egoTEB: Egocentric, Perception Space Navigation Using Timed-Elastic-Bands

Justin S. Smith[1], Ruoyang Xu[1], and Patricio Vela[1]

***Abstract*— The TEB hierarchical planner for real-time navigation through unknown environments is highly effective at balancing collision avoidance with goal directed motion. Designed over several years and publications, it implements a multi-trajectory optimization based synthesis method for identifying topologically distinct trajectory candidates through navigable space. Unfortunately, the underlying factor graph approach to the optimization problem induces a mismatch between grid-based representations and the optimization graph, which leads to several time and optimization inefficiencies. This paper explores the impact of using egocentric, perception space representations for the local planning map. Doing so alleviates many of the identified issues related to TEB and leads to a new method called egoTEB. Timing experiments and Monte Carlo evaluations in benchmark worlds quantify the benefits of egoTEB for navigation through uncertain environments.**

I. Introduction

The navigation system of an autonomous mobile robot aims to identify a path connecting the start pose to a desired terminal pose, preferably with good optimality properties. In a perfectly known environment, navigation reduces to planning. As with navigation, the objective of planning is to find a globally optimal solution from start to goal satisfying the robot's motion and control constraints while avoiding collisions. As this is generally too computationally expensive to be performed in real time and unreliable in unknown environments, practical navigation systems utilize a hierarchical approach consisting of an approximate global planner, which uses globally accumulated environmental data to provide coarse globally optimal paths at a low rate, and a local planner, which searches for locally optimal paths that follow the global path. Global planning is generally performed using graph search algorithms such as Dijkstra's [1], A* and variants [2], RRT and variants [3], [4], [5], probabilistic road-maps [6], kino-dynamic planners [7], etc.

Options for the local planner include the aforementioned planning approaches, reactive planners, sample-based planners [8], or optimal trajectory methods [9] applied to a small region around the current robot pose and targeting waypoints on the global path. The timed-elastic-bands (TEB) approach is an effective local planner, first introduced in [10] and iteratively refined over the span of 6 years [11], [12]. It is an optimization-based approach to trajectory planning. Similar approaches include [13], [14], [15], however these have not been integrated into a hierarchical navigation system as TEB has. Key characteristics of TEB are that kinodynamic constraints are considered as part of the planning process; a richer space of trajectory options are output by the optimization in contrast to sample-based methods; and multiple trajectories with distinct topologies are maintained and optimized. The last design element avoids local minima issues by selecting from several trajectory options as needed to ensure that TEB takes the estimated, most optimal path.

TEB uses factor graphs as the underlying representation in order to optimize timed-elastic-bands (*tebs*) with g2o [16], an open source graph-based optimization framework. The graph representation allows TEB to optimize trajectories with respect to a wide variety of soft constraints, represented as edges on the optimization graph. However, this representation introduces a decoupling between the local environment data structure (typically an occupancy grid) and the optimization data structure (a factor graph). The decoupling is a data structure mismatch that negatively impacts the optimization setup and computation time.

Perception space representations in egocentric frames have the potential to improve computational efficiency and navigational performance by exploiting the geometric structure of the obstacle information as measured by on-board sensors. Using perception space enables more efficient collision checking [17], [18]. Perception space also improves the efficiency of trajectory scoring for local navigation [19]. Furthermore, perception space representations have the potential to better align with the graph-based representation employed by TEB's optimization process. We propose to improve the efficiency of TEB by utilizing a perception space obstacle representation as the source of obstacle data.

Moving to an egocentric representation permits the incorporation of concepts from the family of gap-based local planning approaches. Gap-based local planners generally operate on raw laser-scan data, making them perception space approaches for 2D data. Range information implicitly encodes the navigable free space immediately around the robot, which is analyzed for openings or gaps. In general, local navigation is performed by selecting a suitable gap (based on various criteria) and generating commands to steer through it [20], [21], [22]. Rather than using gaps to directly generate navigation commands, we propose to use them as a heuristic to inform more efficient sampling of candidate trajectories residing in distinct topologies.

The main contributions of this paper include (C1) the introduction of the egocircle representation (§III-A) for the local TEB obstacle map, which captures boundaries rather than areas; (C2) the use of a gap-based local navigation

*This work supported in part by NSF Award #1400256 and #1849333.

[1]J.S. Smith, R. Xu, and P.A. Vela are with the School of Electrical and Computer Engineering and the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30308, USA. {jssmith, rxu74, pvela}@gatech.edu

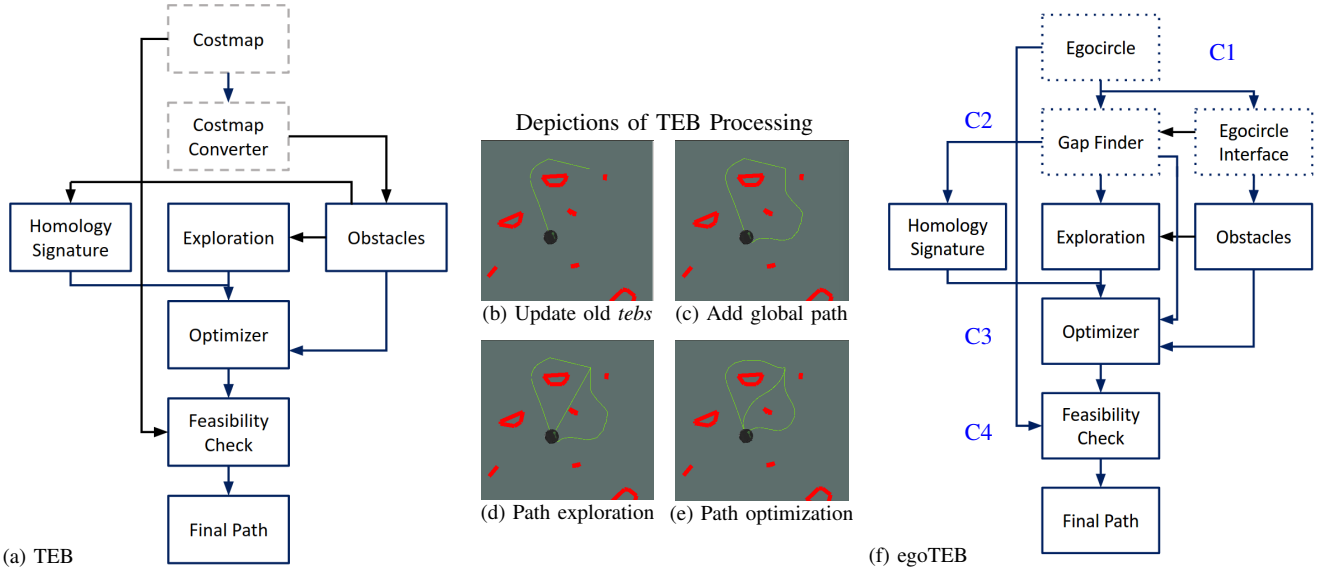

Fig. 1. Block diagrams of TEB approach (a) and proposed egoTEB approach (f), with depiction of TEB calculations (b-e).

scheme (§III-B) to identify distinct local trajectory topologies from visible obstacle boundaries; (C3) a modified gap-based cost (§III-C) to prevent trajectories from jumping from one topology to another during the soft-constraint factor graph optimization; and (C4) a fast, perception space feasibility test (§III-D). Figure 1 depicts the original TEB pipeline (a) and the proposed egoTEB pipeline (f) and serves as a guide to understanding the contents of this paper. The components impacted by the stated contributions are marked.

II. The TEB Optimization Process

TEB is a state-of-the-art approach for optimization based local planning. Trajectories are represented by timed-elastic-bands, or *tebs*. A *teb* consists of a sequence of poses and the time intervals between each consecutive pair of poses. A *teb* is optimized with respect to position and time. In effect, TEB optimization uses a collocation-based approach to optimal trajectory synthesis, with consideration for kinodynamic and collision avoidance constraints, to establish a minimal time path from the robot's initial pose to a final pose [12]. Once the trajectory has been computed, the associated controls are backed out from the trajectory's dynamics. TEB achieves greater robustness by maintaining and optimizing multiple *tebs* during each local planning cycle, allowing it to leave a trajectory caught in a local minimum in favor of a more globally optimal trajectory. This section covers the implementation details of TEB and indicates what problems arise as a function of the internal representations and computations. Then, the following section covers the modifications made to the TEB optimization procedure to arrive at egoTEB.

A. Update obstacle representation

The first step in a new optimization process is to convert the local environment, typically a 2D occupancy grid, into a set of obstacles to populate the factor graph edge cost structure [23]. However, if the number of occupied cells is large, the conversion leads to an excessive number of obstacle points that increase the optimization time (sometimes by several orders of magnitude). One solution is to use a more compact obstacle representation, for which costmap converter components [24] were added to the functionality of TEB to convert a map to a set of points, lines, and/or polygons. Figure 2 depicts a local environment whose occupancy information is given by polygonal elements.

B. Prepare tebs

The preparation process for the *tebs* is depicted in Fig. 1(b-e). First, the start and end poses of any *tebs* from the previous planning cycle are updated with the current g_{robot} and g_{goal}. Next, a new *teb* is created using the current global plan. Finally, additional *tebs* are added by a sampling-based exploration strategy that searches for candidate paths to the goal. The exploration strategy builds a probabilistic roadmap (PRM) for the local space, see Fig. 2, and in the process oversamples the local trajectory space.

A search through this *path graph* may produce many candidate trajectories, though only those with unique H-signatures will be kept. The H-signature (a unique complex number) identifies whether any two paths are of the same homology class. Conceptually, two trajectories belong to the same homology class if one can be warped into the other without crossing over an obstacle (see Fig. 3). The result of this preparation process is a set of topologically distinct *tebs*.

C. Optimize tebs

The next step is to set up and optimize each *teb* using a factor graph. The edge costs in the factor graph act as soft constraints for the included constraints (e.g., vehicle kinematics, obstacles, time optimality, velocity limits, and acceleration limits) leading to a multi-objective optimization with tunable weights.

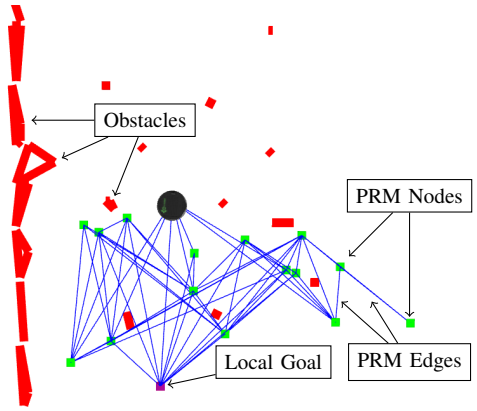

Fig. 2. Path graph produced using PRM approach with polygonal obstacles generated using a costmap converter. Some edges are not helpful.

This optimization does not always yield desirable results, however. As a soft-constraint optimization approach, TEB cannot guarantee that optimized *tebs* will fully satisfy all constraints. Additionally, optimization can cause the poses of a *teb* to jump over an obstacle, changing the *teb*'s H-signature. This *homology jumping* may reduce the number of distinct topologies represented (see Fig. 3).

D. Verify feasibility of 'best' teb

TEB has some flexibility when choosing the 'best' *teb* to follow. Preference may be given to the *teb* selected during the previous planning cycle or to the *teb* initialized with the global plan. Since optimization may not have satisfied all obstacle constraints (see Fig. 4), TEB tests for the feasibility of the 'best' *teb* by checking the first $k_{feasibility}$ poses for collision using the current costmap. If the *teb* is feasible, the trajectory is executed. If not, the planner indicates that it has failed, clears all current *tebs*, and triggers a global replan.

III. egoTEB Implementation

The flowchart of the *egoTEB* system is shown in Fig. 1(f). An egocircle representation replaces the costmap component of TEB, Fig. 1(a). The modified representation structurally changes many of the downstream computations and components while leading to an optimization problem with the same general features as TEB. However, several of the negative characteristics are avoided under the new pipeline.

A. Egocentric Obstacle Representation Using the Egocircle

Moving from a geocentric to an egocentric frame improves the computational performance of maintaining a local obstacle map and aligns the world representation with that of TEB's factor graph approach. As a consequence, conversion costs are avoided and many computational optimizations can be implemented. Perception space representation (a boundary representation) will give better scaling than occupancy grids (an area representation), as will be shown in §IV.

We therefore employ the *egocircle* to store and propagate the most relevent local environment information history in the polar representation [19]. The egocircle data object $\mathcal{L}_{ego}$ stores object boundaries in an ordered circular data structure. Similar to a laser scanner, the egocircle evenly divides the

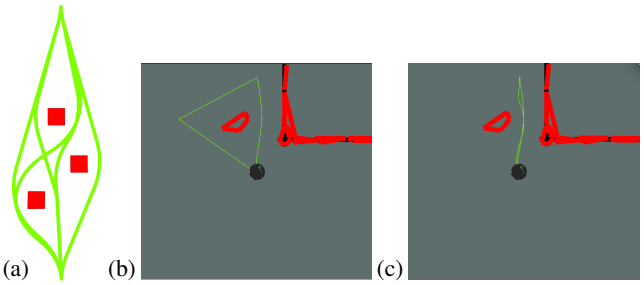

Fig. 3. TEB and homologies. (a) Example of trajectories belonging to different homologies. TEB sometimes starts with (b) trajectories in distinct homology classes prior to optimization, and finishes with (c) some trajectories in the same homology class due to soft constraints.

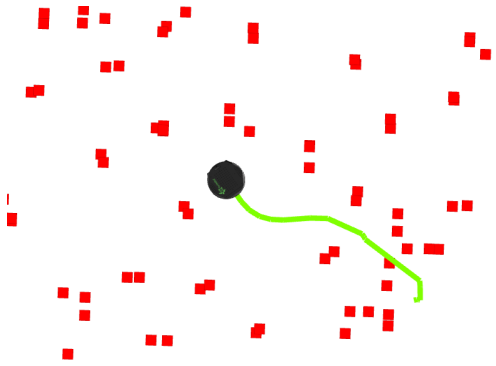

Fig. 4. Example of infeasible trajectory after optimization.

angular space into n_{circ} cells or buckets. As the robot moves, the egocircle points are updated; for $p_{cir} \in \mathcal{L}_{ego}$ and the robot odometry update $g_{move} \in SE(2)$, the transformed point is $p'_{cir} = g_{move} \cdot p_{cir}$ which is the application of the rigid body displacement g_{move} to the planar polar coordinate p_{cir}. The transformed point p'_{cir} gets stored in a (potentially) new bucket within the egocircle. Points with magnitude greater than the egocircle radius R_{max} are discarded. Generating an egocircle scan from stored data entails performing a min operation over all egocircle buckets individually:

$$L_m(i) = \min_{p_{cir} \in \mathcal{L}_{ego}(i)} (\|p_{cir}\|) \tag{1}$$

The process above renders a 1D measurement "image" L_m from all points stored in memory. It is equivalent to a 360° laser scan reading with angular resolution $n_{circ}/(2\pi)$.

The Egocircle Interface decimates egocircle scans to produce a reduced set of point obstacles, serving a similar purpose to TEB's costmap converter. It also generates an inflated egocircle scan, simplifying later processing by allowing the robot to be treated as a point. Conceptually, a circle of radius r_{insc} is placed at the location of each point represented by L_m and a new egocircle scan is generated based on the ranges to these inflated points. The inscribed radius r_{insc} of the robot is used to ensure that the result is liberal (permits false negatives from a collision detection perspective). Figure 5(a) depicts the egocircle scan L_m in black and the corresponding inflated scan L_m^{inf} in gray.

One potential disadvantage of the egocircle is that the min operation destroys beyond-line-of-sight information. The standard TEB implementation's ability to consider non-line-

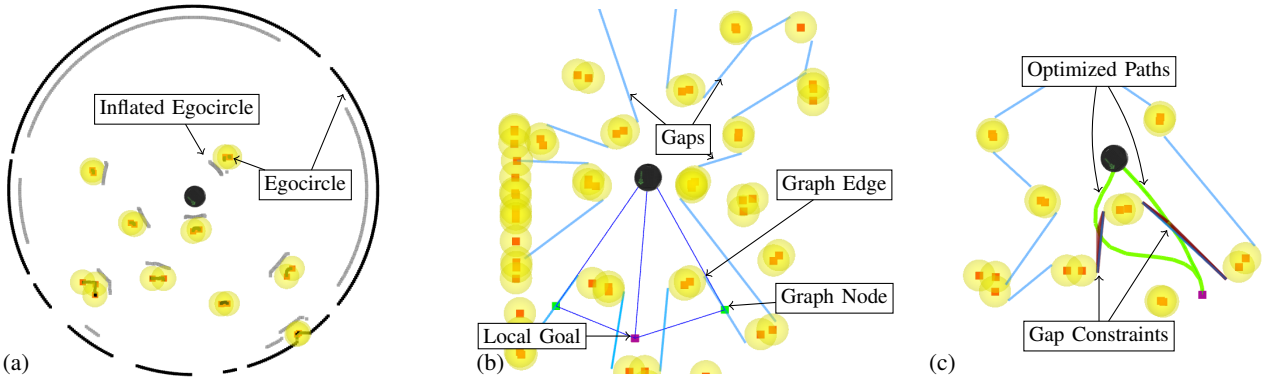

Fig. 5. (a) Egocircle scan (in black) and inflated egocircle scan (grey), with decimated points (red squares). Yellow circles are inflated decimated points. (b) Path graph produced using gap-based approach with decimated egocircle obstacles. (c) Example of scene with 2 *tebs*, each with an attached gap constraint. The local goal point of the global path is denoted by the violet square at the terminal points of the green curves, while the robot is the black circle.

of-sight obstacles using a 2D obstacle representation may allow it to perform better in some scenarios. In order for this ability to be of any use, however, the robot must know of those obstacles. If they are not visible from the robot's current pose, then they must either have been seen earlier or provided as prior obstacle information. In either case, the obstacles would have been added to the global costmap. Since the global plan, produced using this global costmap, is used to initialize a *teb*, the extent of standard TEB's advantage over egoTEB should be minimal.

B. Gap-Based Navigation Graph

In perception space, a trajectory can either stay within the star-shaped, perceived space or exit through a gap, as visualized in Fig. 5(b). Since no information is available regarding the environment beyond the gaps, all trajectories passing through a particular gap are considered equivalent; each gap represents a distinct homology class of trajectories. This section describes how gaps are detected to define a simple navigation graph for seeding trajectory candidates.

1) Egocircle Gaps: Our approach to gap analysis follows that of [25] as applied to the 360 degree inflated egocircle scan. Using the inflated egocircle eliminates the need to remove gaps that are too narrow or within other gaps. Let i be a circular index into the egocircle scan. Candidate regions where a gap will lie occur in the vicinity of elements satisfying one of the following properties:

- Large range difference: $\left|L_m^{\text{inf}}(i) - L_m^{\text{inf}}(i+1)\right| > 2r_{insc}$.
- One range is out-of-range and the other is not: $\left(L_m^{\text{inf}}(i) = R_{max}\right) \oplus \left(L_m^{\text{inf}}(i+1) = R_{max}\right)$

After identifying candidate gap indices in the egocircle, they must be paired up (left edge and right edge) to define the gap regions. Each gap region is represented by a line segment, or *gap segment*, colored light blue in the example of Fig. 5(c).

2) Using Gaps to Determine Distinctiveness: A gap-based homology representation provides an efficient means to maintain trajectories associated to all local homologies with a reduced computational cost relative to the oversampled PRM approach of TEB (Fig. 2). Gap segments found in the robot's frame are transformed to the global planning frame. Define the set of found gap segments $G^P = \{G_i^P\}_{i=1}^{n_{gap}}$, with a segment defined by the two points $G_{i,1}^P, G_{i,2}^P \in \mathbb{R}^2 \ \forall i \in [1, n_{gap}]$. A trajectory is in the homology class of gap i if it crosses the line segment (i.e., passes between the two points).

3) Generating Candidate Trajectories Through Gaps: Generation of trajectories for each homology is significantly simpler with the gap-based approach than with the original sampling-based approach. Trajectory candidates are initialized by creating paths from the robot's current pose through gaps then on to the goal, as well as by going directly from start to goal if it lives within the star-shaped, perceived space. Exploiting this structure greatly reduces the number of possible edges in the created graph. The maximum number of edges in the graph is limited to $2n_{gap}+1$. Furthermore, since straight line paths from the start to a gap are by definition collision free, the number of edges that must actually be evaluated (collision checked) is reduced to $n_{gap} + 1$. In contrast, the total number of edges that must be evaluated in the PRM approach is $n(n-1)$, where n is the total number of points in the graph (sampled points plus start and goal). The impact is improved scalability and reduced computational cost of finding new candidate paths to the goal.

C. Optimization Cost Setup

In an effort to reduce the frequency of homology jumping, we augment the optimization problem with a new gap-based constraint. A gap constraint is added to each trajectory that passes through a gap. The goal of the gap constraint is to prevent trajectories from leaving their respective gaps by penalizing such scenarios. For a given gap G_i^P, define its radius $R_{gap} = \left\|G_{i,1}^P - G_{i,2}^P\right\|/2$ and center $G_{i,c}^P = (G_{i,1}^P + G_{i,2}^P)/2$. The gap cost is based on the distance of the pose from the center of the gap vs. the gap's width:

$$C(d_\alpha) = \begin{cases} \left[\dfrac{d_\alpha - \alpha_{thresh}}{\Delta\alpha}\right]^{2*k_{exp}}, & \text{if } d_\alpha > \alpha_{thresh} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where $d_\alpha(p_{int}) = \left\|p_{int} - G_c^P\right\|/R_{gap}$, p_{int} is the trajectory pose nearest to the intersection of the trajectory with the gap segment, and α_{thresh}, $\Delta\alpha$, and k_{exp} are parameters.

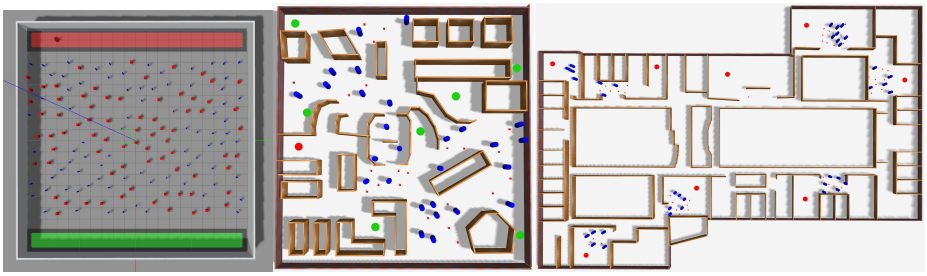

Fig. 6. Benchmark navigation worlds. From left to right: Dense World, Campus World, and Office World. Red and green denote start and goal points, respectively. In Office World, red points may serve as either start or goal points.

D. Perception Space Feasibility Checking

The feasibility checking component of egoTEB is modified to utilize the egocircle rather than a costmap. Since egoTEB does not maintain a local costmap, its feasibility checking component is modified to utilize the egocircle. Pose feasibility is determined using an implementation of the Planning in Perception Space (PiPS) approach to collision checking [18]. The general concept is to perform collision checking by projecting the robot's geometry into the perception space rather than by projecting the sensor data into a costmap. Here, the pose to be evaluated is transformed from the planning frame to the ego frame. Next, the robot's footprint is transformed by the ego frame pose and projected into the egocircle coordinate system. Collision checking compares the robot's footprint against the egocircle values. As long as the range of each point in the footprint is less than the range of the corresponding measurement in the egocircle, the pose is deemed safe. If the opposite holds, then the pose is unsafe. For circular robots, the geometry can be reduced to a point. An unsafe pose for a tested *teb* will result in the *teb* being deemed infeasible.

IV. Experimental Evaluation

This section contains timing test outcomes for the different components of TEB and egoTEB, as well as Monte Carlo outcomes for navigation in benchmark worlds. For the benchmark worlds, randomized obstacles and start/goal points generate a rich set of navigation tests.

A. Benchmarking Environment Setup

The *egoTEB*, TEB, and Dynamic Window Approach (DWA)[8] navigation schemes are run on a series of benchmarking experiments in simulated ROS/Gazebo environments, with the configuration files available at [26]. Benchmark worlds are depicted and labeled in Fig. 6.

Dense World. The dense world is a single large room filled with uniformly placed square and cylindrical posts. Start poses are sampled from a horizontal region near the north wall and place the robot facing inwards. Goal poses are sampled from a horizontal region near the south wall.

Campus World. The campus world is intended to model the outdoor free space of a university campus and consists of several relatively large open areas connected by narrower corridors. There is one starting pose and seven candidate goal poses. A given scenario will randomly select one of these predefined goal poses. Obstacles (consisting of large blue cylinders and small red boxes) are uniformly distributed among the primary open areas of the world.

Office World. The office world is a simplified model of the fourth floor of the building containing our lab. Start and goal poses are randomly selected from a list of locations around the office. A set number of obstacles are randomly placed using the same approach as in Campus World.

The density parameter *min dist* specifies the minimum distance permitted between the centers of any two obstacles that are randomly added to a scenario. The simulated robot platform is a differential drive Turtlebot 2 with a Kobuki base and first generation Microsoft Kinect.

TABLE I

EGOCIRCLE VS COSTMAP UPDATE TIME

	Size Parameter			
Costmap	$5m$	$7m$	$9m$	$11m$
Time	$5.378ms$	$9.355ms$	$11.92ms$	$17.75ms$
Egocircle	$3m$	$4m$	$5m$	$6m$
Time	$0.457ms$	$0.601ms$	$0.776ms$	$0.940ms$

TABLE II

EGOCIRCLE INTERFACE VS COSTMAP CONVERTER

min dist	Egocircle Interface	Costmap Conv (Poly)
0.5	$0.219ms$	$1.653ms$
0.75	$0.219ms$	$0.817ms$
1.0	$0.228ms$	$0.398ms$

TABLE III

EGOTEB VS TEB EXPLORATION

	egoTEB	TEB
min dist	HSig Time per TEB / Total Time / Ave # TEB	
0.5	$33.38\mu s$ / $1.16ms$ / 1.691	$0.8435ms$ / $8.23ms$ / 1.205
0.75	$53.02\mu s$ / $1.62ms$ / 2.286	$0.3192ms$ / $5.62ms$ / 1.558
1.0	$48.09\mu s$ / $1.42ms$ / 1.823	$0.1146ms$ / $2.89ms$ / 1.788

TABLE IV

BUILDGRAPH AND OPTIMIZATION TIME

	egoTEB			TEB
min dist	k_{exp} = 0	k_{exp} = 1	k_{exp} = 10	N/A
0.5	$11.53ms$	$11.87ms$	$11.44ms$	$16.47ms$
0.75	$10.88ms$	$12.96ms$	$12.70ms$	$14.11ms$
1.0	$10.64ms$	$11.63ms$	$12.10ms$	$10.75ms$

B. Computational Efficiency Analysis

The computational benchmarking analysis compares the computational efficiency of various aspects of the approaches. These experiments are performed in the Dense

world with density parameter values $\{0.5, 0.75, 1\}$. Unless otherwise specified, the following parameters are used: $n_{circ} = 512, R_{max} = 3m, r_{insc} = 0.18m, k_{feasibility} = 10, k_{exp} = 2, \alpha_{thresh} = 0.1, \Delta\alpha = 0.5$, costmap size $l_{costmap} = 5m$, costmap resolution $= 0.05m$, costmap converter type is "CostmapToPolygonsDBSConcaveHull" with default parameters. The workstation used is an Intel 3.30GHz i5-4590 Quad core with a single-thread Passmark score of 2115 and a multi-thread score of 7336. The reported results are averages from 4255 planner calls.

1) World Update: This first test compares the update costs associated with assimilating new sensor data into the local world representation. The results are found in Table I. The egocircle representation has a lower baseline cost and lower slope regarding dependence on the local map area. The size parameters are chosen so that the area covered by the costmap and the egocircle are approximately equal: $\pi R_{max}^2 \approx l_{costmap}^2$. The egocircle updates its representation and produces a new egocircle scan every time a sensor measurement is received, whereas the costmap performs updates at a specified frequency (5Hz in all experiments). The fast updating performance of the egocircle makes it more suitable for low latency applications.

2) Data Conversion: The second test evaluates the time cost of translating the local map data for integration into the factor graph. Table II compares egoTEB's Egocircle Interface with TEB's polygonal costmap converter. The egocircle calculations have better scaling properties as a function of obstacle density versus the TEB converter. Both the boundary representation and the line-of-sight only calculations limit the computation's growth.

3) Exploration: Benchmarking results for exploration-related components are shown in Table III. The test measures the time to calculate the homology signature of a single *teb* trajectory, the time for exploring equivalence classes, and the number of *tebs* maintained. egoTEB's exploration strategy takes less time and scales better than TEB's.

4) Optimization: The impact of using an egocentric, perception space representation on the overall factor graph optimization time is quantified in Table IV. The different egoTEB columns provide results for several values of k_{exp}. Here, the compute times of egoTEB are similar to those of TEB, but with better scaling as a function of world density.

5) Feasibility Checking: We benchmarked the time taken for PiPS to collision check trajectories and compared against the default collision checking algorithm utilizing traversability cost. Table V shows the average run-time of both representations. PiPS consistently performs feasibility checking in less than 1/2 of the time used by geocentric methods. The run-times for both algorithms scale linearly for this component.

6) Total Time Cost: In the end, the egoTEB representation results in a local navigation planning algorithm with lower run-time costs and favorable scaling properties as a function of world complexity. See Table VI for total run-times from sensor measurement to navigation decision.

TABLE V

COLLISION CHECKING

	$k_{feasibility}$				
	5	8	11	14	17
PiPS	15.42	18.47	22.30	23.87	27.20
Costmap	31.70	42.19	52.62	60.84	67.46

TABLE VI

TOTAL PLANNING TIME

min dist	egoTEB	TEB
0.5	$11.76ms$	$19.89ms$
0.75	$12.78ms$	$20.60ms$
1.0	$12.33ms$	$17.10ms$

TABLE VII

NAVIGATION TEST

	egoTEB	TEB	DWA
	Success Rate/Path Length(m)/Execution Time(s)		
0.5	79.5% / 28.3 / 86.7	59.5% / 26.7 / 76.7	10.5% / 29.4 / 97.9
0.75	100% / 21.9 / 49.2	99% / 21.8 / 51.3	94% / 22.6 / 59.9
1.0	100% / 19.8 / 41.7	99.5% / 19.8 / 42.2	100% / 20.3 / 47.0
Campus	100% / 22.6 / 46.1	93% / 22.7 / 49.4	99% / 23.3 / 49.5
Office	98% / 47.7 / 96.4	98% / 47.7 / 97.4	100% / 48.7 / 100.6

C. Navigation Performance

Lastly, the performance outcomes for the different navigation methods are given in Table VII. Navigation benchmarks are run 4 at a time on an Intel Xeon E5-2640 @ 2.50GHz processor with a single-thread Passmark score of 1468 and a multi-thread score of 14649. Performance is measured as the average final executed path length, path execution time and navigation task completion rate over 200 runs. The baseline methods do well for all worlds except the most dense one. For all other worlds egoTEB matches or outperforms the original TEB. For the most dense, egoTEB has the best performance, suggesting that egocentric perspectives may be of value in dense environments. Overall, the outcomes indicate that egoTEB successfully translated TEB to a more egocentric, perception space model of operation for the local planner.

V. CONCLUSION

This paper described a modification to the Timed-Elastic-Band (TEB) navigation system whose primary characterization is that of modifying the internal, local planning representation from being world-centric to being egocentric. Exploring the numerical and computational issues associated to TEB and their negative consequences provides guidance on how to incorporate the egocentric representation with the explicit goal of remedying these issues. The final egoTEB implementation nicely unifies alternative local navigation strategies and modern sensor-based navigation methods for unknown environments. The net result is a more efficient implementation with equal or improved collision avoidance properties that also shows significant potential for low computational power devices. This implementation is open source software [27]. As future work, it would be interesting to explore how egoTEB might improve TEB for the case of moving objects.

REFERENCES

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[3] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Technical Report 98-11, Computer Science Department, Iowa State University, 1998.

[4] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 995–1001, vol.2.

[5] 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.

[6] L. E. Kavraki, M. N. Kolountzakis, and J. . Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 166–171, 1998.

[7] R. B. Rusu, I. A. Şucan, B. Gerkey, S. Chitta, M. Beetz, and L. E. Kavraki, "Real-time perception-guided motion planning for a personal robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 4245–4252.

[8] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23–33, 1997.

[9] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *International Conference on Robotics and Automation*, 1993, pp. 802–807, vol.2.

[10] C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *7th German Conference on Robotics*, 2012, pp. 1–6.

[11] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Efficient trajectory optimization using a sparse model," in *European Conference on Mobile Robots*, 2013, pp. 138–143.

[12] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[13] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

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

[15] M. Mukadam, X. Yan, and B. Boots, "Gaussian process motion planning," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 9–15.

[16] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

[17] L. Matthies, R. Brockers, Y. Kuwata, and S. Weiss, "Stereo vision-based obstacle avoidance for micro air vehicles using disparity space," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 3242–3249.

[18] J. S. Smith and P. Vela, "PiPS: Planning in perception space," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 6204–6209.

[19] J. S. Smith, S. Feng, F. Lyu, and P. Vela, "Real-time egocentric navigation using 3d sensing," in *Machine Vision and Navigation*, O. Sergiyenko, W. Flores-Fuentes, and P. Mercorelli, Eds. Springer, pp. 431–484, 2020.

[20] J. Minguez and L. Montano, "Nearness diagram navigation (ND): a new real time collision avoidance approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2000, pp. 2094–2100.

[21] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: "Follow the Gap Method"," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1123–1134, 2012.

[22] M. Mujahed, D. Fischer, and B. Mertsching, "Tangential Gap Flow (TGF) navigation: A new reactive obstacle avoidance approach for highly cluttered environments ," *Robotics and Autonomous Systems*, vol. 84, pp. 15–30, 2016.

[23] C. Rösmann. (2019) teb_local_planner - ros wiki. [Online]. Available: http://wiki.ros.org/teb_local_planner.

[24] ——. (2019) costmap_converter - ros wiki. [Online]. Available: http://wiki.ros.org/costmap_converter.

[25] M. Mujahad, D. Fischer, B. Mertsching, and H. Jaddu, "Closest Gap based (CG) reactive obstacle avoidance Navigation for highly cluttered environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1805–1812.

[26] J. S. Smith, J. Hwang, and P. Vela, "Benchmark worlds for testing autonomous navigation algorithms," 2018, [Repository]. [Online]. Available: http://github.com/ivalab/NavBench.

[27] J. S. Smith, "egoTEB Source Code", 2020, [Repository]. [Online]. Available: http://github.com/ivaros/egoTEB.