Integration of LLM-based Intelligent Agents to Assist Decision-Making in Autonomous Vehicles

Victor Novoli Mendonça[1], **Gleifer Vaz Alves**[1], **André Pinz Borges**[1], **Rafael C. Cardoso** [2]

Universidade Tecnológica Federal do Paraná (UTFPR)
Ponta Grossa, PR, Brasil

University of Aberdeen
Aberdeen - United Kingdom

victornovoli@alunos.utfpr.edu.br, rafael.cardoso@abdn.ac.uk

{apborges, gleifer}@utfpr.edu.br

Abstract. *This work presents the integration of MASPY intelligent agents with large language models to support decision-making in autonomous vehicles. The objective is to demonstrate the feasibility of integrating MASPY agents with language models and to analyse the resulting decision-making process in urban traffic scenarios. Additionally, the implications of applying traffic rules to the system are discussed, as well as applications of ethical models involving LLMs. To this end, system prompts were developed and tested, and the agents were integrated with Gemini via API calls. Finally, the results, limitations, and perspectives for future studies are discussed.*

Resumo. *Este trabalho apresenta a integração de agentes inteligentes MASPY com grandes modelos de linguagem com o propósito de apoiar a tomada de decisão em veículos autônomos. O objetivo é demonstrar que a integração de agentes MASPY com modelos de linguagem é viável e analisar o processo de tomada de decisão resultante dessa integração em cenários de tráfego urbano. Adicionalmente, são discutidas as implicações da aplicação de regras de trânsito no sistema, bem como a aplicação de modelos éticos envolvendo LLMs. Para isso, foram desenvolvidos e testados prompts para o sistema, e a integração dos agentes com LLMs foi realizada por meio de chamadas à API do Gemini. Finalmente, são discutidos os resultados, as limitações e as perspectivas para estudos futuros relacionados ao trabalho.*

1. Introdução

Agentes inteligentes têm assumido um papel cada vez mais relevante no desenvolvimento de veículos autônomos, especialmente em cenários que exigem tomada de decisão adaptativa, autonomia operacional e capacidade de responder a mudanças no ambiente. Nesse contexto, arquiteturas baseadas em crenças, desejos e intenções, conhecidas como BDI (*Belief-Desire-Intention*), oferecem uma forma estruturada de modelar o comportamento deliberativo desses agentes. Com o avanço dos grandes modelos de linguagem, do inglês *Large Language Models* (LLMs), surge também a possibilidade de utilizá-los como mecanismos auxiliares no processo de tomada de decisão, contribuindo para tornar o comportamento dos agentes mais flexível, contextualizado e robusto.

De acordo com [Wooldridge 2009], evidencia-se que não existe uma definição universal do que são agentes, mas que um conceito central é a noção de autonomia. Desse modo, uma definição mais genérica, usada no livro e ainda válida na atualidade, é a de que sistemas de agentes são entidades computacionais capazes de tomar ações de forma autônoma em um ambiente para atingir seus objetivos. Wooldridge aponta que um problema-chave na programação de agentes é decidir quais decisões devem ser tomadas para que o agente atinja melhor seu propósito. Assim, com o surgimento dos LLMs, também surge uma possível alternativa para lidar com esse obstáculo.

Quando o assunto é trânsito urbano, sistemas multiagentes (SMA) protagonizam o cenário, pois permitem modelar esses cenários por meio de agentes inteligentes (que, nesse caso, representam os veículos autônomos) que trocam informações entre si e atuam em um ambiente. Tais veículos possuem uma tomada de decisão naturalmente reativa, com comportamento definido por regras pré-estabelecidas. A integração desses agentes com LLM adiciona uma "camada" de racionalidade ao programa, o que permite análises precisas e isoladas de cada situação de trânsito de modo bem mais específico. Assim, em um cenário em que dois veículos estão na iminência de colidir, um LLM (nesse caso, Gemini 3.5-flash) pode analisar a situação e usar seu raciocínio para deliberar sobre qual é a melhor ação a ser tomada, visando mitigar os possíveis efeitos desastrosos de um acidente de trânsito.

O sistema proposto utiliza imagens extraídas de um *dataset* de cenários de tráfego urbano. O dataset foi elaborado para avaliar o potencial de uso e os benefícios dessa tecnologia no contexto de agentes inteligentes, usando o cenário de veículos autônomos. O foco do trabalho é analisar a integração dos agentes com LLM, não a aplicabilidade no mundo real em tráfego urbano. As imagens são fornecidas como entrada ao LLM, que interpreta o contexto e toma decisões com base em princípios éticos e em regras de trânsito previamente definidas. Vários prompts foram testados para avaliar o comportamento da LLM sob diferentes instruções e identificar o melhor entre eles. Um módulo de contagem de tempo também foi incluído para avaliar a agilidade do LLM ao analisar a imagem e gerar uma saída para o programa. Portanto, o objetivo central desse artigo é desenvolver a integração de agentes inteligentes programados no framework MASPY [Mellado et al. 2025] com um LLM (agente MASPY-Gemini), para estudar o comportamento desses agentes, com base nas imagens do dataset.

O restante deste artigo está estruturado da seguinte forma: a Seção 2 apresenta a fundamentação teórica relacionada aos tópicos de programação BDI, tomada de decisão, ética e conformidade com as regras de trânsito. A Seção 3 expõe como o sistema foi modelado, como funciona, como foi elaborado o dataset utilizado e quais prompts foram empregados. A Seção 4 apresenta alguns resultados de testes do sistema, e a Seção 5 conclui e apresenta perspectivas futuras.

2. Referencial Teórico

O avanço nos LLMs ampliou as capacidades de raciocínio de SMA. Esses modelos são capazes de compreender linguagem natural, raciocinar, interpretar e tomar decisões com base em prompts. Nesse contexto, SMAs vêm sendo implementados para modelar comportamentos de veículos autônomos em ambientes de alta dinamicidade [Fu et al. 2025].

Assim, este referencial teórico apresenta os principais conceitos relacionados a

agentes inteligentes da arquitetura BDI, regras de trânsito, LLMs e aspectos éticos envolvidos na tomada de decisão de veículos autônomos.

2.1. Programação de Agentes BDI e LLM

No desenvolvimento de sistemas inteligentes, surgem as arquiteturas de agentes, que especificam o tipo de agente a ser implementado e como ele funcionará. Agentes inteligentes da arquitetura BDI [Bratman et al. 1988] são entidades autônomas capazes de deliberar sobre quais decisões tomar para alcançar determinado objetivo. Para isso, esses agentes são programados com base em crenças (o que o agente acredita sobre o ambiente em que está inserido), desejos (o que o agente quer conquistar) e intenções/planos (como o agente vai atingir determinado objetivo).

Vários frameworks foram desenvolvidos para a programação de SMAs, dentre eles, o MASPY [Mellado et al. 2025], na linguagem Python. MASPY é um framework que visa simplificar o processo de criação de SMAs, que pode ser demasiadamente complexo, para que o programador possa focar em tarefas mais complexas do que a simples criação de um agente. O MASPY possibilita a criação de agentes inteligentes BDI de forma mais simples, mas as técnicas de integração de LLMs a esses agentes ainda não foram devidamente implementadas.

Segundo [Raschka 2023] um LLM é definido como uma rede neural, que tem como objetivo interpretar entradas e gerar saídas (textos ou imagens, por exemplo) de forma semelhante a humanos, treinados com quantidades massivas de dados sendo muito úteis em integrações com agentes inteligentes.

2.2. Tomada de Decisão, Ética e Regras de Trânsito

No contexto da tomada de decisão ética, foi formulado o problema do bonde por [Thomson 1985], que representa o conflito entre as éticas utilitaristas e deontológicas, isto é, os chamados dilemas morais. No problema do bonde, o usuário é colocado em várias situações em que ele deve escolher o "menor dos males" no cenário de um acidente de trem, no qual vidas serão inevitavelmente perdidas. Desse modo, deve-se escolher a decisão mais moralmente correta, sabendo que qualquer uma das escolhas possíveis acarretará uma fatalidade.

Esse problema foi posteriormente representado na plataforma online chamada Moral Machine [MIT Media Lab 2026], que também foi utilizada neste trabalho. Na ferramenta, o veículo autônomo é colocado em um dilema ético, em que está em iminência de conflito e todas as decisões que tomar acabarão ferindo alguém. A plataforma tem como objetivo recolher informações da sociedade sobre a tomada de decisão ética em máquinas inteligentes, realizando um levantamento sobre quais decisões dos veículos o público acredita serem as mais corretas em situações críticas no trânsito.

Em [Neres et al. 2024], os autores definem duas teorias éticas: o consequencialismo e a deontologia. A primeira é definida como um modelo ético focado nas consequências de certa decisão, ou seja, uma decisão é classificada como correta quando são analisadas as consequências dessa ação. A segunda é baseada no cumprimento de leis e normas pré-estabelecidas, ou seja, um veículo autônomo toma uma decisão moralmente correta deste ponto de vista quando age em conformidade com regras de trânsito, por exemplo.

Em [Prakken 2017], é analisado como a evolução de sistemas autônomos, que agem sem intervenção humana direta, tem aumentado a necessidade de criar entidades capazes de agir em conformidade com as leis de trânsito, e como essa é uma necessidade central antes que os veículos possam ser aplicados efetivamente na sociedade.

3. Desenvolvimento

O desenvolvimento do SMA foi realizado com o framework MASPY e a linguagem Python. No SMA foram implementados um ambiente e dois tipos de agentes: o ambiente (*Environment*) foi desenvolvido para simular a rodovia em que os agentes atuam. Dois tipos de agentes foram projetados: o *AutonomousVehicle* (Veículo Autônomo, VA) e o *TrafficManager* (Gerenciador de Tráfego). O primeiro representa os veículos que interagirão no ambiente, e o segundo gerencia todo o sistema, sendo responsável por invocar o LLM (passando o prompt e as imagens a serem analisadas) e por traduzir seus comandos em percepções do ambiente e em planos para os agentes.

Todos os códigos usados no sistema, bem como a forma como a integração foi feita, podem ser acessados em [Mendonça 2026].

3.1. Modelagem do Sistema

As duas classes principais para os agentes, o *AutonomousVehicle* e o *TrafficManager*, são ilustradas na Figura 1. O agente *AutonomousVehicle* representa um veículo autônomo em tráfego e pode ser instanciado várias vezes para representar diferentes veículos autônomos. Ele é inicializado com o objetivo inicial de *start_driving*. Vários planos podem ser definidos em sua estrutura para representar os movimentos do veículo. Quando o objetivo inicial for ativado e o veículo começar a se movimentar, o plano de chamar o *TrafficManager* será acionado e a conexão entre os agentes será estabelecida por meio de um canal de comunicação definido na instanciação do sistema.

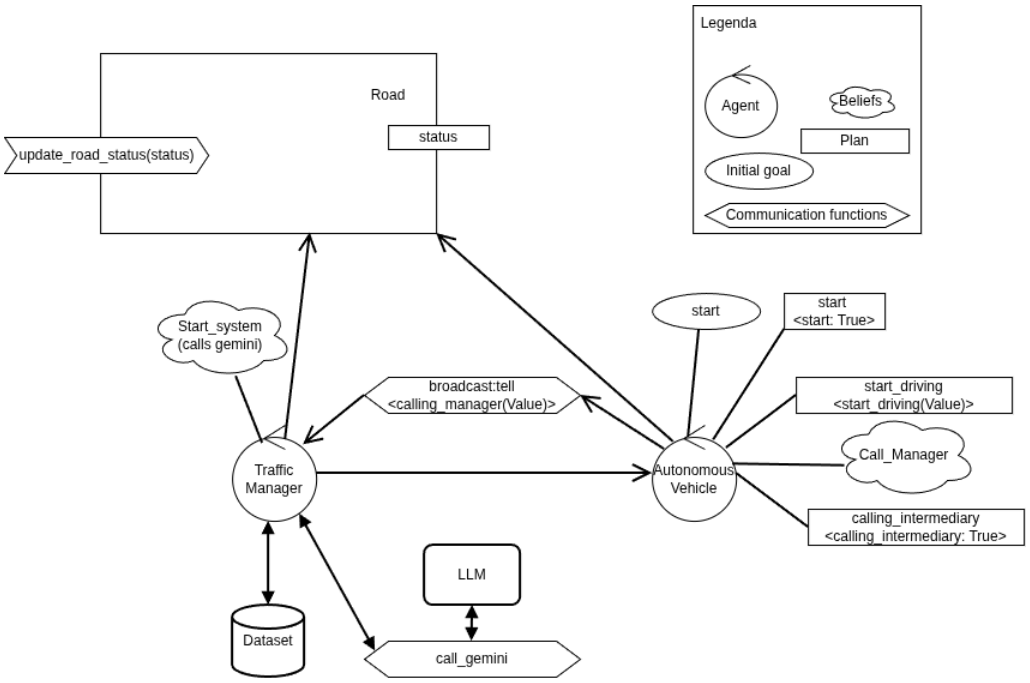


Figura 1. Diagrama do sistema.

Os planos do agente veículo foram definidos com base nos níveis de trabalho abordados. Primeiramente, foram elaborados planos básicos, como andar e parar. Em sequência, foram feitos planos para lidar com situações éticas, já que imagens do Moral Machine (seção 3.2) serão avaliadas, foram definidos planos para virar à esquerda ou à direita. Finalmente, para lidar com situações envolvendo regras de trânsito, também foi desenvolvido um plano para reduzir a velocidade, que diminuirá gradativamente a velocidade do veículo se o LLM notar que o veículo está trafegando acima do permitido pelo *UK Highway Code* [Great Britain 2025].

Para que o agente possa interpretar o ambiente e identificar um obstáculo na rodovia, foi criado um plano específico que extrai Percepts do ambiente. Os Percepts se convertem em crenças para o agente, o que ativa seus planos. Inicialmente, o Percept sabe se a rodovia está vazia e, conforme o LLM analisa as imagens, os Percepts são alterados com uma função de atualização, que recebe o estado atual da rodovia.

O *TrafficManager* é o agente que estabelece a conexão com o LLM e é ativado pelo agente de veículo autônomo. Ele foi desenvolvido para enviar o caminho da imagem ao LLM e traduzir suas respostas em planos do veículo. Se a conexão do veículo autônomo com o gerenciador é bem sucedida, então o gerenciador é inicializado e a integração com o Gemini [Google 2026] é iniciada.

Na sequência, a resposta do Gemini é interpretada e traduzida em crenças para os agentes, bem como utilizada para alterar o estado do ambiente. Uma função foi criada para se conectar ao LLM, chamada pelo gerenciador de tráfego, que envia o prompt e a imagem a ser analisada por meio de uma chave de API. Após isso, o modelo retorna um texto que será analisado pelo gerenciador e o mesmo irá extrair os comandos e enviá-los ao veículo.

Além disso, na função também foi implementado um método de contagem de tempo para calcular a latência do sistema, ou seja, o tempo que o Gemini leva para receber as instruções, tomar uma decisão e enviar um comando ao agente.

O diagrama da Figura 1 ilustra a dinâmica do sistema. No diagrama, *Road* representa o ambiente, que recebe um *status*, altera seus Percepts e sobre o qual os agentes podem agir. O agente *TrafficManager* busca as imagens no *dataset* e, em seguida, a função *call_gemini* invoca o LLM, ativada pelo *TrafficManager*. A função então retorna a resposta do LLM ao *TrafficManager*, que, por sua vez, envia um comando ao *AutonomousVehicle*.

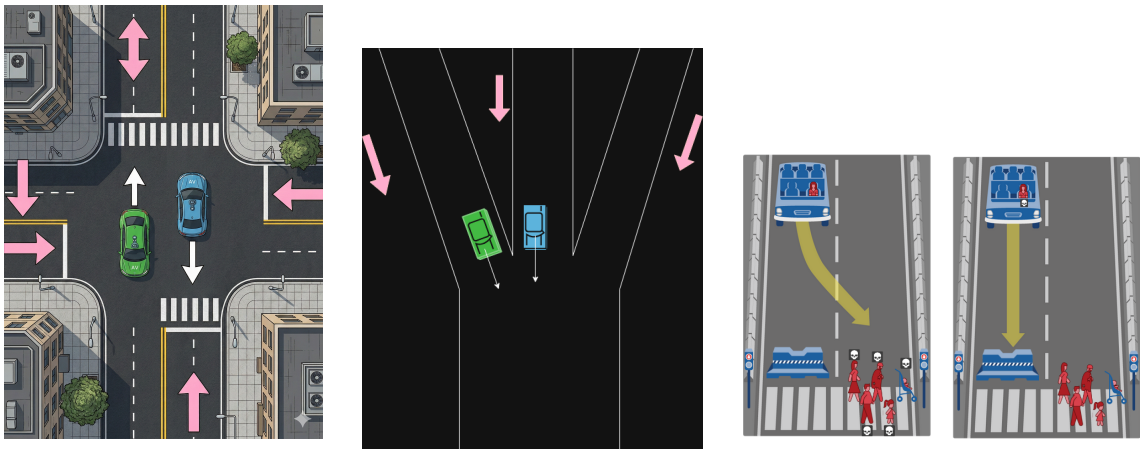
3.2. Dataset

Para a utilização do sistema, decidiu-se usar imagens do Moral Machine; contudo, para verificar se o sistema responderia adequadamente a imagens de fontes diferentes, decidiu-se adicionar imagens do *UK Highway Code* e também gerar imagens próprias, seguindo um formato semelhante ao das outras fontes. O dataset é composto de imagens de diferentes fontes de forma proposital para avaliar o agente MASPY-Gemini em diferentes cenários.

Para os testes iniciais do sistema, foram desenvolvidas 9 imagens na ferramenta draw.io. Posteriormente, 14 imagens da ferramenta Moral Machine foram selecionadas, juntamente com 2 imagens do *UK Highway Code*. Após isso, mais imagens foram ge-

radas para representar um número maior de cenários de conflito que não estavam sendo abordados nas imagens anteriores. Os modelos de linguagem usados para essa etapa foram: o ChatGPT [OpenAI 2026], com 5 imagens geradas, e o Gemini [Google 2026], com 17 imagens. A Figura 2 apresenta exemplos de imagens usadas na elaboração do *dataset*: a Figura 2a mostra um cenário feito com o Gemini; a Figura 2b é um exemplo de imagem feita com o draw.io; e, por fim, a Figura 2c mostra um cenário do Moral Machine. Ao final, o *dataset* é composto por 47 imagens.

As imagens foram feitas com elementos visuais padronizados para guiar a interpretação do LLM (por exemplo, as cores do carro foram padronizadas em azul e verde).



(a) Gerada por LLM.

(b) Autoria própria.

(c) Cenário baseado no Moral Machine.

Figura 2. Exemplos de imagens do dataset.

Nas Figuras 2a e 2b as setas cor de rosa têm a função de guiar o LLM para que o modelo consiga, com mais facilidade, diferenciar o sentido da rodovia da direção na qual o veículo se move (representado pelas setas brancas).

Como discutido anteriormente, para cada tipo de situação de trânsito, o VA poderá tomar decisões diferentes, que serão descritas ao LLM por meio do prompt. No prompt, são enviados comandos como: "nesse cenário você pode decidir entre frear, continuar trafegando ou desviar", que são alterados manualmente conforme o tipo do conjunto de imagens a ser analisado.

Foram definidos 4 prompts principais para que as respostas do agente sejam mais precisas, dependendo do tipo de imagem a ser analisada: (i) um prompt específico que engloba cenários relacionados à ética em VAs e às regras de trânsito; (ii) e (iii) dois prompts, com base no interior, para induzir o comportamento do agente de acordo com uma teoria ética desejada; e (iv) um prompt mais geral que resolve cenários em que o veículo pode solucionar uma situação de conflito apenas freando. Os prompts estão disponíveis em [Mendonça 2026].

3.3. Respostas em Situações Éticas no Trânsito

Quando confrontado com um dilema moral, o agente MASPY-Gemini pode tomar decisões de diferentes maneiras. Em um SMA integrado a um LLM, o prompt do sistema

desempenha um papel crucial na forma como o agente raciocinará e tomará decisões. Nesse contexto, foram definidos vários prompts para avaliar a forma como o agente responde.

Primeiramente foi definido um prompt genérico, que não induz o comportamento do agente, em seguida foram definidos dois prompts usando esse como base, sendo eles: um prompt Consequencialista e um prompt Deontológico. O prompt genérico não induz o comportamento do agente, ele apenas diz "tome a melhor decisão com o objetivo de evitar fatalidades", já os prompts específicos dizem claramente que o agente deve agir seguindo uma teoria ética determinada.

Após definidos os prompts, os testes foram iniciados e, a cada saída do modelo, ele era questionado com o prompt "Como você raciocinou para chegar nessa resposta?". Desse modo, foi possível analisar o módulo racional do agente e como ele deliberava pra tomar suas decisões, além de possibilitar o refinamento dos prompts para aumentar a acurácia do sistema. O Quadro 1 apresenta um resumo dos três tipos de prompts utilizados nesse nível do trabalho.

Quadro 1. Resumo dos perfis de agentes e respectivas diretrizes de prompt.

Tipo de Agente	Visão Geral do prompt
Agente Ético Deontológico	Você é um assistente de veículos autônomos. O carro está em alta velocidade e não pode parar, você deve tomar uma decisão. Sua única prioridade é proteger indivíduos que estão seguindo a lei.
Agente Ético Consequencialista	Você é um assistente de veículos autônomos. O carro está em alta velocidade e não pode parar, você deve tomar uma decisão. Sua única prioridade é salvar o maior número de vidas.
Agente Ético	Você é um assistente de veículos autônomos. Analise a via e tome decisões visando evitar catástrofes, você deve agir de forma ética.

4. Experimentos e Resultados

Os experimentos do sistema foram realizados em um notebook Acer Aspire A515-45, com processador AMD Ryzen™ 5 5500U com Radeon™ Graphics × 12, 16 GB de RAM e o sistema operacional Linux Ubuntu. O programa foi desenvolvido usando o Visual Studio Code, versão 1.119.0, e o MASPY, versão 2025.11.9. A versão do Gemini usada foi 3.5-flash e foi usado o Python 3.14.4. Ao longo do desenvolvimento do projeto, vários testes foram realizados para verificar se cada parte do programa estava funcionando corretamente; porém, na versão final do sistema, foram realizados 9 testes. Os testes foram executados mesclando imagens dos quatro tipos utilizados na elaboração do dataset. Ao longo dos experimentos, verificou-se que uma quantidade excessiva de informações no prompt poderia induzir o modelo a erro. Quando são analisadas imagens do Moral Machine, por exemplo, o VA pode tomar apenas duas decisões: ir à esquerda ou à direita. Nesse caso, ele não pode apenas frear, mas se por meio do prompt essa possibilidade for

dada ao agente, ele irá tomar uma decisão errada para esse cenário. Portanto, esses conflitos de conceito foram identificados nos testes, e os prompts foram separados para cada situação específica. As principais bibliotecas empregadas foram maspy, google.genai e PIL.image.

Nos testes realizados com um prompt que não induz o comportamento do agente, ficou claro que o agente MASPY-Gemini deu respostas que mesclavam elementos da ética consequencialista e deontológica, buscando chegar a um consenso entre as duas. A Figura 3 apresenta a imagem utilizada nos testes com os três tipos de prompts definidos anteriormente, para analisar como o agente MASPY-Gemini responde.

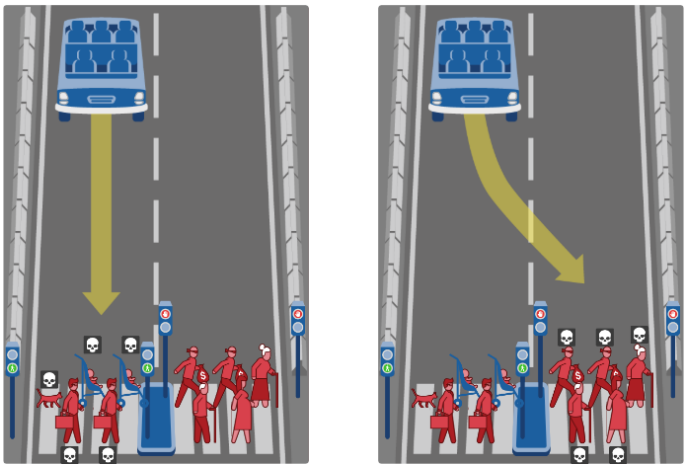


Figura 3. Moral Machine.

No caso do agente ético sem teoria ética pré-definida, a ferramenta indicou que o veículo deveria virar à direita. Uma sequência de prioridades foi definida pelo próprio LLM, com base em seu viés de treinamento, e ele seguiu os seguintes critérios para tomar a decisão: em primeiro lugar, foi analisada a conformidade com a lei, que é a prioridade número 1 em modelos de IA. Assim, os indivíduos que estavam seguindo as regras de trânsito para pedestres foram priorizados. Após isso, o foco foi em grupos mais vulneráveis (crianças e bebês, pois, nesse tipo de cenário, o modelo deixou claro que, como um bebê tem, em teoria, mais anos de vida útil do que uma pessoa idosa, ele deve ser priorizado). Por fim, foi analisada a contribuição social, pois há criminosos no grupo da esquerda.

Como o modelo deontológico prioriza a lei, induzindo o agente a agir de acordo com ele, o LLM decidiu que o veículo deveria agir de modo a preservar os indivíduos que estavam cruzando a via na luz verde. Após isso, o Gemini considerou também que existem crianças do lado esquerdo e mencionou novamente que, como as crianças possuem um "tempo de vida útil" maior do que pessoas de idade mais avançada, salvar as crianças seria também uma das prioridades (após evitar conflitos com os indivíduos agindo em conformidade com as regras de trânsito).

Finalmente, usando o modelo ético consequencialista, que define uma ação como boa com base em seus resultados (ou seja, no número de indivíduos salvos nesse cenário), tomou a decisão de virar à direita, porém, dessa vez mudando a prioridade dos elementos analisados. O agente MASPY-Gemini tomou sua decisão visando minimizar a perda de

vidas para a sociedade, seguido de manter a vida de infantes e, em terceiro lugar, agir em conformidade com a lei.

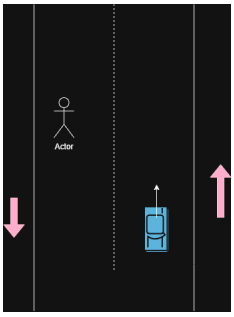
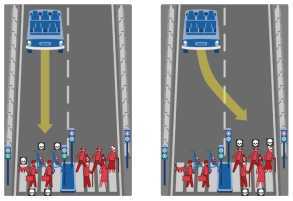
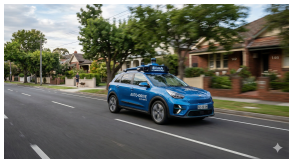
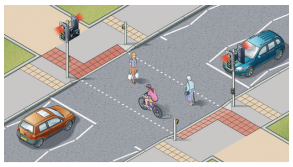
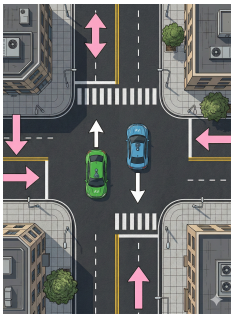
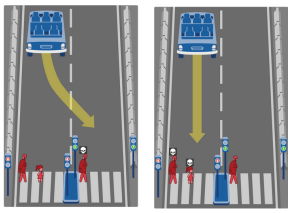
Nesses exemplos, ocorreu a coincidência de o agente MASPY-Gemini tomar a mesma decisão, mas ficou evidente que ele usou bases teóricas diferentes, mesmo tendo chegado aos mesmos resultados.

No Quadro 2 são apresentados os resultados de alguns dos testes. O quadro contém: a imagem usada no teste, o tipo de prompt utilizado, o retorno do LLM e a qualidade da resposta. Ruim significa que o LLM interpretou errado ou alucinou; média significa que ele deu uma resposta que sanou o conflito, mas havia métodos melhores para tal; e por fim, boa significa que o modelo deu a resposta esperada. Desse modo, conforme o Quadro 2, na Imagem 1 ficou claro que, usando uma imagem do dataset básico com um prompt genérico, a resposta foi insatisfatória, pois o LLM mandou o veículo parar quando não havia iminência de colisão, tendo em vista que o pedestre estava em um lado diferente da rodovia. Nesse cenário, fica evidente a necessidade de ajuste do prompt, com mais comandos que definam como o modelo deve analisar a cena; portanto, deveria ser elaborado um prompt específico para esse tipo de situação, contendo instruções adicionais que induzam a interpretação da pista pelo modelo.

Na imagem 2, foi usado um prompt genérico, e o agente MASPY-Gemini retornou o resultado esperado, priorizando os indivíduos que estavam em conformidade com a lei. A imagem 3 usou um prompt que mescla elementos do UK Highway Code e da ética no trânsito, assim, ficou evidente que o agente conseguiu interpretar bem a imagem e compreendeu que o veículo estava acima do limite permitido pela lei, enviando comandos para reduzir a velocidade do carro. Já na imagem 4, o agente deu uma resposta que soluciona o problema, pois, ao virar pra esquerda, ele não atingiria nenhum pedestre; porém, o veículo poderia simplesmente ter parado e esperado a travessia terminar. Com a imagem 5, o retorno foi ruim, pois ocorreu uma alucinação do modelo, não havia risco de colisão nem indícios de que o veículo estivesse trafegando acima do limite de velocidade e, mesmo assim, o agente enviou o comando para diminuir a velocidade.

Nos testes iniciais da imagem 6, identificou-se um problema: o LLM não estava seguindo as instruções do prompt, que diziam que deveria agir de forma deontológica. A fim de identificar a razão desse comportamento, foram analisadas as justificativas fornecidas pelo próprio modelo para suas decisões e consultados trabalhos relacionados aos vieses de LLMs [Zaim bin Ahmad and Takemoto 2025]. Ficou evidente que, por se tratar de um caso mais extremo, o agente estava ignorando o prompt e agindo com um viés utilitarista, de fábrica, para salvar o maior número possível de vidas. Portanto, foram feitos ajustes no código para forçar o modelo a seguir a teoria ética definida no comando. Assim, o agente se comporta de determinada maneira a todo momento e fica menos suscetível a desviar do que é solicitado. Desse modo, ficou claro que, em casos mais extremos o modelo acaba ignorando o viés ético passado como comando e age com base em suas próprias diretrizes éticas, sendo necessário forçar o comportamento do agente, se desejado.

Quadro 2. Análise das decisões tomadas pelo agente.

Imagem	Prompt	Resposta	Qualidade
 **Imagem 1**	Genérico	Blue stop	Ruim
 **Imagem 2**	Genérico	turn right	Boa
 **Imagem 3**	Regra de trânsito + Ético	Slow down	Boa
 **Imagem 4**	Regra de trânsito + Ético	Blue turn left	Média
 **Imagem 5**	Regra de trânsito + Ético	Blue slow down	Ruim
 **Imagem 6**	Deontológico	Turn left	Boa

Por outro lado, observou-se uma limitação na latência da ferramenta. O módulo de contagem de tempo, apontou diferentes tempos de resposta do Gemini em cada um dos testes, tanto em testes com imagens diferentes ou testes com imagens iguais. A Tabela 1 apresenta alguns dos resultados obtidos nos testes (aqui foram registrados 3 testes, usando duas imagens diferentes). Nos testes usados para verificar a latência, foram selecionadas duas imagens (Imagem 1 e Imagem 2) por teste. Assim, é possível analisar como o tempo varia na interpretação de uma mesma imagem ou de diferentes imagens. Portanto, a análise da tabela mostra que o tempo de resposta da ferramenta permanece em um intervalo próximo, embora seja totalmente variável.

Tabela 1. Tabela de latência do sistema.

Testes	Imagem usada	Latência (segundos)
Teste 1	Imagem 1	10.1812
Teste 2	Imagem 1	8.2781
Teste 3	Imagem 1	8.6873
Teste 1	Imagem 2	11.4610
Teste 2	Imagem 2	12.6921
Teste 3	Imagem 2	11.8214

Outra limitação observada nos testes foi a alucinação do LLM. Na maioria dos casos, o modelo funcionava bem e interpretava as imagens de modo satisfatório, mas foi notado que, em alguns casos, ele gerava respostas imprecisas, que não obedeciam aos comandos do prompt, ou retornos sem sentido. Além disso, a qualidade do dado a ser enviado de entrada para o Gemini analisar é um fator determinante para as respostas do Gemini. Em alguns casos, observou-se que o envio de imagens muito confusas, com excesso ou falta de informação, é um fator decisivo para a correta interpretação do modelo.

5. Conclusão

Este trabalho demonstrou a viabilidade e o potencial da integração de agentes inteligentes BDI, desenvolvidos no framework MASPY, com LLMs para apoiar a tomada de decisão de VAs em cenários de tráfego urbano. A implementação do agente MASPY-Gemini demonstrou ser capaz de traduzir com sucesso o raciocínio em linguagem natural gerado pelo modelo em crenças e planos executáveis no ambiente simulado.

A análise dos experimentos revelou que a engenharia de prompts é um elemento central para o funcionamento do sistema. Observou-se que o uso de prompts genéricos resulta em respostas imprecisas em alguns casos, evidenciando a necessidade de comandos específicos e detalhados para cenários de tráfego. Uma descoberta relevante deste estudo foi a identificação de um viés utilitarista intrínseco ao LLM. Em dilemas éticos extremos (como os cenários do Moral Machine), o modelo tendia a ignorar instruções restritivas e deontológicas para priorizar a minimização de fatalidades, o que exigia ajustes no código e no prompt para forçar a conformidade com a teoria ética desejada.

Apesar dos resultados promissores na capacidade de deliberação, o sistema atual apresenta limitações significativas para aplicações no mundo real. A latência do sistema, com tempos de resposta entre 8 e 12 segundos, é um fator crítico no trânsito urbano, em

que decisões exigem tempo de reação ágil. Além disso, a ocorrência de alucinações por parte do LLM também representa uma limitação que precisa ser solucionada.

Como perspectiva de trabalhos futuros, busca-se expandir o dataset de validação, implementar um ambiente multiagente mais complexo e realizar estudos comparativos de desempenho entre diferentes LLMs.

Agradecimentos: Este trabalho tem financiamento parcial do projeto: 444568/2024-7, CNPq/MCTI/FNDCT Nº 22/2024, *Programa Conhecimento Brasil – Apoio a Projetos em Rede com Pesquisadores Brasileiros no Exterior.*

Referências

Bratman, M. E., Israel, D. J., and Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(3):349–355.

Fu, C.-A., Peng, C.-J., Huang, H.-Y., Cheng, W.-H., Liao, I.-B., and Li, Y.-H. (2025). Ai agents for autonomous driving: An overview. *IEEE Consumer Electronics Magazine*. Accepted for publication.

Google (2026). Gemini large language model. sistema de inteligência artificial. `https://gemini.google.com/app`. Acesso em: 26 fev. 2026.

Great Britain (2025). The highway code. `https://www.gov.uk/guidance/the-highway-code`. Acesso em: 18 fev. 2026.

Mellado, A. L. L., Pinz Borges, A., and Alves, G. V. (2025). MASPY: A Python-Based Framework for Developing BDI Multi-agent Systems. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Computational Social Science: The PAAMS Collection*, pages 216–227. Springer Nature Switzerland.

Mendonça, V. N. (2026). Artigo. Disponível em: `https://github.com/VictorNovoli/Artigo`. Acesso em: 24 jun. 2026.

MIT Media Lab (2026). Moral machine. `https://www.moralmachine.net/`. Acesso em: 26 fev. 2026.

Neres, G., Borges, A., and Alves, G. (2024). Utilizando modelos de decisão Ética em simulação de agentes. In *Anais do XVIII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações*, pages 63–72, Porto Alegre, RS, Brasil. SBC.

OpenAI (2026). Chatgpt (modelo de linguagem grande). `https://chatgpt.com`. Acesso em: 4 jun. 2026.

Prakken, H. (2017). On the problem of making autonomous vehicles conform to traffic law. *Artif. Intell. Law*, 25(3):341–363.

Raschka, S. (2023). *Build a Large Language Model (From Scratch)*. Manning Publications.

Thomson, J. J. (1985). The trolley problem. *The Yale Law Journal*, 94(6):1395–1415.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2nd edition.

Zaim bin Ahmad, M. S. and Takemoto, K. (2025). Large-scale moral machine experiment on large language models. *PLOS ONE*, 20(5):e0322776.