Blending of Human and Obstacle Avoidance Control for a High Speed Mobile Robot

Justin G. Storms, Student Member, IEEE, and Dawn M. Tilbury, Fellow, IEEE

Abstract—Humans remain in the loop in teleoperation because they have some knowledge that the robot they are controlling does not. At the same time teleoperated robots can be programmed to be very good at many tasks, such as avoiding obstacles. Therefore, sharing control between human and semi-autonomous behaviors on a robot has great potential. This paper presents a model predictive control (MPC) shared control framework for blending human inputs with autonomous behavior inputs. This work adds consideration of how the human input differs from that of an autonomous controller in addition to threat of collision. The framework is applied to a high speed differential drive robot moving through an obstacle field. Preliminary tests by the authors compared the MPC shared control framework to switching obstacle avoidance on/off and the proposed MPC shared control gives the human up to 26% more control with a 35% reduction in collision penalty. Compared to pure human control, MPC shared control demonstrated a 66% reduction in collision penalty. Results show promise for increased user control with better performance.

I. INTRODUCTION

Despite the fact that a growing number of tasks can be performed autonomously by robots, there are still many scenarios where human judgment is required or preferred [1]. In particular most search and rescue missions require a human remain in the loop with the teleoperated mobile robot. However, many current teleoperated mobile robots in search and rescue are not easy to use for the human operating them [2]. Operating in pure teleoperation is often too slow in time critical missions. Thus, assistive semi-autonomous behaviors have been considered for teleoperated robots.

Assistive behaviors on mobile robots such as cruise control, obstacle avoidance, and rollover prevention have all been well researched and implemented extensively [3], [4], [5]. The interaction of human inputs and inputs from assistive behaviors has also been researched extensively [6], [7], [8], [9], [10]. Many methods of handling human inputs and assistive behavior inputs have been developed ranging from simply switching control between the human and semi-autonomous behavior to more advanced blending of human and semi-autonomous behavior inputs.

Human judgment is often required in teleoperation scenarios because there is some ability the human has that the robot is not programmed for. Giving preference to the human operator for high speed mobile robots without sacrificing safety and performance is a challenge. This paper presents a model predictive control (MPC) shared control framework that blends human and obstacle avoidance controller inputs by considering human preference and threat of a collision for a high speed mobile robot.

A brief background of previous work done in shared control of vehicles is discussed in Section II. The proposed shared control framework is given in Section III. A description of the robot model and obstacle avoidance behavior used is given in Section IV. Simulation setup and results are given in Section V. Finally, conclusions are in Section VI.

II. BACKGROUND

Combining human and semi-autonomous behavior inputs has been extensively explored. A brief overview of some of the research related to this work follows. Desai and Yanco present a sliding scale autonomy system that allows users to adjust the level of autonomy of a mobile robot on the fly [11]. This system blends the autonomous input with the human input by a linear scaling factor that is set by the user. As the navigation scenario changes the operator has to know what level to manually set the blending to, which can be challenging and cumbersome.

More advanced work has been done in shared control for wheelchair navigation. Goil et al. developed a machine learning controller that blends with human input to assist a wheelchair operator in navigating through a doorway [6]. Vanhooijdonck et al. present a framework for shared control in wheelchair navigation systems where the path is chosen based on user intention [7]. Philips et al. present an adaptive shared control framework that allows for control of a wheelchair through a brain-computer interface with automatic collision avoidance, obstacle avoidance, and orientation recovery [8]. However since these systems consider wheelchair navigation, all are at low speed operation.

Iagnemma’s group has done extensive work in the area of shared control (semi-autonomous control) for hazard avoidance in vehicles. In [9] they develop a model predictive control (MPC) based framework that blends human and hazard avoidance controller inputs based on threat of collision. They extend this work in [12] to include stability control. In [13] they approach the semi-autonomous hazard avoidance control in a new way such that they consider “homotopies” (or regions) where there are no hazards rather than considering only a single optimal path. In each of these
systems the semi-autonomous control ultimately overrides
the human once it thinks the threat is too large.

Enes and Book have developed a shared control framework
that gives authority to the human operator when his/her
inputs differ greatly from that of the automatic controller
or if they are far away from the goal [10]. However, when
deciding how to blend the human and automatic controller
inputs they only consider the current input at that moment.

The shared control method proposed in this paper falls
between the work of Ignemma’s group and Book’s group.
The automatic controller is assumed not to be perfect and
an optimal blending of human and automatic controller is
calculated based on how the inputs differ and the threat.

III. CONTROL BLENDING METHOD

In order to present the proposed model predictive control
(MPC) shared control framework, a high speed mobile robot
navigating through an obstacle field is considered. Many mo-
obile robot tasks such as search and rescue or reconnaissance
missions require a human remain in the loop to make certain
cjudgments. The robot may have an autonomous behavior that
makes it better at avoiding obstacles than the human operator,
but the human operator has a better grasp of the mission
goal. Therefore, it is desirable to blend inputs to the robot
that work towards accomplishing the goal the human has in
mind (searching for something the robot cannot sense) and
the robot’s autonomous behavior goal (avoiding obstacles).

In order to blend human and autonomous controller inputs,
this paper proposes to use a MPC blending method. The
overall structure is displayed in Fig. 1. The proposed MPC
blending will select a scaling factor \( \alpha \) between 0 and 1
by minimizing a cost function (that will be discussed later)
defined by the designer of the blending system. The scaling
factor \( \alpha \) will be used to calculate the input \( u \) given to the
robot as shown in (1). The variable \( u_h \) represents the input
to the system from the human, while \( u_a \) represents the input
to the system from an autonomous controller.

\[
u = \alpha u_h + (1 - \alpha) u_a \tag{1}
\]

The cost function will sum over the interval from \( j = 1, \ldots, n \). This interval corresponds to the discrete time pre-
diction horizon. The prediction horizon is defined to be \( T_p \)
in seconds. The discrete time step used over \( T_p \) is \( dt \) in seconds.
Thus \( n \) will be the number of time steps \( dt \) required to predict
\( T_p \) seconds ahead from the current time. The number of time
steps is \( n = T_p / dt + 1 \). For simplicity, it is assumed that
the control horizon \( T_c \) used for this model predictive control will
be the same as the prediction horizon \( T_p \).

Another assumption made is that the human input \( u_h \)
remains constant for the entire prediction horizon. In other
words \( u_h = u_{h,j}, j = 1, \ldots, n \). This is a reasonable assump-
tion for small control horizons (\( T_c \ll 0.25 \) s). However, this
method could be modified to adjust the human input over the
control horizon (e.g. the rate of change of human input at the
current time could be projected over the control horizon).

The cost function should be defined such that it leverages
the strengths of both the human and autonomous controllers.
If a human teleoperator is giving a particular control input,
it is likely because they have some knowledge or capability
that the autonomous robot behavior does not. Therefore, if
their input differs significantly from that of the autonomous
behavior, the human should be given some preference. Fur-
thermore, if the robot has autonomous behaviors that are
good at identifying obstacles and avoiding them, then the
robot should be given more preference when it detects that
the human is in danger of running into an obstacle. Many
cost functions that capture this behavior when minimized
could be created. The one proposed in this paper is (2).

\[
J = k_1 \sum_j^n \Phi_j + k_2 \sum_j^n |u_h - u_j| \tag{2}
\]

The term \( \Phi_j \) represents the threat of an obstacle collision
and is calculated at each time step \( dt \) over the prediction
horizon \( T_p \). These threats \( \Phi_j \) are summed up over the length
of the prediction horizon \( n \) and multiplied by weighting \( k_1 \).
The absolute values of the differences between the human
input \( u_h \) and predicted input to the robot \( u_{a,j} \) represent the
cost associated with the autonomous behavior disagreeing
with the human. These differences are summed over the the
prediction horizon \( n \) and multiplied by weighting \( k_2 \).

The second line of (2) results from substituting (1) into the
first line. This is shown to give some more insight to the
effect of \( \alpha \) on the cost function. Looking at the cost function,
one can see that increasing the value of \( k_1 \) will give more
weighting to minimizing the threat of an obstacle collision.
While increasing the value of \( k_2 \) will give more weighting
to keeping the human and autonomous behavior inputs close.

The general form of the MPC blending method enclosed
by the dashed red line in Fig. 1 is described in Algorithm 1.
In summary, MPC is used to determine the blending scalar
\( \alpha \). Based on the robot’s current state \( y \), the current human
input \( u_h \) and the current plus future autonomous behavior
inputs \( u_{a,j} \), the future states of the robot \( y_{j+1} \) are predicted
using a model of the robot with a particular \( \alpha \). The \( \alpha \) that
minimizes the specified cost function \( J \) is output from the
MPC blending method and used to determine the actual input
to the robot as specified in (1).

IV. MODEL AND CONTROLLER DESCRIPTION

In order to test the proposed blended shared control
framework a few simple models will be used in simulation.
The robot model and obstacle avoidance method used are
described in the following sections. In addition the specifics
for the MPC blending method are given.
function MPC BLending\((u_h, u_a, y)\)
\[
\min_{\alpha \in \{0, 1\}} \text{Cost Function}\(\alpha, u_h, u_a, y\)
\]
function Cost Function\(\alpha, u_h, u_a, y\)

Inputs:
\[\alpha: \text{blending scalar}\]
\[u_h: \text{current human controller input}\]
\[u_a: \text{current autonomous controller input}\]
\[y: \text{output of robot states}\]
\[y_1 = y\]
\[n = T_p/dt + 1\]

for \(j = 1\) to \(n\) do
\[
\Phi_j \leftarrow \text{calculate } \Phi \text{ at } y_j
\]
\[
u_{a,j} \leftarrow \text{calculate } u_a \text{ at } y_j
\]
\[
u_j = \alpha u_h + (1 - \alpha) u_{a,j}
\]
\[y_{j+1} \leftarrow \text{apply } u_j \text{ for } dt \text{ to robot model at } y_j
\]
end for
\[
J = \sum_{j=1}^{n} \Phi_j + k_2(1 - \alpha) \sum_{j=1}^{n} |u_h - u_{a,j}|
\]
return \(J\)
end function

Algorithm 1: Model Predictive Conrlo Blending Method

A. Robot Model

The simple differential drive robot with width \(w\) and length \(l\) shown in Fig. 2 was considered. The kinematic equations for this differential drive robot are the following.
\[
\begin{align*}
\dot{x} &= V \cos(\phi) \\
\dot{y} &= V \sin(\phi) \\
\dot{\phi} &= \alpha u_h + (1 - \alpha) u_a
\end{align*}
\]
(3)

The forward velocity, \(V\), is assumed to be constant. The inputs to the system are the turning rate input by the human operator \(u_h = \dot{\phi}_h\), and the turning rate calculated by the obstacle avoidance controller \(u_a = \dot{\phi}_a\). As previously stated, \(\alpha \in [0, 1]\). In the case where \(\alpha = 1\) the human operator has full control of the robot turn rate and when \(\alpha = 0\) the obstacle avoidance controller has full control. Calculation of \(\dot{\phi}_a\) is discussed next.

B. Obstacle Avoidance Method

Many obstacle avoidance methods exist such as potential field [3] and vector histogram [4] methods. Since many mobile robots are not omnidirectional, nonholonomic constraints usually must be considered when using obstacle avoidance methods to find a safe path. However, the steering potential function obstacle avoidance method developed by Huang et al. can be used without considering nonholonomic constraints when applied to a differential drive robot [14]. The steering potential function was selected for this shared blended control framework because of its effectiveness and simple integration with a differential drive mobile robot. However, another obstacle avoidance and path planning method could readily be swapped out into this framework.

Detailed description of the steering potential function obstacle avoidance method, including description of constants used, can be found in [14]. The major difference between the brief description of the method here and in the paper by Huang et al. is omission of the velocity control term. Velocity control was omitted in order to simplify the analysis to a single input system for this first demonstration of the framework, but will be considered in future work. The obstacle potential field, \(\Phi_a\), is given in (4) where \(i = 1, ..., p\) for \(p\) obstacles. Parameter \(c_3\) represents the “gap shooting” behavior as described in [14]. Refer to Fig. 2 for description of obstacle angle, \(\psi_{oi}\), and obstacle angular width, \(\theta_{oi}\).
\[
\Phi_a = \sum_{i}^{p} k_o c_3|\phi - \psi_{oi}| + 1 \left( e^{-c_3|\phi - \psi_{oi}|} \right)
\times (\tan(\theta_{oi} + c_5) - \tan c_5)
\]
(4)

The obstacle angular width, \(\theta_{oi}\), is calculated according to (5), where \(r_{oi}\) is the radius of obstacle \(i\) and \(d_{oi}\) is the distance from the center of the robot to center of obstacle \(i\).
\[
\theta_{oi} = 2 \tan^{-1} \left( \frac{r_{oi}}{d_{oi}} \right)
\]
(5)

The parameter \(c_5\) is calculated as shown in (6). Note that the value of \(c_5\) depends on obstacle radius \(r_{oi}\) and robot radius \(r_r\). In the case of a rectangular robot a conservative estimate of the robot radius can be \(r_r \approx \sqrt{(l/2)^2 + (w/2)^2}\).
\[
c_5 = \frac{\phi}{2} - 2 \tan^{-1} \left( \frac{r_{oi}}{r_{oi} + r_r} \right)
\]
(6)

The steering angular acceleration is calculated based on the gradient of the obstacle potential function minus an angular damping constant, \(b\), as shown in (7).
\[
\ddot{\phi}_a = \frac{d\Phi_a}{d\phi} - b \dot{\phi}
\]
(7)

The resulting angular acceleration steering potential function is shown in (8).
\[
\ddot{\phi}_a = -b \dot{\phi} + \sum_{i}^{p} k_{oi} (\phi - \psi_{oi}) \left( e^{-c_3|\phi - \psi_{oi}|} \right)
\times (\tan(\theta_{oi} + c_5) - \tan c_5)
\]
(8)
C. Shared Control - Model Predictive Control Method

The overall structure of the shared control approach is displayed in Fig. 3. The Diff Drive Robot block is described by (3). The Obstacle Avoidance Controller block is described by (8). The Model Predictive Controller (MPC) block outputs the scalar blending factor $\alpha$. The workings of the MPC block are summarized in Algorithm 1 and will be detailed below.

The MPC finds an optimal $\alpha$ based on the current output of the robot $y$, the human input $\phi_h$, and obstacle location. The robot states, obstacle potential field $\Phi_a$, and obstacle avoidance controller $\phi_o$ outputs are calculated over prediction horizon $T_p$, based on the inputs over control horizon $T_c$. From these model predictions the blending scalar $\alpha$ is selected by minimizing the cost function $J$. The cost function in (2) applied to this specific case is displayed in (9).

$$J = k_1 \sum_j \Phi_a + k_2 \sum_j |\phi_h - \phi| = k_1 \sum_j \Phi_a + k_2(1 - \alpha) \sum_j |\phi_h - \phi_a|$$  \hspace{1cm} (9)

Note that in this case threat level is defined to be the obstacle potential field ($\Phi_a$). As the robot moves closer to obstacles the potential field increases. Thus, by minimizing the cost function, the obstacle potential field decreases resulting in an $\alpha$ value that will blend the inputs in a way that keeps the robot away from obstacles, while still considering user preference.

Algorithm 1 displays the overall structure of this constrained nonlinear optimization problem. With the help of an optimization solver, it is possible to determine the shared control blending scalar $\alpha$ that will balance human control authority and obstacle threat level.

V. SIMULATION TESTS AND RESULTS

The proposed shared control framework was implemented in MATLAB to evaluate its effectiveness. A simulation of a differential drive robot moving through an obstacle field was created. The values of the parameters used are summarized in Table I. The user had a live overhead view of the simulated robot (see Fig. 4) and interacted with the robot through a Logitech F710 wireless gamepad. Only the authors tested the simulation. As the results will show, the proposed method appears promising and does justify future user tests requiring IRB approval.

A. Simulation Setup

The robot forward velocity was kept constant and the user’s only input was robot turn rate $\phi_h$ as they traveled through a set of randomly generated obstacle. Further details of the simulation environment setup are discussed below.

1) Obstacle Field Generation: Obstacle fields like that in Fig. 4 were generated pseudo-randomly with consideration for a few rules. Each obstacle field was generated in a 10 m by 10 m section. Walls were defined to extend from top to bottom along the left and right sides of the section. Each section contained 8 obstacles with diameters specified by the set $D_o = \{1.5, 1.5, 1.5, 2.0, 2.0, 2.5, 3.0, 4.0\}$ m. The location of the center of each obstacles was assigned in a random order. However, the outer edge of each obstacle had to be at least 0.9 m away from the walls and all other obstacle’s outer edges. The one exception was that the center of an obstacle could lie on the edge of a wall.

Note that 0.9 m was chosen as the separation distance between obstacles because in initial simulation testing this was the smallest reasonable distance to navigate the robot through. A series of 10 sections were then stacked end-to-end to complete a single track 10 m wide by 100 m long. Using this method a set of 5 tracks was generated for the simulation tests.

2) Comparison Methods: The simplest method of control for the robot is 100% control by the human operator. In other words $\alpha = 1$, which makes $\dot{\phi} = \phi_h$ in (3). This scenario will be referred to as “OAC Off”.

One method of combining the human and obstacle avoidance controller inputs is by switching between human control ($\alpha = 1$) and obstacle avoidance control ($\alpha = 0$). This has been done in the automotive industry for collision mitigation systems - when the probability of collision reaches some threshold an automatic controller takes over [15].

A similar switching method is used for comparison in this paper. The obstacle potential $\Phi_o$ will be calculated during each time step of the simulation. While $\Phi_o$ is below a threshold $\Phi_t$ the human will have control ($\alpha = 1$). Once $\Phi_o$ exceeds $\Phi_t$ the obstacle avoidance controller will take over ($\alpha = 0$). This input method will be called “OAC Switch”.

3) Simulation Test Description: A total of 60 tests were run with the authors as users in the MATLAB simulation. The user tested all of the following combinations of test cases:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell$</td>
<td>0.3 m</td>
<td>robot length</td>
</tr>
<tr>
<td>$w$</td>
<td>0.2 m</td>
<td>robot width</td>
</tr>
<tr>
<td>$r_r$</td>
<td>0.18 m</td>
<td>approximated robot radius</td>
</tr>
<tr>
<td>$b$</td>
<td>5.5 *sec$^{-1}$</td>
<td>angular rate damping</td>
</tr>
<tr>
<td>$c_3$</td>
<td>4.0 (unitless)</td>
<td>“gap shooting” behavior</td>
</tr>
<tr>
<td>$k_{o_r}$</td>
<td>5000 *sec$^{-2}$</td>
<td>obstacle repulsion</td>
</tr>
<tr>
<td>$k_1$</td>
<td>0.05 sec$^2$</td>
<td>obstacle potential weighting</td>
</tr>
<tr>
<td>$k_2$</td>
<td>0.95 sec</td>
<td>human input weighting</td>
</tr>
<tr>
<td>$T_p$</td>
<td>0.20 sec</td>
<td>prediction horizon</td>
</tr>
<tr>
<td>$T_c$</td>
<td>0.20 sec</td>
<td>control horizon</td>
</tr>
<tr>
<td>$dt$</td>
<td>0.05 sec</td>
<td>prediction time step</td>
</tr>
<tr>
<td>$\Phi_t$</td>
<td>300 *sec$^{-2}$</td>
<td>obstacle potential threshold</td>
</tr>
</tbody>
</table>

Fig. 3. Diagram of shared control blending between human inputs, $\dot{\phi}_h$, and obstacle avoidance controller inputs, $\dot{\phi}_a$. 

Fig. 4. The location of the center of each obstacles was assigned in a random order. However, the outer edge of each obstacle had to be at least 0.9 m away from the walls and all other obstacle’s outer edges. The one exception was that the center of an obstacle could lie on the edge of a wall.

TABLE I

**OBSTACLE AVOIDANCE, BLENDING, AND ROBOT PARAMETER VALUES.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell$</td>
<td>0.3 m</td>
<td>robot length</td>
</tr>
<tr>
<td>$w$</td>
<td>0.2 m</td>
<td>robot width</td>
</tr>
<tr>
<td>$r_r$</td>
<td>0.18 m</td>
<td>approximated robot radius</td>
</tr>
<tr>
<td>$b$</td>
<td>5.5 *sec$^{-1}$</td>
<td>angular rate damping</td>
</tr>
<tr>
<td>$c_3$</td>
<td>4.0 (unitless)</td>
<td>“gap shooting” behavior</td>
</tr>
<tr>
<td>$k_{o_r}$</td>
<td>5000 *sec$^{-2}$</td>
<td>obstacle repulsion</td>
</tr>
<tr>
<td>$k_1$</td>
<td>0.05 sec$^2$</td>
<td>obstacle potential weighting</td>
</tr>
<tr>
<td>$k_2$</td>
<td>0.95 sec</td>
<td>human input weighting</td>
</tr>
<tr>
<td>$T_p$</td>
<td>0.20 sec</td>
<td>prediction horizon</td>
</tr>
<tr>
<td>$T_c$</td>
<td>0.20 sec</td>
<td>control horizon</td>
</tr>
<tr>
<td>$dt$</td>
<td>0.05 sec</td>
<td>prediction time step</td>
</tr>
<tr>
<td>$\Phi_t$</td>
<td>300 *sec$^{-2}$</td>
<td>obstacle potential threshold</td>
</tr>
</tbody>
</table>


Four speeds: \( V = 2, 4, 6, 8 \) \( \text{m/sec} \)

Five tracks: track1, track2, track3, track4, track5

Three input control methods:

1) Pure human control - “OAC Off” (\( \alpha = 1 \))
2) Switching between human and obstacle avoidance controller - “OAC Switch” (\( \alpha = 0 \lor \alpha = 1 \))
3) MPC blending of human and obstacle avoidance controller - “OAC Blend” (\( \alpha \in [0, 1] \))

For each of the different test speeds the minimum turning radius of the robot was kept constant at \( R = 1 \) \( \text{m} \). Therefore, the maximum allowable turn rate of the robot at \( V = 2 \) \( \text{m/sec} \) was \( 2 \) \( \text{rad/sec} \), at \( V = 4 \) \( \text{m/sec} \) it was \( 4 \) \( \text{rad/sec} \), etc. While this is generally not the case for a single robot traveling at different speeds, it more closely mimics different robots that each have different maximum speeds.

The tests were performed in the following order. All of the tests for each speed were performed before moving onto the next speed, i.e. all of the tracks and input control methods were tested at \( V = 2 \) \( \text{m/sec} \) before moving onto \( V = 4 \) \( \text{m/sec} \). The test speeds were conducted from slow to fast, i.e. all tests at \( V = 2 \) \( \text{m/sec} \) were performed, followed by \( V = 4 \) \( \text{m/sec} \), and so on. The order that input control methods were tested was randomized, i.e. for \( V = 2 \) \( \text{m/sec} \), track1 the order was “OAC Switch”-“OAC Blend”-“OAC Off”, for \( V = 2 \) \( \text{m/sec} \), track2 the order was “OAC Blend”-“OAC Off”-“OAC Switch”, etc.

The collision penalty (coll. pen.) was defined to be the weighted sum of all of the collisions for a given run by the user on a track. Collisions were categorized as minor collisions (min. coll.) or major collisions (maj. coll.). Minor collisions were defined to be when the robot passes over an obstacle and only one side of the robot (left or right) intersects the obstacle. Major collisions were defined to be when the robot passes over an obstacle and both sides of the robot (left and right) intersect the obstacle. Minor collisions were assigned a weighting of 1, while major collisions were assigned a weighting of 2.

\[
\text{coll. pen.} = 1 \cdot (\text{no. min. coll.}) + 2 \cdot (\text{no maj. coll.})
\]

4) Obstacle Avoidance Controller Parameters: The parameters for the obstacle avoidance were set primarily using the values given in [14]. The only parameter set differently was \( k_{\text{eq}} \). Increasing this parameter was necessary to compensate for the removal of the velocity reduction control term. This parameter was “hand-tuned” to \( k_{\text{eq}} = 5000 \text{ sec}^{-2} \) to cause the obstacle avoidance method to provide sufficiently large turning rates and avoid obstacles before collision.

The obstacle avoidance method selected does is explicitly defined for flat walls. Therefore, the left and right walls are approximated by circular obstacles of diameter 1.0 \( \text{m} \) that have \( x \)-positions of \(-5.5 \text{ m} \) and \( 5.5 \text{ m} \) respectively. The \( y \)-position of these “wall obstacles” are always 0.5 \( \text{m} \) greater than the robot’s current \( y \)-position. In other words the “wall obstacles” act as moving circular obstacles that always stay on the wall ahead of the direction the robot is traveling.

5) Model Predictive Controller Parameters: The prediction and control horizon for all test cases were selected to be \( T_p = 0.20 \text{ sec} \) and \( T_c = 0.20 \text{ sec} \), respectively. The time step was set to \( dt = 0.05 \text{ sec} \). These horizons are of similar length to the reaction time of the user.

The weighting factors \( k_1 \) and \( k_2 \) in the cost function were setup such that \( k_1 + k_2 = 1 \). The magnitude of the obstacle potential is much larger than that of the maximum turning rate, so \( k_1 \) was chosen to be much smaller than \( k_2 \). The weightings were hand tuned to give a low collision penalty and were given values of \( k_1 = 0.05 \) and \( k_2 = 0.95 \).

The user maintained a view similar to Fig. 4 (a window 10 \( \text{m} \) wide by 10 \( \text{m} \) long) while driving. The view range along the horizontal axis does not change as the user drives. Only the view along the vertical axis changes as the user drives such that the user can always see 5 \( \text{m} \) ahead and behind their robot. An assumption made in both the “OAC Switch” and “OAC Blend” cases is that the location of all of the obstacles in the user’s view of the environment are known. Lastly, the optimization solver used for the MPC was \texttt{fmincon} in \texttt{MATLAB}.

B. Results and Discussion

The results of the 60 tests are displayed in Fig. 5. The results of the 5 tracks for each forward speed and control input method were averaged together to get each data point shown. Error bars representing one standard deviation above and below each point are included. The metrics for comparison of the control input methods were chosen to be average level of human control and collision penalty. The average level of human control was calculated by averaging the value of \( \alpha \) over the time of the test and multiplying by 100%.

From Fig. 5(a) it is obvious that the “OAC Off” results in an average level of human control of 100%. At the lower speeds of 2 and 4 \( \text{m/sec} \) the “OAC Switch” and “OAC Blend” methods are very close to each other and maintain a high level of human control. However, as the speed increases to 6 and 8 \( \text{m/sec} \) the “OAC Switch” method begins to drop off rapidly in the level of human control. The “OAC Blend” method maintains a relatively high level of human control by intervening less when it detects the human is giving
similar inputs to that of the obstacle avoidance and when the human is steering the robot into areas of lower potential. In particular the “OAC Blend” gave the user 26% more control compared to “OAC Switch” at $V = 8 \text{ m/sec}$.

From Fig. 5(b) it is evident that the “OAC Blend” control input method results in the smallest collision penalties, especially at higher speeds. Using the “OAC Blend” at $V = 8 \text{ m/sec}$ resulted in a 35% reduction in collision penalty compared to the “OAC Switch” at the same speed. Compared to pure human control in the “OAC Off” case at $V = 8 \text{ m/sec}$ the reduction in collision penalty from using the “OAC Blend” was 66%.

Figure 5(b) shows that the OAC is not able to avoid all obstacles. After removing the velocity control portion, it was not guaranteed that the calculated avoidance maneuver could be achieved due to limitations in the turning radius of the robot. However, the OAC did help decrease the number of collisions compared to the human alone at high speeds. Future work will consider velocity control to make the scenario more realistic and better avoid collisions.

The results here show that the model predictive control based blending of human inputs has the potential to increase the amount of control given to the user without sacrificing safety. Fig. 5(b) shows that the human alone really struggles to avoid obstacles at $V = 8 \text{ m/sec}$ compared to the other two methods that have assistance from the obstacle avoidance. When cross-examining this with Fig. 5(a) it is possible to see that the human still maintains a high level of control with “OAC Blend” obstacle avoidance helping.

VI. CONCLUSIONS

This paper suggests a model predictive control based framework for blending human inputs with a semi-autonomous controller. The advantage of this framework is that it provides a way for shared control system designers to give the user a higher level of control authority without compromising the safety added by certain semi-autonomous behaviors. This framework was applied to a differential drive robot with obstacle avoidance traveling through an obstacle field. The results showed that compared to switching between automatic controller and human control, the suggested framework gives the user significantly more control while reducing collisions as operation speed increases.

These results justify further exploration of this input blending method. Future work will look at other obstacle avoidance methods and more types of semi-autonomous behaviors. Sensitivity of MPC parameters such as control horizon, prediction horizon and cost function weightings will be further analyzed. User tests will be performed in more realistic environments (high fidelity simulation or on hardware) to demonstrate extension to real world scenarios.

ACKNOWLEDGMENT

The authors would like to thank Yingshi Zheng for his assistance creating the simulation.

REFERENCES