

Behavior Circuits, a Framework to combine Human and Machine Commands

Jan Baumgärtner

*Institute for Computer Engineering
Heidelberg University
Heidelberg, Germany
jan.baumgaertner@ziti.uni-heidelberg.de*

Holger Dieterich

*Institute for Computer Engineering
Heidelberg University
Heidelberg, Germany
holger.dieterich@ziti.uni-heidelberg.de*

Jens Wagner

*Institute for Computer Engineering
Heidelberg University
Heidelberg, Germany
jens_wagner@online.de*

Simon Otto

*Institute for Computer Engineering
Heidelberg University
Heidelberg, Germany
simonberthold@web.de*

Essameddin Badreddin

*Institute for Computer Engineering
Heidelberg University
Heidelberg, Germany
badreddin@ziti.uni-heidelberg.de*

Abstract—This paper presents a framework for shared control of semi-autonomous ground vehicles on an operational level. The so-called behavior circuits can extend and unify existing arbitration strategies realizing a rule-based fusion of inputs. This allows human users to build an internal model of the assistance.

We explain how to design behavior circuits for desired arbitration strategies. Using one sample circuit, we demonstrate that the use of behavior circuits for arbitration between human and multiple assistive systems results in immediate assistive benefit.

Index Terms—mobile robotics, shared control, input mixing, human-machine interface

I. INTRODUCTION

Machines are slowly transforming from devices we control to devices with which we cooperate to achieve a shared goal. To most people, this transformation is probably nowhere more apparent than in the automotive industry. With new advanced systems such as BMW’s driving assistant [1], humans are no longer the sole drivers of their cars. The issue now arises that given different steering commands between a human and an assistive system, it is unclear whom the car should follow. In some situations, the assistive system might prevent an accident, while in others it might misjudge the situation whereas the human reacts correctly.

Such a scenario in which multiple agents want to control the behavior of a system is often referred to as shared control. Since the term is regularly used without a concise definition, we refer to the definition of Abbink et al. [13]. According to Abbink et al., shared control can take place on different levels. This paper proposes a new framework for the fusion of uncoupled command inputs on an operational level using input-mixing [13]. In broad terms, this means that the framework directly combines multiple continuous steering commands into a single command, which then controls the system.

Various input-mixing approaches have been proposed in the literature:

In deliberative shared control, human input can override an assistive device that is otherwise fully autonomous [4] [10]. By contrast, in safeguarded navigation, the user provides the default behavior while the machine intervenes in specified cases like imminent collisions [3] [4].

Such discrete switches between human operator and assistive system can lead to jerky behavior if both want to steer in different directions [7]. In the case of driving, such jerks are not only unpleasant for the operator; he will also try to compensate for them, resulting in even worse performance [7].

To overcome this problem, other approaches smoothly blend the steering commands of both the human and the assistive system. From now on, we will refer to these steering commands as behaviors.

In behavior-based shared control, this blending is usually performed using self-defined functions [8] [9]. Since the functions are arbitrary their design is nontrivial. Furthermore, not every function has an intuitive interpretation which might make it difficult for the user to anticipate the blended behavior [13].

Dynamic shared control approaches make use of a weighting factor $\eta \in [0, 1]$ to blend assistant behavior v_a and human behavior v_h into a final behavior v [4] [11]:

$$v = \eta v_a + (1 - \eta) v_h \quad (1)$$

Since η can be a function, an agent’s influence on the resulting behavior can shift dynamically. Because such a function is arbitrary the approach also suffers from the same problem of interpretability.

Both dynamic shared control and behavior-based approaches are conventionally used with one assistive system [4] [11]. Using multiple assistive systems, therefore either requires switching between them depending on the situation [9] or extending the approaches. The first solution reintroduces the

previously encountered jerky motion that should be avoided. The second solution increases the complexity and can make it more difficult to form an internal model of the input mixing. Thus making it hard to predict the blended behavior.

To recapitulate, the previously mentioned approaches either suffer from nonsmooth behavior or make it difficult for the human to build an internal model of the assistance. This paper proposes behavior circuits, a rule-based fusion framework that helps the human user to build an internal model of the assistance.

These rules are implemented using a small set of functions based on logic gates. These functions can be interconnected into circuit-like structures that model desired types of input mixing.

Such a function system was first proposed by Badreddin [2] where they were used to fuse the behavior of multiple robot subsystems. Our work applies these concepts to input mixing and proposes a new rule-based design framework for circuit design. We also expand the original function set and redesign the functions for better performance and broader use cases.

We will show that behavior circuits can model smooth versions of deliberative shared control and safeguarded navigation. Furthermore, we will explain how behavior circuits can be used to combine multiple input mixing approaches to create new systems.

The rest of this paper is structured as follows: Section II motivates the usage of logic-gate-like functions before explaining how they are used for input-mixing. These functions will be called analogical gates and are derived in section III. Since the basic gates on their own are not very expressive, section IV shows how they can be connected to compound gates modeling more complex rules. Section V shows how to design circuits combining multiple behaviors using this extended set of gates. Example circuits are then designed in section VI to prove that the framework can model and extend current approaches. An experiment was performed to show that the usage of such a circuit results in improved driving performance. Section VII describes the experiment setup while section VIII describes the results. The paper closes with a discussion in section IX and a conclusion in section X. Here the strengths and weaknesses of the proposed system are recapitulated and further research directions are outlined

II. HOW BEHAVIOR CIRCUITS WORK

As mentioned above, the gates implement rules that govern the interaction between the inputs. These rules are based on binary logic. To motivate this we will use a simplified car control scenario. Here, how the car *turns* is controlled by a human behavior and one assistive system behavior. Assuming that one can either *turn* or *not turn*, this is a binary problem. In this simple scenario, every input mixing function can be expressed as some kind of logic circuit. These circuits all implement rules governing when the system should turn. The rule *turn if the human and the assistive system want to turn* is implemented by a single AND gate. Consequently, the

rule *turn if either one of the two actors wants to turn* is implemented by an OR gate.

In a real driving scenario, the choice is not between *turning* or *not turning*, but how much to turn. That is to say that the steering command v is defined on a range of $[v_{min}, v_{max}]$.

The main idea of behavior circuits is to extend binary logic gates so that the approach of the previous scenario can be applied to continuous commands.

These analogical gates are monotonic functions that preserve the rules of their binary counterparts while producing smooth values on their defined range. They can be connected to form behavior circuits which implement smooth extensions of every possible binary rule. Constructing a circuit in this context means combining the analogical gates into a composite function. An example of this can be seen in Fig. 1.

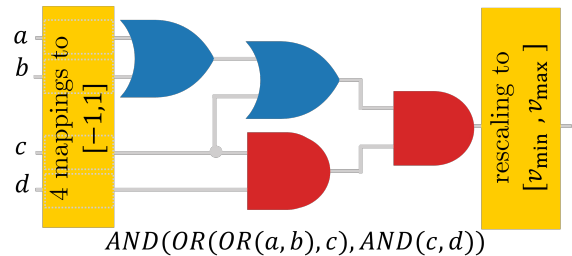


Fig. 1. Sample Circuit fusing behaviors a , b , c and d including input mapping and output rescaling

Analogical gates are defined on the continuous interval $[-1, 1]$ instead of the interval $[0, 1]$. The negative sign is used to distinguish between actions like turning to the right and turning to the left. This example also serves as an interpretation of the value -1 ; it is the opposite control action to 1 , while 0 means no action. In the case of turning a car, this means -1 can be interpreted as turning fully to the left, 1 as turning fully to the right, and 0 as not turning at all. This interpretation extends to all values between -1 and 1 . -0.5 can thus be interpreted as turning half as much to the left as -1 .







Since in the previous driving example the steering command is defined on the interval $[v_{min}, v_{max}]$ circuit inputs first have to be downscaled to a range of $[-1, 1]$. This can be done for example by first dividing each circuit input by $\max(|v_{max}|, |v_{min}|)$. The circuit output can then be re-scaled.

Any time no logical opposite can be defined, mapping the input only to $[0, 1]$ is advised to preserve the interpretation of positive and negative values.

To summarize: Behavior circuits are made up of analogical gates. These represent simple rules that govern the interactions of the input behaviors.

Table I, among other things, summarizes these rules for different Gates using the example of *turning*. Note that instead of *turning* any other control action such as *driving forward* could be substituted.

TABLE I
TURN RULES FOR EACH GATE

Gate	Rule	Equation	Symbol
AND	Turn in a direction if both a and b want to turn in the direction	$ab \frac{\tanh((a+b)/\lambda)}{\tanh(2/\lambda)}$	
OR	Turn in a direction if a or b want to turn in the direction	$a + b - ab \frac{\tanh((a+b)/\lambda)}{\tanh(2/\lambda)}$	
AMP	Turn like b amplified by how a wants to turn	ab	
PREVAIL	Turn like b except if a wants to turn, then turn like a	$OR(a, OR(a, b))$	
MULTIMIXER	Turn like a or b depending on η	$OR(AND(a, \eta), AND(b, NOT(\eta)))$	
INVOKE	Turn like a or b if a wants to turn	$AND(a, OR(a, b))$	

III. ANALOGICAL GATE DERIVATION

The first investigation into analogical gates defined on $[-1, 1]$ was performed by Badreddin [2]. While the analogical gates derived in this paper follow the idea of [2] that the gates should be point symmetric, they have a different description that adheres closer to the behavior of their binary counterparts and is less nonlinear. The derivation of these new analogical gates is a two-step process:

- extend the binary AND and OR to $[0,1]$ using t-norms and s-norms respectively
- extend the t-norms and s-norms to $[-1,1]$ such that monotony is maintained and the resulting gates are point symmetric.

T-norms and s-norms can be understood as multivalued extensions of binary logic [12]. Since the values between zero and one are not uniquely defined by binary logic there are multiple possible t-norms and s-norms. To avoid jerky motion these norms should be smooth. Furthermore, they should not introduce unnecessary nonlinearities. These considerations have led to the choice of the product t-norm and s-norm [12].

$$\begin{aligned} \text{Product t-norm } \top_{Prod}(a, b) &= ab \\ \text{Product s-norm } \perp_{Prod}(a, b) &= a + b - ab \end{aligned} \quad (2)$$

where a and b are input values defined on $[0, 1]$. The simplest way to achieve the desired extension specified in B) is to multiply the norms with $sign(a+b)$. However this produces a nonsmooth gate since the sign function is not smooth at zero. Since smoothness is a desired property of the gates, the sign function was smoothed using the tangens hyperbolicus. This results in the final description

$$\begin{aligned} AND(a, b) &= ab \frac{\tanh((a+b)/\lambda)}{\tanh(2/\lambda)} \\ OR(a, b) &= a + b - ab \frac{\tanh((a+b)/\lambda)}{\tanh(2/\lambda)} \end{aligned} \quad (3)$$

where a and b are defined on $[-1, 1]$ and λ is the smoothing magnitude. The factor $\tanh(2/\lambda)$ is used to ensure that $AND(1, 1) = OR(1, 1) = 1$. A contour plot of both functions can be seen in Fig. 2. While a bigger λ leads to a smoother

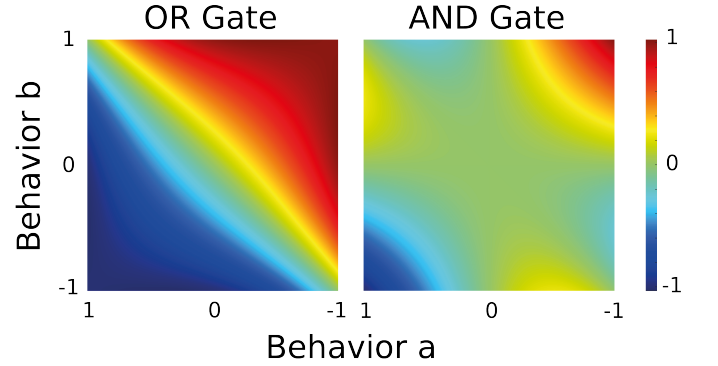


Fig. 2. Surfaceplot of AND and OR gate with smoothing magnitude $\lambda = 1$

gate behavior, it also results in a slight deviation from the boundary values. For example $OR_{\lambda=0.5}(1, 0.1)$ is only 0.973 instead of 1.0. In practice, a smoothing magnitude of $\lambda = 1$ seems to be a good compromise. For the rest of this paper, we will assume that all gates use a smoothing magnitude of $\lambda = 1$.

Since more complex circuits might require negation, a NOT operation can also be defined:

$$NOT(a) = (1 - |a|) \frac{\tanh(a/\lambda)}{\tanh(1/\lambda)} \quad (4)$$

Where λ is again the smoothing magnitude.

While this set of gates is already very expressive, they offer no way to map the input pairs $(-1, -1)$ and $(1, 1)$ to the same nonzero value. This means it is not possible to implement rules which behave the same for negative and positive signs. For this, an additional gate is required.

For this purpose we proposed the AMP or AMPLIFICATION gate. Its simplest implementation is a multiplication:

$$AMP(a, b) = ab \quad (5)$$

IV. COMPOUND GATES

Three types of basic gates mean that three basic rules can be implemented. Moreover, these rules are all symmetrical. That means it is not possible to model rules where one input takes precedence over another using a single analogical gate.

Fortunately, like logic gates, analogical gates can be combined into more complex circuits. These circuits can then model more complex rules. Such a reusable circuit, which models a rule between two inputs, will be called a compound gate. Unfortunately, the problem of synthesizing a circuit from a desired rule is as of yet unsolved. Until then the design of compound gates is mainly based on intuition.

In the case of one input a taking precedence over another input b , a compound gate can be built with two OR gates:

$$PREVAIL(a, b) = OR(a, OR(a, b)) \quad (6)$$

The resulting circuit is called a PREVAIL gate and was first proposed in [2]. One should note that in the binary case, this gate would simplify into an OR gate. Unfortunately, most binary simplification rules no longer hold for analogical gates. Yet, it is still possible to construct any simplified binary circuit using analogical gates. This will produce a smooth variant of the binary counterpart. An example of this is the analogical multiplexer, which we call MULTIMIXER. It smoothly transitions between two signals as the controlling input η moves between 0 and 1. This implements a version of the dynamic shared control of [11]. In our opinion, the slight nonlinearities introduced due to the tanh function are not noticeable in practice. Another example of a compound gate is the INVOKE gate which will be used later. The rules the compound gates implement as well as their formulation can be seen in Table I.

V. BUILDING CIRCUITS

Constructing a circuit is synonymous with specifying the rules that should govern the interaction of all behaviors. For one human behavior and one assistive system behavior, this is analogous to the motivating example of section II. Meaning the rules of Table I can be used to find a suitable gate fusing the two behaviors. On the off-chance that no current gate captures the desired type of assistance, a new compound gate has to be designed.

Since these cases are very uncommon in our experience the main difficulty in behavior circuit design is combining multiple assistive systems with human input. In this case, not only the choice of gates is important but also their interconnection.

This can be further complicated by systems with multiple control axes. Meaning each input behavior is also made up of multiple values. Fortunately in practice different controlled axes can often be controlled by independent sub-circuits.

The problem of interconnection can also often be simplified. Since the purpose of behavior circuits is not to combine

assistive systems with each other, but with human behavior, circuits often have a cascaded topology. Meaning that assistive behaviors are combined one after another with the previous combination of human and assistive behaviors. An example of this can be seen in Fig. 3 where two assistive systems were combined with human input. It should be noted that sometimes it is still helpful to first combine multiple assistive systems into a more abstract type of assistance. For example, if the systems themselves are black-box modules.

In most cases, however, a cascaded topology is the most practical. This topology simplifies circuit design since one only has to decide in which order to combine the systems and which gates to use. This is a matter of design and depends on the assistive systems and the desired result.

Another matter of design is the tuning of the circuit. While it is possible to specify the rules governing the input mixing, the final shape of the function is predetermined by the shape of the gates used in the circuit. Sometimes one might want to fine-tune this shape.

Suppose two behaviors are connected by a PREVAIL gate. While the gate models the desired interaction of behavior a taking precedence over behavior b , this might happen too abruptly for the user.

It is generally advised to solve this by tuning the parameters of behavior a . However, if this is for some reason not possible the behavior circuit can be used in conjunction with self-designed nonlinear mapping functions $f(a)$. These provide a custom mapping from $[-v_{max}, v_{max}]$ to $[-1, 1]$ (see Fig. 1) and can be used to compensate for undesirable input behavior. While the choice of function is arbitrary it is advisable to divide it into a nonlinear function $\tilde{f}(a) [-1, 1] \rightarrow [-1, 1]$ and a linear mapping $m(a) [-v_{max}, v_{max}] \rightarrow [-1, 1]$, where

$$f(a) = \tilde{f}(m(a)) \quad (7)$$

Such a division makes the shape of the nonlinearity independent of the input range which increases reusability.

VI. EXAMPLE CIRCUITS

Section IV showed that a MULTIMIXER implements equation (1). This section will expand on this and show sample circuits for different types of input mixing. All circuits will be built to control the angular velocity ω and linear velocity v of a mobile robot.

Following the advice of section V, each circuit can be divided into two independent sub-circuits, one for each control axis. The circuit for dynamic shared control can be seen in Fig. 5 and uses two MULTIMIXERS.

A sample circuit that models a type of smooth deliberative shared control between a human and an assistive device can be seen in Fig. 4. Instead of using a button to switch control between the user and the assistive system, the control smoothly shifts to the human as his command deviates from a default value.

This default value was defined to be $v = 1, \omega = 0$. Because of this, a PREVAIL gate is used to allow the human operator's angular velocity to overwrite the angular velocity

of the assistive system. Assuming that the robot can not drive backward an AND gate can be used to overwrite the linear velocity with a smaller value. While this system needs no extra button, its disadvantage is being unable to overwrite the assistance system with the default values. From a practical standpoint, such a setup only makes sense when the assistive device is designed in tandem to work around this problem.

For a classical switch, a binary button can be used to control the control input η of the circuit in Fig. 5. This transition could also be smoothed with ramp functions to minimize jerking.

One of the advantages of the behavior circuit approach is its ability to use more than one assistive system and combine existing methods. This is showcased by the circuit in Fig. 3 which is once again divided into two independent sub-circuits. Here a safeguarded navigation is combined with a behavior-based shared control approach. This requires two assistive systems.

The first is an emergency collision avoidance system that turns and breaks if the system would otherwise collide with an obstacle.

The second assistive system, the steering assistance, tries to help the human steer to a specified target position by planning and following a path from its current position to the target.

The sub-circuit for linear velocity implements the rule *drive like the human OR the steering assistance if the human AND the collision avoidance system want to drive*.

The sub-circuit for angular velocity implements the rule *turn like the human OR the steering assistance EXCEPT IF the collision avoidance wants to turn, then turn like the collision avoidance*.

These rules ensure that the system will not collide with an obstacle no matter what happens, while still allowing the steering assistance to support the human operator.

The collision avoidance takes on the role of a virtual sensor, which in a safeguarded navigation system decides when to break [3]. In contrast to classical safeguarded navigation, it can not only decide if it should break but also by how much. This works best if the collision avoidance is tuned or designed in tandem with the circuit, see section V.

In the absence of collision avoidance, the operator input is combined with the steering assistance using a self-defined function. This makes this circuit a type of behavior-based shared control. All behavior circuits can be seen as a type of behavior-based shared control which uses a function system to design custom functions. Instead of switching between active behaviors the circuit itself decides when each behavior should influence the output.

VII. EXPERIMENTAL VALIDATION

A series of experiments were designed with two goals in mind. First to show that behavior circuits provide an assistive benefit. And secondly to show that they are expressive: Meaning that the contribution of each input behavior dynamically changes depending on the situation.

For this purpose, a scenario was set up in which a human teleoperated a mobile robot. The operator was in this case

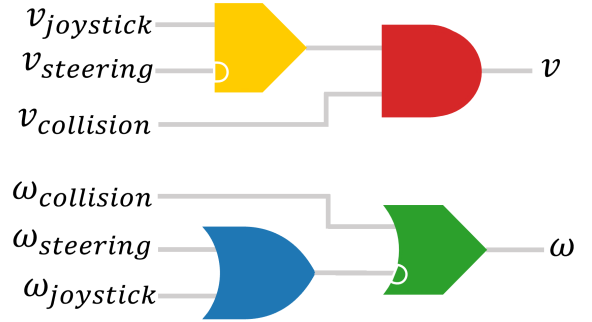


Fig. 3. Custom Circuit for example scenario

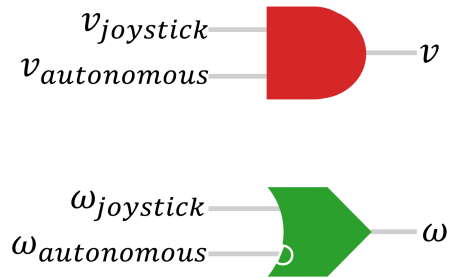


Fig. 4. Sample circuit for deliberative shared control of angular velocity ω and linear velocity v

tasked with navigating from a start point to a specified target. To benefit from assistance three types of impediments were added to the output of the operator's joystick which prevents him from accurately controlling the robot:

- jittery operator motion simulated by additive Gaussian random noise ξ
- low control precision of the operator simulated by a quantization of the operator's input
- slow reaction time simulated by a randomized time delay τ until the robot considers the next operator command.

$$\begin{aligned} v_{impeped}(t + \tau) &= \text{quantize}(v_{normal}(t)) + \xi \\ \xi &\sim \mathcal{N}(0, 0.01) \quad \tau \sim \mathcal{N}(0, 2)^2 \\ \omega_{impeped}(t + \tau) &= \text{quantize}(\omega_{normal}(t)) + \xi \\ \xi &\sim \mathcal{N}(0, 0.3) \quad \tau \sim \mathcal{N}(0, 2)^2 \end{aligned} \quad (8)$$

In this scenario, the linear velocity was quantized into three steps while the angular velocity was quantized into five steps. It should be noted that these impediments are severe and may not be representative of typical operator impediments.

Two baseline experiments were conducted to compare against the performance of circuit-based assistance. First, the operator was tasked to drive to the specified target using a normal unimpeded joystick. These experiments were then

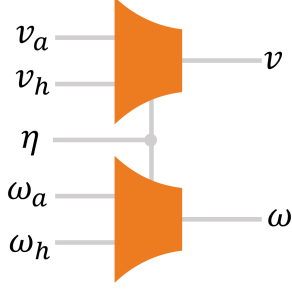


Fig. 5. Sample circuit for arbitration of angular velocity ω and linear velocity v

repeated with the impeded joystick. In the third scenario, the operator was aided by the circuit of Fig. 3.

All experiments took place in an arena set up in our lab with different regions modeling different complications. A turtlebot burger was used as a robot. An overview of the arena can be seen in Fig. 7 while the robot is pictured inside the arena in Fig. ???. The first region is a wide-open area with a small



Fig. 6. Turtlebot Robot at the Entrance of Region 2

opening leading to the next region. While a steering assistant can help produce a smoother trajectory, a collision-avoidance system is not needed. The second region is a narrow corridor where it is easy for an impeded driver to collide with a wall. The third region features a huge spool at its center. Since the robot's laser scanners only pick up the much smaller inner diameter of the spool, the spool models an obstacle that is only seen by the human.

The experiments were conducted with five participants with no prior exposure to driving assistance using behavior circuits. Before using the assistive systems, they were first shown the behavior of each assistive system, the fusion circuit seen in Fig. 3 and rules implemented by each gate as seen in Table I.

VIII. RESULTS

The trajectories created by driving with a normal joystick impeded joystick and assisted joystick respectively were plotted in Fig. 8. Note that since the turtlebots odometry was used to update its position the measurement gets less accurate as

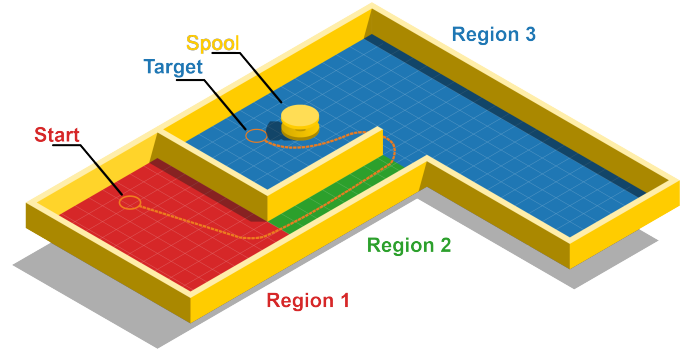


Fig. 7. Arena for Experiments

time goes by. However the resulting Fig. is only meant to showcase the qualitative differences between normal joystick driving, impeded joystick driving and assisted driving. Each trajectory was plotted until the robot either reaches the target or collides with an obstacle.

Comparing the impeded joystick trajectory with the assistive trajectory, one can already see significant improvements. For example, all but two impeded joystick operators collided with an obstacle while no assisted operator did. Note that the two remaining impeded operators did in fact reach the target. Their uneven movement caused the odometry to be much worse, distorting their true position and causing participant 5 to appear as if he temporarily left the arena.

The assistive trajectories are also much smoother though they also show strange behavior on the first two meters. This is, in all likelihood, due to the operator getting accustomed to the assistance provided by the circuit. This would indicate that the operators are quickly able to grasp how the system will assist them and use this knowledge to their advantage resulting in improved performance from then on out. However, since the experiment had no control group and only a small sample size this is currently only conjecture.

For a more quantitative analysis and comparison of the experiments, three metrics were chosen. These were the smoothness of the trajectory, the fluency of the trajectory, and the time the participants needed to reach the target (time to target). Fluency $\Gamma(x)$ and smoothness $\Theta(\phi)$ were defined by:

$$\begin{aligned}\Theta(\phi) &= \exp\left(-R\left(\frac{\partial\phi}{\partial t}(t)\right)\right) \\ \Gamma(x) &= \exp\left(-R\left(\frac{\partial\|x(t)\|}{\partial t}\right)\right)\end{aligned}\quad (9)$$

Where ϕ is the orientation of the robot x its position and $R(L)$ is the one-step autocorrelation of the signal L .

Smoothness primarily describes how smooth the robot changes its orientation, while fluency describes how smooth it covers distance. The measures have been adopted from [11] with the original mean replaced by the autocorrelation as the mean is unsuited for situations where the system has to stop at some point.

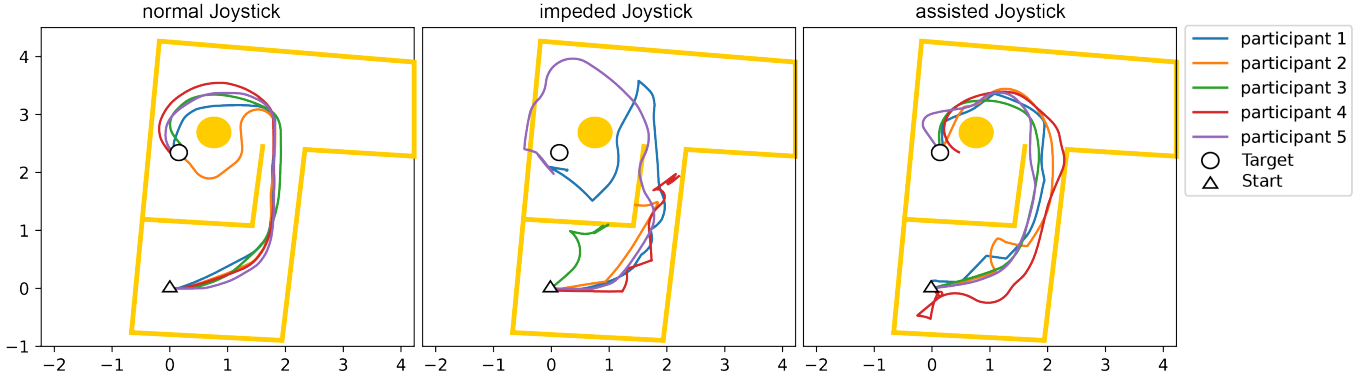


Fig. 8. Robot Trajectories with different Joystick Configurations

The results of these measures for each participant can be found in Table II. As expected, the participants performed worst with the impeded joystick in all but one case. For participant 4 the impeded joystick was smoother than the assisted joystick. This can probably be explained by the initial control problems the participant experienced at the start as seen in Fig 8. In all other cases, the performance of the participants improved when using the assisted system, in some cases even surpassing the performance with the normal joystick.

While this indicates that behavior circuit-based assistance improves driving performance, the question remains: Is this due to the assistive system forcing the robot into the desired behavior, or the assistance dynamically interplaying with the human command.

To illustrate that the influence of the input behaviors dynamically shifts, Fig. 9 shows the behavior of all assistant systems for participant 5. The plot is divided into two subplots one for each control axis.

Looking at the subplot of the angular velocity ω the influence of the OR gate can be seen from $t = 0s$ to $t = 20s$. Similar to a bounded sum, the turning commands are added on top of each other, the faster-reacting assistant is in this case able to finetune the slower reacting time-delayed command of the impeded joystick.

Between time $t = 20s$ and $t = 30s$ the collision avoidance becomes active. The positive input of the combined steering assistance and impeded joystick is discarded and the output assumes the negative values of the collision avoidance instead. This is the intended behavior of the PREVAIL gate. Note that the amplitude of the output is greater than the amplitude of the collision avoidance. This is a side effect of the PREVAIL gate which introduces nonlinearity.

Between $t = 20s$ and $t = 30s$ the collision avoidance not only wants to *turn* but also *brake*. This can be seen in the subplot of the linear velocity v . Here the AND gate causes the output to decrease as the collision avoidance input decreases.

The effects of the INVOKE gate start to be observable after $t = 35s$ since before participant 5 almost always drove with

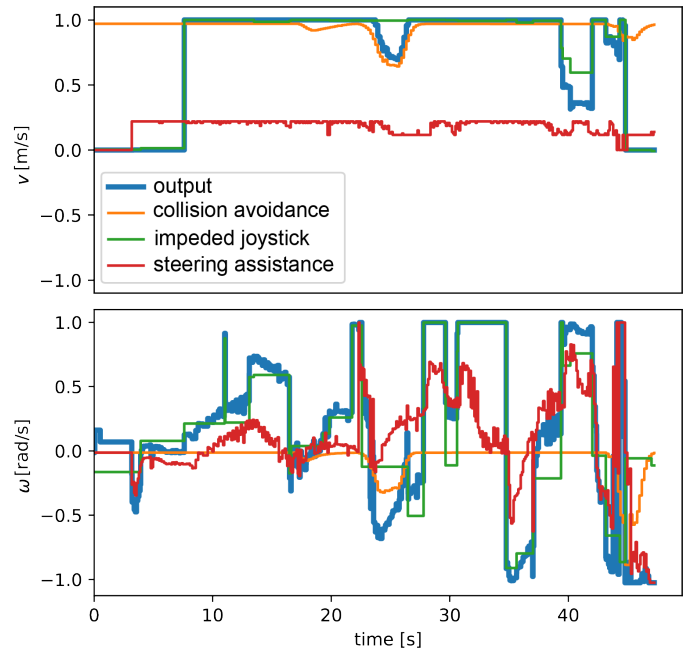


Fig. 9. comparison between circuit input and output behaviors (top: linear velocity, bottom: angular velocity)

full linear velocity.

Now the output follows the general shape of the impeded joystick, however, the steering assistance still fine-tunes the behavior similar to the OR gate. The reason that the output decreases more than the joystick command is due to the nonlinearity of the INVOKE gate.

To conclude, the contribution of each gate can be seen as they dynamically shift the influence of each behavior depending on the situation. This shows that behavior circuits are capable of expressively interweaving the different behaviors even without an external weighting factor η .

IX. DISCUSSION

The experimental evidence proves that it is possible to use behavior circuits to improve the driving performance of an

TABLE II
ROBOT TRAJECTORY METRICS

	Smoothness $\Theta(\phi)$			Fluency $\Gamma(x)$			Time to Target T [s]		
	normal	impeded	assisted	normal	impeded	assisted	normal	impeded	assisted
participant 1	0.982	0.872	0.884	0.982	0.982	0.986	54.1	88.1	62.4
participant 2	0.968	0.806	0.911	0.979	0.969	0.987	53.7	170.1	76.0
participant 3	0.985	0.892	0.981	0.979	0.974	0.978	51.6	85.4	39.1
participant 4	0.984	0.848	0.838	0.977	0.973	0.975	52.5	72.9	64.4
participant 5	0.983	0.954	0.957	0.976	0.970	0.980	47.5	52.2	48.1

impeded operator. We acknowledge that one particular example is not enough to prove the viability of a framework. We are currently working on a more comprehensive comparison between behavior circuits and other approaches. In this sense, the experiment can be seen as a showcase motivating the use of behavior circuits for input mixing. Specifically, it showed that behavior circuits can dynamically shift the influence of the input behaviors even in the absence of an external weighting factor η .

The results also indicate that the human operators could quickly grasp how the system is assisting them at any given time. However, the sample size of this study was quite limited, and the participants were not representative of the general population. A more comprehensive study should be conducted to thoroughly investigate the learnability of behavior circuit-based systems.

We also showed that the framework offers extensions and generalizations of different shared control approaches as seen in section VI. The section also demonstrated that behavior circuits allow the unification of different approaches in a single system.

However, there are also disadvantages. Due to their design, gates introduce nonlinearity into the fusing process that could be avoided with custom build functions. This is especially apparent for the compound gates. At the same time, this nonlinearity is only barely noticeable during driving from our experience when testing and tuning the gates and circuits.

X. CONCLUSION

In this paper, a new input mixing framework was proposed to fuse human input with the input of multiple assistive systems. This framework uses extensions of binary logic, called analogical gates, to build behavior circuits. The input of these behavior circuits is the normalized behavior of the human and each assistive system. The circuits then produce a normalized output fusing the inputs. We showed that the behavior circuits can emulate different input mixing strategies. The experimental results also indicate that such a circuit could improve the driving performance of an operator using an impeded joystick. These experiments were however limited only to one circuit and a small set of participants. More broad studies need to be carried out to further investigate human ability to develop an internal model for the workings of the circuits.

We hope nevertheless that the presented framework will be useful to develop more sophisticated shared control applications.

To further this goal we are currently working on a more structured approach to behavior circuit design analogous to the design of logic circuits given functional requirements.

REFERENCES

- [1] "Overview of the main driver assistance systems" February 2, 2021 Accessed on: Sep. 22, 2021. [Online]. Available: <https://www.bmw.com/en/innovation/the-main-driver-assistance-systems.html>
- [2] E. Badreddin, "Fuzzy relations for behaviour-fusion of mobile robots", *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994, pp. 3278-3283, doi: 10.1109/ROBOT.1994.351066
- [3] C. Mandel, K. Huebner, and T. Vierhuff "Towards an autonomous wheelchair: cognitive aspects in service robotics" *Proceedings of towards autonomous robotic systems*, 2005 pp. 165172
- [4] C. Urdiales, J. M. Peula, M. Fdez-Carmona, C. Barru, E. J. Prez, I. Snchez-Tato, ... and Sandoval, F. (2011). "A new multi-criteria optimization strategy for shared control in wheelchair assisted navigation", *Autonomous Robots*, 2011, pp. 179-197.
- [5] A. Erdogan and B. D. Argall, "Prediction of user preference over shared-control paradigms for a robotic wheelchair," *2017 International Conference on Rehabilitation Robotics (ICORR)*, 2017, pp. 1106-1111, doi: 10.1109/ICORR.2017.8009397 .
- [6] A. Benloucif, A. T. Nguyen, C. Sentouh, and J. C. Popieul "Cooperative trajectory planning for haptic shared control between driver and automation in highway driving" *2009 IEEE Transactions on Industrial Electronics*, 2019, pp. 9846-9857.
- [7] Y. Horiguchi and T. Sawaragi, "Effects of probing behaviors to adapt machine autonomy in shared control systems," *2005 IEEE International Conference on Systems, Man and Cybernetics*, 2005, pp. 317-323 Vol. 1, doi: 10.1109/ICSMC.2005.1571165 .
- [8] RS. Rao, K. Conn, SH. Jung, et al., "Human-robot interaction: application to smart wheelchairs," *Proceedings 2002 IEEE International Conference on Robotics and Automation*, 2002, pp. 3583-3588 vol.4, doi: 10.1109/ROBOT.2002.1014265 .
- [9] T. Rfer, C. Mandel, A. Lankenau, B. Gersdorf, and U. Frese "15 years of Rolland", *Specification Transformation Navigation*, 2009.
- [10] M. Mazo, "An integral system for assisted mobility [automated wheelchair]", *IEEE Robotics & Automation Magazine*, vol. 8, no. 1, pp. 46-56, March 2001, doi: 10.1109/100.924361 .
- [11] Q. Li, W. Chen and J. Wang, "Dynamic shared control for human-wheelchair cooperation," *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4278-4283, doi: 10.1109/ICRA.2011.5980055 .
- [12] L. A. Zadeh, G. J. Klir, and B. Yuan "Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers" (Vol. 6). World Scientific. 1996
- [13] D. A. Abbink et al., "A Topology of Shared Control Systems Finding Common Ground in Diversity," *IEEE Transactions on Human-Machine Systems*, vol. 48, no. 5, pp. 509-525, Oct. 2018, doi: 10.1109/THMS.2018.2791570 .