.

4) Crowd masses can move homogeneously or chaotically. A crowd where individuals are moving randomly and a crowd where they are standing still may have the same density and average velocity. However, intuitively we expect that a robot traversing these crowds will be impacted in dissimilar ways. We propose to model this using the concept of *turbulence* in fluids.

Finally, we recognise that flow modeling is macroscopic in nature. While very local person-robot interactions should be individual-based in order to avoid collisions, flow-based planning should ideally target a higher-level planning abstraction, reasoning at the level of global trajectories to a goal in a flow map. Each of these considerations motivate our crowd flow modeling and planning formulation below.

III. Flow Modeling

In this section we present the details of our approach for modeling a crowd as a pseudo-fluid.

A. State Representation

Given the desired pseudo-fluid properties outlined in Section II, we select the following variables to model flow-like properties of the crowd flow state over space and time:

$$\begin{aligned} \textbf{Density} \quad & \rho(\mathbf{x}, t) : \mathbb{R}^2 \times \mathbb{R}^+ \to \mathbb{R}^+ \\ \textbf{Velocity} \quad & \mathbf{v}(\mathbf{x}, t) : \mathbb{R}^2 \times \mathbb{R}^+ \to \mathbb{R}^2 \\ \textbf{Turbulence} \quad & \tau(\mathbf{x}, t) : \mathbb{R}^2 \times \mathbb{R}^+ \to \mathbb{R}^+ \end{aligned} \tag{1}$$

Where $\mathbf{x}$ is a spatial (x, y) coordinate and t denotes the timestep. **Density** corresponds to the amount of people per unit area. **Velocity** is the mean (v_x, v_y) velocity of the crowd (e.g. 0 if everyone moves chaotically). **Turbulence** is defined as the mean particle speed minus the magnitude of the mean crowd velocity. It represents the absolute deviation in velocity,

$$\tau(\mathbf{x}, t) = E(\|\mathbf{v}(\mathbf{x}, t)\|) - \|E(\mathbf{v}(\mathbf{x}, t))\|. \tag{2}$$

These virtual fields can be thought of as terms controlling the distributions from which individual people detections are independently sampled when measurements are made.

B. Fluid Mechanics-inspired Model

Here we discard many of the terms present in equations for real fluids, and focus on the effect which is most important to our navigation objective. As outlined in Section II, the main flow aspect we care about is viscosity, which approximates the tendency of humans to move at similar speeds when close together. As a result, we simplify viscosity to a constraint on the second derivative of velocity,

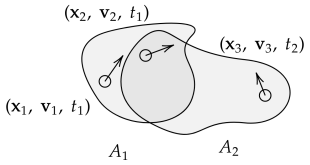

Fig. 2. An example set of observation areas $Z_a = \{(A_1, t_1), (A_2, t_2)\}$ and associated detections $Z_p = \{(\mathbf{x}_1, \mathbf{v}_1, t_1), (\mathbf{x}_2, \mathbf{v}_2, t_1), (\mathbf{x}_3, \mathbf{v}_3, t_2)\}$ which result from two consecutive sensor measurements.

$$\frac{\partial^2 v_x}{\partial y^2} < \frac{1}{\mu}, \quad \frac{\partial^2 v_y}{\partial x^2} < \frac{1}{\mu} \tag{3}$$

Viscosity μ could be modeled as a varying quantity itself, which could depend on ρ, or space, or even other properties such as crowd mood, etc. In this preliminary work, we consider μ to be a constant, though it may be valuable to model this term in more detail in further research. Unlike the viscosity model above, pressure terms lead to computationally expensive differential equations, while providing only a slight increase in how well the model fits actual crowd behavior at the time-scales we are interested in, and is much less relevant to robot behavior. For this reason, in this work we disregard explicit pressure terms. However, we do assume that a continuity in the flow field (which is implicitly related to pressure) leads to constraints on the variation of velocity along the velocity direction.

$$\frac{\partial^2 v_x}{\partial x^2} < \frac{1}{\omega}, \quad \frac{\partial^2 v_y}{\partial y^2} < \frac{1}{\omega} \tag{4}$$

As the time-based dynamics of crowd flows are not well understood, we opt for a quasi-static flow model. We assume that quantities may change over time, and that this change is continuous, and the rate of this change bounded. But we make no further assumptions about this rate of change (no $\frac{\partial^2 \mathbf{v}}{\partial t^2}$ constraints, etc). Effectively, this allows the robot to adapt to changes in the overall flow of the crowd.

We use this fluid mechanics-inspired model to estimate the effects of the crowd on the robot's trajectory. These considerations are explicitly dealt with in Section IV on robot planning. Here we assume that the effect of the robot's motion on the crowd is insignificant compared to the dominant crowd flow characteristics and therefore do not explicitly model any robot-crowd interactions.

C. Partial Observability

Since we consider the crowd as a fluid, although the density can vary, we consider it as having no holes. This means the velocity is not zero where there are no people, it is simply not measured at that moment. Only where there are people standing still is the velocity measured as zero. This has the consequence that our flow-based planner operates in the space of smooth velocity fields, not individuals.

However, although we assume a continuous fluid, due to practical constraints, measurements are discrete in nature. So, as inputs to the measurement model we consider the set of detected people $Z_p = \{(\mathbf{x}_j, \mathbf{v}_j, t_i)\}$, and the set of observed

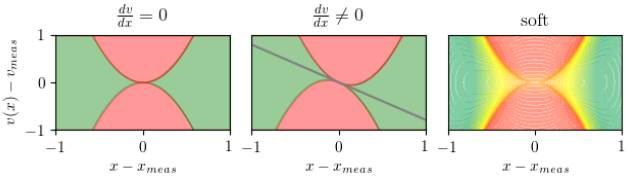

Fig. 3. Allowed (green) and disallowed (red) values of the scalar field $v_y(x)$ around a measurement $(\mathbf{x}_i, \mathbf{v}_i)$, according to the constraint set by $\frac{\partial^2 v_y}{\partial x^2} < \frac{1}{\mu_{max}}$. Left is the linearized case where $\frac{\partial v_y}{\partial x} = 0$. Middle is where $\frac{\partial v_y}{\partial x}$ is an arbitrary non-zero value. Right is the relaxation from hard to soft constraint.

areas $Z_a = \{(A_i, t_i)\}$ over all measurement timesteps, see Fig. 2. The observed areas Z_a are important to quantify how frequently a given region has been observed. This allows us to distinguish between situations where we detect no people in a region and situations where we just have no (or very few) observations of the region.

D. State Estimation

1) Density Estimator: We assume that $\rho(\mathbf{x})$ is a smooth function, which when integrated over a measurement area A_i gives us the expected amount of detections obtained from that measurement.

For a set of detections and measurements Z_p, Z_a, we use the following density estimator:

$$\hat{\rho}(\mathbf{x}) = \frac{1}{2\pi\sigma^2}\frac{1}{N_{obs}(\mathbf{x})}\sum_{j\in Z_p} e^{-\frac{1}{2}((\frac{x-x_j}{\sigma})^2+(\frac{y-y_j}{\sigma})^2)}, \quad (5)$$

where $N_{obs}(\mathbf{x})$ is the number of times $\mathbf{x}$ was observed, i.e. $N_{obs}(\mathbf{x}) = |\{\forall A_i \in Z_a | \mathbf{x} \in A_i\}|$.

The parameter σ controls the spatial smoothness of our recovered estimate. We approximate it as a constant.

2) Velocity Estimator: We soften the viscosity constraints so that they become error terms, which can then be minimized (see Fig. 3). For tractability, we linearize around the measurement, i.e for a single measurement we assume $\frac{\partial \mathbf{v}}{\partial x} = 0$, which leads to a well-defined error function. Despite this linearization, over many measurements we are able to recover the true derivative, as shown in our results.

For this error function $\xi_j : R^2, R^2 \to R^2$, we choose a parabolic function as it is analytically differentiable. This implies the following form,

$$\xi_j(\mathbf{x}, \mathbf{v}) = \alpha(\mathbf{x})((\mathbf{v} - \mathbf{v}_j) \odot (\mathbf{v} - \mathbf{v}_j)) + \beta(\mathbf{x}) \quad (6)$$

Where the terms α and β ($R^2 \to R$) are functions of $\mathbf{x}$. For a set of measurements Z_p, the total soft-constraint error is,

$$\xi(\mathbf{x}, \mathbf{v}) = \sum_{(\mathbf{x}_j, \mathbf{v}_j)\in Z_p} \xi_j(\mathbf{x}, \mathbf{v}), \quad (7)$$

which can be minimized to find an estimate of v_x,

$$\hat{v}_x = \underset{v_x}{\operatorname{argmin}} \frac{\partial \xi}{\partial v_x} = \frac{\sum_{j\in Z_p} \alpha v_{xj}}{\sum_{j\in Z_p} \alpha}. \quad (8)$$

Similarly for v_y. While any weight function could be selected for α, we have the additional smoothness requirement required by (4). For this, we choose,

$$\alpha = e^{-\gamma(\|\mathbf{x}-\mathbf{x}_j\|)^2}, \quad (9)$$

where γ estimates the crowd flow 'viscosity' μ.

Note: a shortcut taken here is the assumption that $\omega = \mu$. For different values of ω and μ (described in Eq 3 and 4, α in (6) would need to take matrix form.

3) Turbulence Estimator: To compute turbulence, we employ the same estimator, but apply it to the magnitude of the velocity vector, which can then be used to compute τ,

$$\tau = \|\hat{\mathbf{v}}\| - \|\hat{\mathbf{v}}\|, \quad (10)$$

where,

$$\|\hat{\mathbf{v}}\| = \frac{\sum_{j\in Z_p} \alpha \|\mathbf{v}_j\|}{\sum_{j\in Z_p} \alpha}. \quad (11)$$

The resulting state estimation procedure is described in Algorithm 1.

Algorithm 1 Discretized state estimation

while 1 **do**
  t
  if Detection $(\mathbf{x}_j, \mathbf{v}_j, t)$ **then**
    $Z_p \leftarrow (\mathbf{x}_j, \mathbf{v}_j, t)$
  if Observation A_i **then**
    $Z_a \leftarrow (A_i, t)$
  $Z_p^{sampled} \leftarrow \text{sampling}(Z_p, t)$ ▷ (12)
  $Z_a^{sampled} \leftarrow \text{sampling}(Z_a, t)$ ▷ (13)
  for $\mathbf{x} \in Grid$ **do**
    $\hat{\rho}(\mathbf{x}) = \frac{1}{2\pi\sigma^2}\frac{1}{N_{obs}(\mathbf{x})}\sum_{j\in Z_p^{sampled}} e^{-\frac{1}{2}((\frac{x-x_j}{\sigma})^2+(\frac{y-y_j}{\sigma})^2)}$
    $\hat{v}_x = \frac{\sum_{j\in Z_p^{sampled}} \alpha v_{x,j}}{\sum_{j\in Z_p^{sampled}} \alpha}$
    $\hat{v}_y = \frac{\sum_{j\in Z_p^{sampled}} \alpha v_{y,j}}{\sum_{j\in Z_p^{sampled}} \alpha}$
    $\|\hat{\mathbf{v}}\| = \frac{\sum_{j\in Z_p^{sampled}} \alpha \|\mathbf{v}_j\|}{\sum_{j\in Z_p^{sampled}} \alpha}$
    $\tau = \|\hat{\mathbf{v}}\| - \|\hat{\mathbf{v}}\|$

E. Temporal Measurement Model

Due the quasi-static state assumption mentioned in Section III-B, the underlying flow state could be changing over time. Given this uncertainty, we use a naive method, where we sample a subset of the complete measurements set (Z_p, Z_a), decreasing the probability of sampling past samples the older they are. Due to current timestep t,

$$Z_p^{sampled} = \{(\mathbf{x}_j, \mathbf{v}_j, t_i) \in Z_p | \text{binomial}(\lambda^{t_i - t})\}. \quad (12)$$

This time-decay sampling rate λ is an empirically chosen parameter which represents our prior over the flow dynamics.

In order to keep the ratio of measurement to observation unbiased, Z_a must be sampled using the same formula,

$$Z_a^{sampled} = \{(A_i, t_i) \in Z_a | \text{binomial}(\lambda^{t_i - t})\}. \quad (13)$$

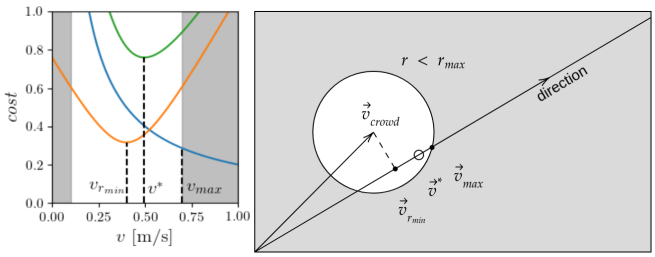

Fig. 4. Left: Illustration of the cost-optimal v^* velocity along a chosen direction. The total cost (green) is the sum of the resistance cost term (orange) and the time cost term (blue). The fluid-resistance constraint limits the possible choice of robot velocities (white). Right: The same terms shown in a top-down view of the velocity-space (bottom-left corner is origin), for a given robot position. The velocities satisfying the fluid-resistance constraint are now a white circle around the current crowd velocity at that position. Velocities minimizing each cost term along the given direction are shown as black circles.

F. Static Obstacles

A practical approach for taking into account static obstacles such as walls and pillars in our state estimation is to generate 0-velocity detections at their positions. These 0-velocity detections are only considered when calculating the velocity and turbulence estimates, not density.

IV. Planning

Our desired planner should take as input a fluid state estimate and robot state, and find an optimal trajectory through the environment.

A. Cost Function

To do so, we must first define a cost function. We return to our previous focus on fluid viscosity and define a resistance term at any robot position and velocity in the field as,

$$r(\mathbf{x}_{robot}, \mathbf{v}_{robot}) = \frac{1}{\kappa}\|\mathbf{v}_{robot} - \hat{\mathbf{v}}(\mathbf{x}_{robot})\|. \tag{14}$$

where κ can be thought of as the 'allowed' deviation from the current crowd velocity, in m/s,

$$\kappa = \frac{1}{\rho\mu} + \tau. \tag{15}$$

We can select a simple heuristic cost, which minimizes the resistance and travel time for any trajectory T,

$$J(T) = \sum_{(\mathbf{x}_t, \mathbf{v}_t) \in T} r(\mathbf{x}_t, \mathbf{v}_t) + J_{time}(\mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{v}_t). \tag{16}$$

J_{time} is the time required to traverse the distance between $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ at velocity $\mathbf{v}_t$. Note that when the fluid state is unknown or there are no humans present, the resistance term r goes to 0 and the planner reverts to a standard time-optimizing navigation planner.

B. Velocity Constraints

Apart from the obvious physical limits of the robot, i.e $\|\mathbf{v}_{robot}\| < v_{\max}$, we also define constraints which limit the space of possible velocity commands.

1) Fluid-robot constraints: We restrict the robot to select velocities which do not deviate too much from the crowd flow. This maximum deviation is calculated by defining a maximum allowable resistance which prevents unsafe or unacceptable behaviors,

$$r(\mathbf{x}_t, \mathbf{v}_t) < r_{\max}, \quad \forall(\mathbf{x}_t, \mathbf{v}_t) \in T. \tag{17}$$

This constraint can be visualized in Fig. 4.

2) Cost-optimality constraint: Along a given direction ($\mathbf{d}$), we constrain the velocity to a single possible value, v^*, the velocity for which the cost equation (16) is minimized, subject to (17). See Fig 4 for an illustration.

$$v^*(\mathbf{x}, \mathbf{d}) = \underset{v}{\operatorname{argmin}} \frac{1}{\kappa}\|\mathbf{d}v - \hat{\mathbf{v}}(\mathbf{x})\| + \frac{1}{v}. \tag{18}$$

This constraint allows us to greatly reduce the space of possible actions that need to be considered for planning. At a given coordinate $(\mathbf{x}, t)$, our planner only needs to select a direction ($\mathbf{d}$, unit vector) and a distance to move.

3) Slow Movement: We consider stopping to be an exception to the fluid-robot viscosity constraint. (In this regime, the robot could be considered a static obstacle.) Inspired by [11], for robots which are light enough not to present a safety risk, we can extend this region slightly by allowing very slow movement in any case (crawl behavior), $\|\mathbf{v}\| < v_{crawl}$. Though this may not apply to the most extremely packed, static crowds, previous work [2] shows that up to relatively high-density ($2\frac{people}{m^2}$ scenarios, a value of $0.1\frac{m}{s}$ may be used for v_{crawl}.

C. Optimization Algorithm

The above definitions in principle allow use of any conventional planning method, such as rapidly-exploring random trees [12] or gradient-based approaches [13], to minimize navigation costs on the flow-model. Here we demonstrate the approach using Dijkstra's algorithm to minimize the navigation cost. We define each node as an (x, y) coordinate in a discrete grid of fixed resolution and we use an 8-connected scheme to define the edges between nodes. According to the quasi-static state assumption mentioned in Section III-B, we consider the fluid state as constant at the planning stage, this allows us to drastically reduce the number of nodes in the graph. Due to the nature of velocities, the graph is a directed graph. For each graph edge, a unique velocity is calculated using (18). Edge costs are then calculated with (16). The result is a trajectory containing the desired nominal velocity at every timestep $T^* = \{(\mathbf{x}_t, \mathbf{v}_t)\}$.

D. Low-level Control

The flow-based planning algorithm outputs a recommended immediate velocity, and a trajectory to the goal which is flow-aware. However, by design, this nominal velocity does not take into account individuals around the robot, which in short-term planning can lead to undesired collisions. For this reason, the nominal velocity tracking the flow-aware trajectory can be given as input to a short-term collision-avoidance planner, such as RVO. We benchmark these alternatives in Section V-B.

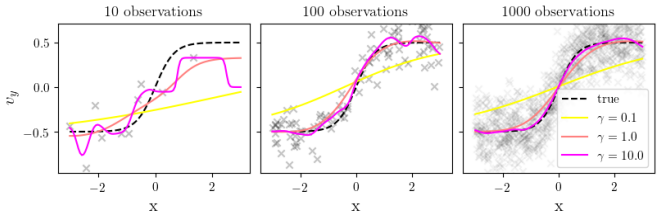

Fig. 5. Simulated testing of the state estimation model. True velocity field (dashed), sampled measurements (crosses), and state estimates for various values of the parameter γ. We see how different values directly affect the resulting 'viscosity' or 'smoothness' of the state estimate.

V. Experiments

A. Validity of the Measurement Model

Assuming a true velocity field over space, we draw samples of person detections stochastically by sampling velocities from a normal distribution around the modeled 'true' velocity value at that point. This simulates detection and tracking errors as well as turbulent crowd movement.

We find that for a high enough value of the parameter γ, the state estimate given by (8) approaches the true value as the number of samples increase (Fig. 5).

The value of the parameter γ is tied to the expected viscosity of the crowd pseudo-fluid, too low a value prevents the estimate from having large curvature in the velocity field, whereas a very high value can lead to instability in the state estimate when few measurements are available.

B. CrowdBot Challenge

The CrowdBot Challenge is a simulation benchmark for comparing the performance of various navigation planning algorithms in a range of challenging scenarios [14]. These scenarios take place in a long corridor, while simulated crowds move in various patterns. The robot must cross the corridor from one end to the other in order to succeed.

These scenarios can be broadly separated into low-density and high-density crowds, crowds moving with the robot, against the robot, both, or sideways, in addition to reactive vs. non-reactive crowds. There are 40 scenarios in total.

To evaluate our flow-based method, we test it in the Crowdbot Challenge benchmark, which allows us to compare its performance to that of already-implemented planners, such as Dynamic Window Approach (DWA) and Reciprocal Velocity Obstacles (RVO).

As in [14], each scenario is repeated three times. Each time, the simulated crowd is controlled using one of three simulation algorithms (ORCA [15] with 0.5s planning horizon, ORCA 1.5s, and Social Forces [16]).

Given the nature of the crowd flux in this simulation, we posit the hypothesis that a flow-aware planning method would perform as well or better than a non flow-aware equivalent.

In order to test this hypothesis, we run two versions of the FlowBot planner. The first version, *FlowBot (nominal)*, executes the nominal velocity predicted by the cost optimizer directly on the robot as a commanded velocity. This means that there is no local collision avoidance, though the nominal velocity tends to match that of the crowd whenever possible.

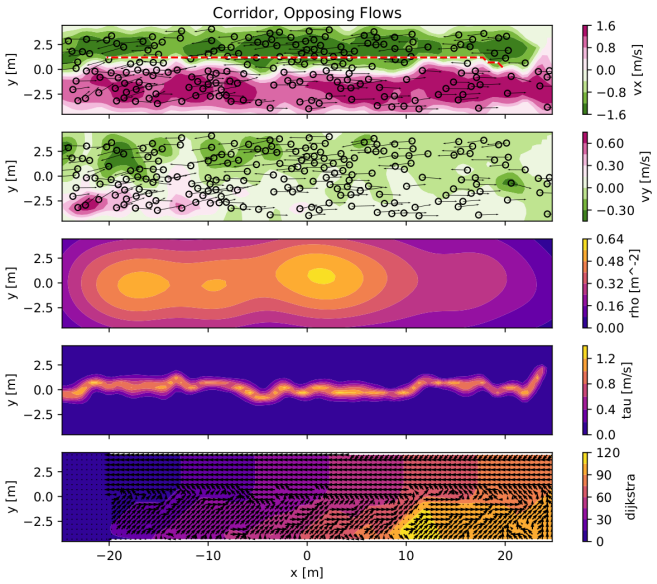

Fig. 6. Visualization of the detections (circles with arrow) state estimate (v_x, v_y, ρ, τ), the resulting Dijkstra cost field superimposed with resulting optimal velocities (bottom), and the planned trajectory (red, dashed), when applying the FlowBot planner to a corridor scenario with opposing flows, in the CrowdBot-Challenge simulator.

We consider that a comparable method to the *baseline* algorithm in the original CrowdBot-Challenge benchmarks, where the robot moves straight towards the goal at all times, as that method also does not perform local collision avoidance.

The second version, *FlowBot+RVO*, does not execute the nominal velocity directly but updates it based on local collision avoidance as dictated by RVO. Therefore, we compare the performance of this planner to that of the RVO algorithm alone, in order to test whether the added flow-aware trajectory planning leads to better outcomes.

We use the crowd flow parameters $\gamma = 1$ and $\sigma = 1$, *without any parameter tuning*. The per-crowd-simulation results are shown in Table I, and mean results overall are visualized in Fig. 7 using the original crowd navigation metrics from [14], where higher is better for the normalized time and collision scores (1.00 would be zero collisions). The results of our flow estimation, and plan, during testing in an example CrowdBot-Challenge scenario with opposite flows is shown in Fig. 6. In summary we find that:

- The *FlowBot+RVO* variant has a higher average success rate, reaches the goal faster, and collides less than pure *RVO* on all crowd-simulators (ORCA 0.5s, ORCA 1.5s, Social Forces). Overall, the success rates are 85% (FB+RVO) vs 75% (RVO), navigation time is 20% shorter for FB+RVO, and collisions are reduced by almost half (55%) for FB+RVO compared to RVO alone. Note that these significant differences in performance are not evident at first glance from the CrowdBot Challenge metrics alone, due to their unintuitive choice of scale. For example, as collision scores are 1 minus the time spent in collision, an improvement from 0.98 to 0.99 implies a 50% reduction in total collision time. Despite this unintuitiveness, we still include the full

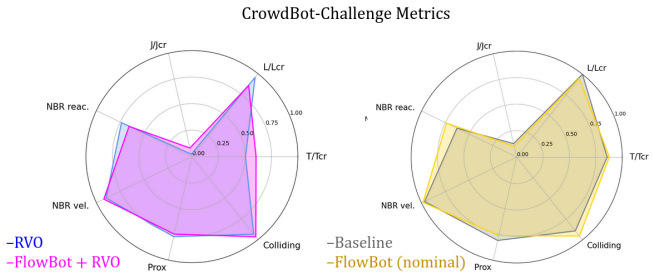

Fig. 7. Radar plot of the metrics obtained from testing the flow-based planner in the CrowdBot Challenge benchmark. Left: results for the FlowBot planner combined with RVO, compared to RVO alone. Right: results for executing the raw FlowBot nominal velocity, compared to the baseline (heading to the goal with constant velocity). In both comparisons, the addition of flow-aware planning allows goals to be reached faster and with fewer collisions. Small differences in the collision metric have a large impact on the total amount of collsions, due to it being an inverse. Here, the FlowBot (nominal) planner scores 0.95 in the collision metric vs 0.89 for the baseline planner, which is a reduction of more than half in total collisions.

CrowdBot Challenge metrics from our experiments in Table I for the sake of completeness.

- The *FlowBot (nominal)* variant has the same average success rate as the *Baseline*. It is slightly slower in the two ORCA Simulations and faster in the Social Forces simulation but importantly collides significantly less than the baseline in all (ORCA 0.5, ORCA 1.5, and Social Forces). Overall, success rate is 100% for both planners, navigation time is 2% shorter for FlowBot.

Overall, the FlowBot (nominal) planner reaches the goal faster than the baseline (which was the fastest planner tested in the previous CrowdBot-Challenge benchmark), while still colliding less on average than RVO (which was previously the safest benchmarked planner).

These results are evidence that the theoretical benefits of flow-aware planning translate to practice.

C. ROS Planner

We speculated earlier that one of the ways in which flow-aware planning can improve efficiency in practice is by detecting and avoiding areas to be avoided entirely due to unfavorable flow characteristics (counter-flow), using our FlowBot modelling, estimation and planning approach to compute a longer-but-more-efficient trajectory around them.

Another particularity of the simulated experiments in Section V-B is the fact that the positions of all people are provided to the planner. This is not realistic, as typical sensors can only detect pedestrians in a limited range around the robot. Therefore, we set out to create a more realistic test of the FlowBot planner, incorporating this partial observability of the crowd, among other things.

Due to unrelated circumstances, we have not had the opportunity to test the approach in real dense crowds yet. Despite this, to qualitatively confirm the potential for flow-aware planning to improve robot navigation in practice, we design a simulated environment which emulates a dense real-life flow typical in for example a train station, with directional crowd flows moving along a corridor. A robot objective is given, where the shortest route is through the counter-flow area.

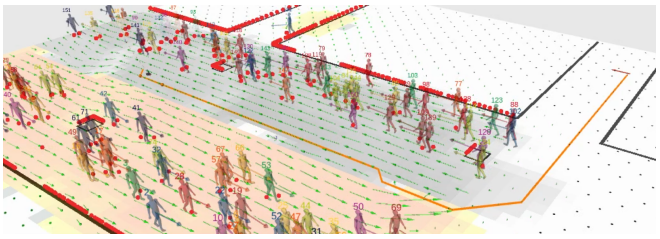

Fig. 8. Qualitative testing of the FlowBot planner in more realistic conditions. Only pedestrians which are in LiDAR range (red dots) are detected. Unlike the ROS planners, the FlowBot algorithm plans a trajectory (orange) which avoids the counter-flow. The flow state estimate is shown as green arrows.

In this experiment, we implement the FlowBot planner as a ROS node, simulate a 2D LiDAR sensor, and use it to provide detections only for the surrounding pedestrians that are not obscured by obstacles or other pedestrians.

We compare the FlowBased planner to the standard ROS navigation stack (Global + Timed Elastic Band (TEB) planner from [17]) on the train-station scenario.

We find that even despite the partial observability of the crowd, as the robot moves the flow model over time becomes a more accurate estimate of the true crowd flux. As a result, the planned trajectory is able to avoid the counter-flow regions. On the other hand, the standard navigation planners push through the counter-moving crowd, incurring high navigation costs such as time, avoidance effort, and increasing the expected collision risk.

The flow-based planning loop is able to run at 10Hz on a Intel i5 CPU with little code optimization. Planning time is mostly proportional to the model grid size (we use a resolution of 0.5m), and the number of samples used in the state estimation (we use a max sample size of 2000).

We make all code publicly available on `github.com/danieldugas/flowbot`.

VI. Conclusions

We consider navigation in crowded environments where the crowd exhibits flow-like behavior, like in a train station. We propose a novel pseudo-fluid based model of crowd flow which has an intuitive physical interpretation and does not require much tuning. We further propose an observation model for this which we validate empirically. Finally, we propose a planner operating on the pseudo-fluid crowd-flow and demonstrate the potential of the approach in simulated navigation scenarios. Despite the models presented in this work being proof-of-concept, with little-to-no parameter-tuning, they achieve state of the art results on the CrowdBot navigation benchmark, and also compare favorably against the standard ROS TEB planner on a partially observable environment, showing how the flow-aware planner successfully estimates and plans around counter-flows in the crowd in real-time. We conclude that flow-based planning shows promise for navigating efficiently in environments that exhibit flow-like behavior, with much potential for further improvement and tuning of the model in future work, to further improve its performance in a variety of scenarios.

TABLE I
CROWDBOT-CHALLENGE METRICS

		Success [%]	Time Metric	Collision Metric	Prox	NBR vel.	NBR reac.	J/Jcr	L/Lcr
ORCA 0.5s	Baseline	100	0.90	0.89	0.81	0.97	0.63	0.13	1.00
	DWA	82.5	0.60	0.92	0.75	0.93	0.80	0.12	0.67
	RVO	75	0.53	0.92	0.78	0.90	0.71	0.03	0.92
	FlowBot (nominal)	100	0.87	0.95	0.77	1.00	0.71	0.05	0.95
	FlowBot + RVO	87.5	0.62	0.96	0.76	0.94	0.67	0.14	0.84
ORCA 1.5s	Baseline	100	0.91	0.89	0.80	0.96	0.58	0.22	1.00
	DWA	87.5	0.58	0.91	0.75	0.90	0.81	0.12	0.67
	RVO	72.5	0.49	0.93	0.76	0.89	0.67	0.02	0.94
	FlowBot (nominal)	100	0.87	0.96	0.74	0.98	0.73	0.07	0.95
	FlowBot + RVO	82.5	0.57	0.97	0.75	0.91	0.63	0.08	0.82
Social Forces	Baseline	100	0.77	0.89	0.81	0.97	0.67	0.03	1.00
	DWA	80	0.59	0.92	0.75	0.90	0.83	0.05	0.72
	RVO	77.5	0.50	0.95	0.78	0.90	0.83	0.02	1.00
	FlowBot (nominal)	100	0.89	0.95	0.77	0.97	0.77	0.19	0.96
	FlowBot + RVO	87.5	0.62	0.96	0.74	0.92	0.67	0.03	0.92

REFERENCES

[1] Z. Chen, C. Song, Y. Yang, B. Zhao, Y. Hu, S. Liu, and J. Zhang, "Robot navigation based on human trajectory prediction and multiple travel modes," *Applied Sciences*, vol. 8, no. 11, p. 2205, 2018.

[2] D. Dugas, J. Nieto, R. Siegwart, and J. J. Chung, "IAN: Multi-behavior navigation planning for robots in real, crowded environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 11 368–11 375.

[3] X. Zhang, Q. Yu, and H. Yu, "Physics inspired methods for crowd video surveillance and analysis: a survey," *IEEE Access*, vol. 6, pp. 66 816–66 830, 2018.

[4] S. Behera, D. P. Dogra, M. K. Bandyopadhyay, and P. P. Roy, "Understanding crowd flow movements using active-langevin model," *arXiv preprint arXiv:2003.05626*, 2020.

[5] N. Bellomo and C. Dogbe, "On the modelling crowd dynamics from scaling to hyperbolic macroscopic models," *Mathematical Models and Methods in Applied Sciences*, vol. 18, no. supp01, pp. 1317–1345, 2008.

[6] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[7] Y. Du, N. J. Hetherington, C. L. Oon, W. P. Chan, C. P. Quintero, E. Croft, and H. M. Van der Loos, "Group surfing: A pedestrian-based approach to sidewalk robot navigation," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6518–6524.

[8] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 981–986.

[9] C. Jiang, Z. Ni, Y. Guo, and H. He, "Learning human–robot interaction for robot-assisted pedestrian flow optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 4, pp. 797–813, 2017.

[10] T. Fan, D. Wang, W. Liu, and J. Pan, "Crowd-driven mapping, localization and planning," in *International Symposium on Experimental Robotics*. Springer, 2020, pp. 354–368.

[11] D. Paez-Granados, V. Gupta, and A. Billard, "Unfreezing social navigation: Dynamical systems based compliance for contact control in robot navigation," *arXiv preprint arXiv:2203.01053*, 2022.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[13] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 489–494.

[14] F. Grzeskowiak, D. J. Gonon, D. Dugas, D. Paez-Granados, J. J. Chung, J. I. Nieto, R. Siegwart, A. Billard, M. Babel, and J. Pettré, "Crowd against the machine: A simulation-based benchmark tool to evaluate and compare robot capabilities to navigate a human crowd," in *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*. IEEE, 2021, pp. 3879–3885. [Online]. Available: https://doi.org/10.1109/ICRA48506.2021.9561694

[15] J. v. d. Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[16] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[17] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, 11 2016.