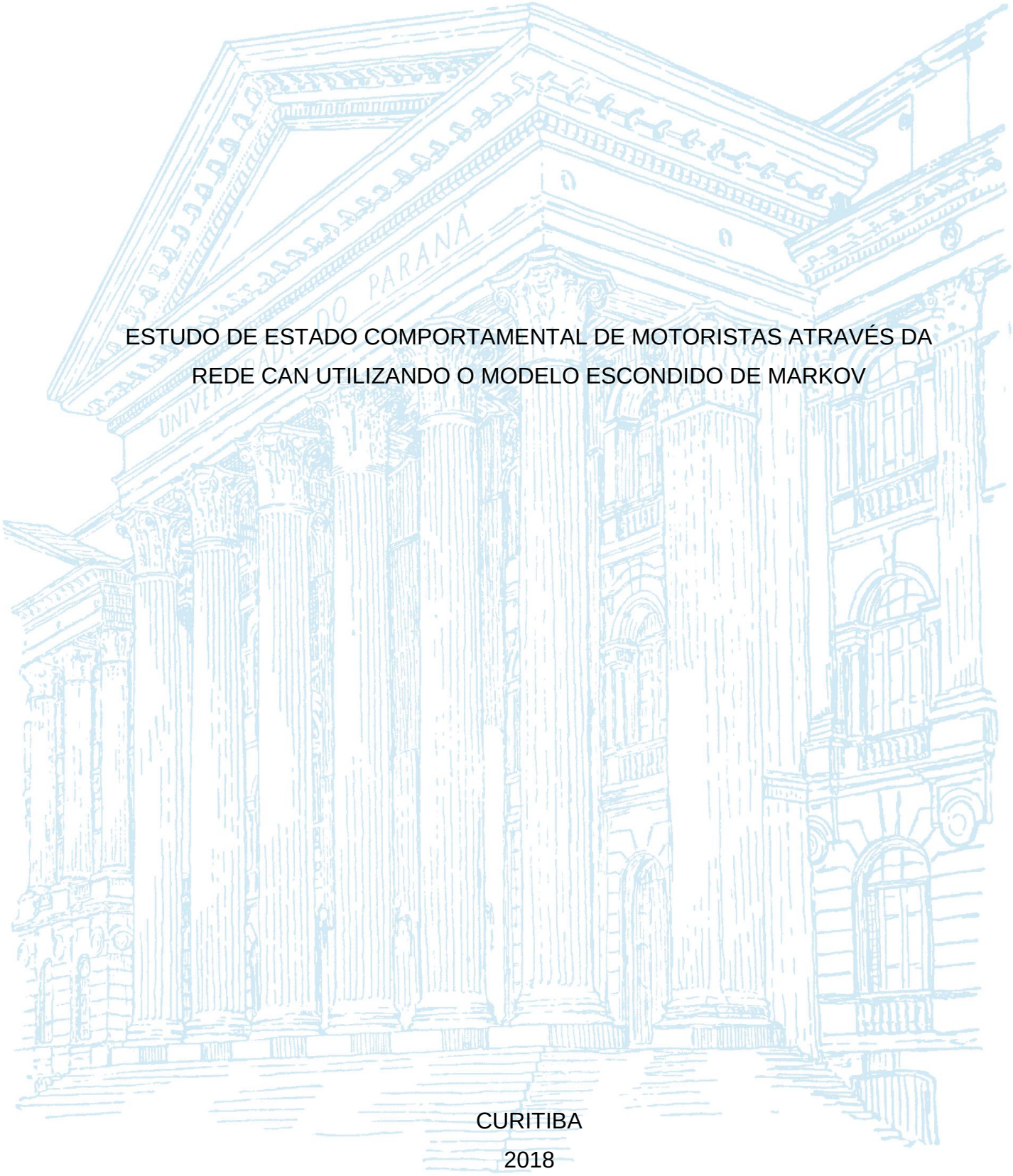
UNIVERSIDADE FEDERAL DO PARANÁ

ANDREY DE OLIVEIRA BOARÃO



ESTUDO DE ESTADO COMPORTAMENTAL DE MOTORISTAS ATRAVÉS DA REDE CAN UTILIZANDO O MODELO ESCONDIDO DE MARKOV

CURITIBA

2018

ANDREY DE OLIVEIRA BOARÃO

ESTUDO DE ESTADO COMPORTAMENTAL DE MOTORISTAS ATRAVÉS DA REDE CAN UTILIZANDO O MODELO ESCONDIDO DE MARKOV

Trabalho de conclusão de curso apresentado ao curso de Engenharia Elétrica, Setor de Tecnologia, da Universidade Federal do Paraná, como requisito à obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Carlos Marcelo Pedroso

CURITIBA

2018

TERMO DE APROVAÇÃO

ANDREY DE OLIVEIRA BOARÃO

ESTUDO DE ESTADO COMPORTAMENTAL DE MOTORISTAS ATRAVÉS DA REDE CAN UTILIZANDO O MODELO ESCONDIDO DE MARKOV

Trabalho de Conclusão de Curso apresentado ao curso de Graduação em Engenharia Elétrica Setor de Tecnologia, Universidade Federal do Paraná, como requisito à obtenção do título de Bacharel em Engenharia Elétrica.

_____

Prof. Dr. Carlos Marcelo Pedroso
Orientador – Departamento de Engenharia Elétrica, UFPR

_____

Prof. Dr. Henri Frederico Eberspacher
Departamento de Engenharia Elétrica, UFPR

_____

Prof. M. Sc. Ricardo Schumacher
Departamento de Engenharia Elétrica, UFPR

Curitiba, 04 de dezembro de 2018.

AGRADECIMENTOS

Agradeço aos meus pais, Airton e Mara, aos meus irmãos, Natacha e Guilherme e à minha noiva, Andressa, por todo o incentivo, apoio e força proporcionados.

RESUMO

Após a regularização do trabalho do motorista de veículo de transporte de carga no Brasil, exige-se que o motorista faça o controle e registro de sua jornada. Com o avanço tecnológico crescente em veículos e a grande quantidade de sensores presente neles, identificou-se a possibilidade de analisar as mensagens presentes na rede CAN, para definição do estado comportamental do motorista durante seu período de trabalho e descanso. Para tanto, o Modelo Oculto de Markov foi utilizado para o cálculo probabilístico do estado comportamental do motorista em função do tempo. Foram feitos registros da rede CAN e o tratamento de sinais previamente selecionados, de modo a aplicar o treinamento do sistema e sua validação. No estudo, foram definidos três estados, Direção, Repouso e Abastecimento. Além disso, foram estudadas situações de taxa de amostragem reduzida e remoção de sinais com alta correlação. Para a validação do sistema, os dados obtidos foram comparados com uma referência. Comparando os resultados, identificou-se alta precisão no sistema com maior taxa de amostragem, mesmo considerando o atraso gerado entre alteração de estados. Nos outros sistemas, obteve-se uma precisão satisfatória, mas com baixo tempo de transição entre estados.

Palavras-chave: Rede CAN; Modelo Oculto de Markov; Modelo Escondido de Markov; Controle de jornada de trabalho de motorista.

ABSTRACT

After the regulation of transport vehicle driver's work in Brazil, it is demanded for the driver to control and register its journey. With the advanced technology in vehicles and its high quantity of sensors, it has been identified the possibility to analyze the messages within CAN bus to define driver's behavior state during its work and rest period. Hidden Markov Model has been used to calculate the probability of driver's behavior state as a function of time. CAN bus have been recorded and its signals were previously defined to apply the learning of the system and its validation. In this work, it has been defined three driver's state, Driving, Rest and Refueling. It has been analyzed different sampling rate and signals with high correlation removal. To validate the system, the data obtained was compared with a reference file. By comparing results, it has been identified high accuracy in the system with higher sampling rate, even considering delay between state changes. In other systems, a satisfactory accuracy was obtained, but with lower delay between state transition.

Keywords: CAN bus; Hidden Markov Model; Driver journey management.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Com a busca pela inovação, atualização e a demanda do mercado, as montadoras de veículos automotores estão investindo cada vez mais em segurança, conforto, economia, luxo e entretenimento. Para tanto, os veículos atuais possuem redes de comunicação entre suas unidades eletrônicas (ECUs), as quais recebem e interpretam sinais provenientes da grande quantidade de sensores presentes nos veículos.

A partir da regulamentação da profissão do motorista não autônomo, de veículos automotores da categoria de transporte de cargas e passageiros, lei número 13.103 de 2015, exige-se que o empregado faça o controle de sua jornada e registre-a por meio físico ou eletrônico, de modo a respeitar as exigências aplicadas pela legislação trabalhista.

Este trabalho propõe a definição do estado comportamental do motorista utilizando os sensores já presentes em um veículo automotor de transporte de carga. Tal definição será feita a partir do estudo dos sinais em função do tempo, sendo comparado com dados base obtidos a partir do comportamento do motorista em estados pré-definidos. Para estimar seu efetivo estado, será utilizado o Modelo Oculto de Markov, o qual fará uma análise dos sinais recebidos, e definirá o estado comportamental a partir de uma análise probabilística.

Identificou-se estudos e análises comportamentais de motoristas através de dados de veículos. Alguns estudos focaram na análise e definição do comportamento agressivo ou defensivo de motoristas. Outros, um levantamento de perfil de motorista, de modo a estimar seu comportamento em pistas e rodovias.

1.1 JUSTIFICATIVA

Este trabalho busca obter informações ocultas a partir de sinais presentes em um veículo automotor. Além da estimativa do estado comportamental, tais informações podem ser utilizadas como um estudo sobre como o motorista se comporta em relação ao caminhão durante as diferentes atividades realizadas no veículo.

1.2 OBJETIVOS GERAIS

O trabalho de conclusão de graduação tem o objetivo geral de interpretar o estado comportamental do motorista a partir das informações presentes nos barramentos CAN do veículo. Particularmente definindo se o mesmo encontra-se em estados típicos na rotina de trabalho de um motorista como dirigindo, abastecimento, manutenção, descanso, ou outros estados.

Para atingir tal objetivo, serão registrados comportamentos de motoristas utilizando dispositivos de análise de rede CAN e seus dados serão aplicados sobre um sistema de aprendizado, o qual interpretará, a partir do aprendizado, qual o estado comportamental que o motorista esteve em momentos específicos.

1.3 OBJETIVOS ESPECÍFICOS

Como objetivos específicos do trabalho, podem ser citados:

- Levantamento bibliográfico do material de estudo;
- Registro de dados de motoristas;
- Aplicação do sistema de aprendizado;
    - Discretização de dados;
    - Treinamento do sistema;
    - Análise de resultados;
    - Validação.

Obter um sistema confiável de estimativa de estado comportamental de motoristas, através da leitura da rede CAN do veículo utilizando o Modelo Oculto de Markov como sistema de aprendizado.

1.4 METODOLOGIA

Com o uso de equipamentos de análise de mensagens CAN, um motorista qualificado fará simulações de situações comuns no trabalho de um motorista de caminhão. Pode-se usar como exemplo, dirigir a um destino, descansar, reabastecer, aguardar a carga/descarga, etc. Seu comportamento no interior do veículo será gravado com o uso de um equipamento de análise de mensagens CAN. Tal informação será tratada e interpretada por um software, o qual aplicará sobre o

sistema um aprendizado sobre as informações recebidas. Com este aprendizado, será analisado se o sistema conseguirá definir corretamente o comportamento do motorista em situações similares às demonstradas anteriormente.

2 REVISÃO DE LITERATURA

O trabalho se baseia no estudo da rede CAN de um veículo, especificamente um caminhão, para estimar o estado comportamental de um motorista durante seu horário de trabalho e descanso. A definição, ou estimativa, do estado será feita através da aplicação do Método Oculto de Markov.

2.1 REDE CAN

Desenvolvido pela empresa alemã *Robert Bosch GmbH* em 1986, a rede CAN (*Controller Area Network)* é um protocolo de comunicação veicular serial síncrono. Focado na comunicação entre unidades de controle, as mensagens lançadas ao barramento por uma ou mais unidades são recebidas por todas as outras (*multicast)* e lidas conforme sua prioridade*(multi-mestre).* Sua comunicação é feita através de dois pares trançados, CAN HIGH (3,5V) e CAN LOW (1,5V). Ao final de cada barramento, há dois terminadores, os quais são resistores de 120Ω, utilizados para garantir a perfeita propagação do sinal elétrico, sendo exemplificado pelos retângulos pretos na Figura 1.

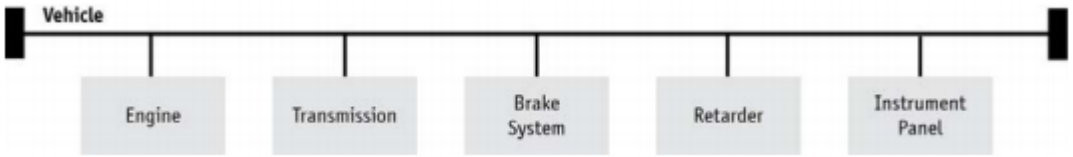


Figura 1 - Exemplo de rede CAN.
Fonte: vector.com

Para que todos os módulos compreendam as mensagens contidas na rede, eles devem possuir o que é conhecido como *Dicionário de dados*, o qual é uma matriz que demonstra quais mensagens são de responsabilidade de quais módulos, a frequência de atualização de valores, intervalo de transmissão, entre outras informações. Todos os módulos de um sistema devem possuir o mesmo dicionário, o qual é implementado na unidade via *software*.

2.2 SAE J1939

Acima da camada da rede CAN, tem-se o protocolo SAE J1939, o qual é utilizado no ramo automotivo, especialmente em ônibus e caminhões. Este protocolo possui uma estrutura de mensagem com *identificador* de ECU e o campo de dados. Dentro do identificador, encontram-se os seguintes campos:

Figura 2 - Conteúdo do identificador da mensagem CAN.
Fonte: [4]

- Prioridade, um menor valor binário corresponde a uma maior prioridade.
- Página de dados, contém informações a respeito do grupo de parâmetro.
- PDU e PDU específico, formam o campo PGN (*Parameter Group Number*).
- Endereço de origem, indica o endereço da ECU que está enviando a mensagem.

Um valor de PDU abaixo de 240 em decimal, quer dizer que a mensagem é enviada especificamente para um módulo. Para valores iguais ou maiores que 240, a mensagem é um *broadcast* (também conhecido como *multicast*) onde todas as ECUs recebem a informação enviada.

Os Números de Grupos de Parâmetros, também conhecido como PGN, são grupos de mensagens/parâmetros direcionados a uma informação específica. Por exemplo, o PGN da temperatura do motor contém a informação de temperatura do líquido de arrefecimento do motor, do óleo, combustível, etc.

2.3 CADEIAS DE MARKOV

A teoria da probabilidade moderna estuda processos aleatórios onde o conhecimento de resultados passados influencia previsões de resultados futuros [11]. Em 1907, Andrei Andreyevich Markov iniciou os estudos em processos onde o resultado de um experimento pode afetar o resultado de um próximo experimento.

Tais processos ficaram conhecidos como Cadeias de Markov, os quais são métodos estocásticos que têm a propriedade de que a probabilidade de transição para um estado futuro, quando o atual é conhecido, não é alterada pelo conhecimento dos estados anteriores [7].

Supondo que tenhamos um conjunto de estados S=$\{s_1, s_2, \dots, s_n\}$. O processo inicia-se em um desses estados e varia sucessivamente de um estado para outro. Se a cadeia encontra-se em s_i e muda para outro estado s_j, a probabilidade de mudança, ou *probabilidade de transição*, é denominada P_{ij} o qual independe do estado anterior do sistema.

A Figura 3 demonstra um exemplo clássico do modelo, onde se analisa as condições meteorológicas e as probabilidades de transição entre si, tanto da troca de estado, quanto a probabilidade de manter-se no mesmo.

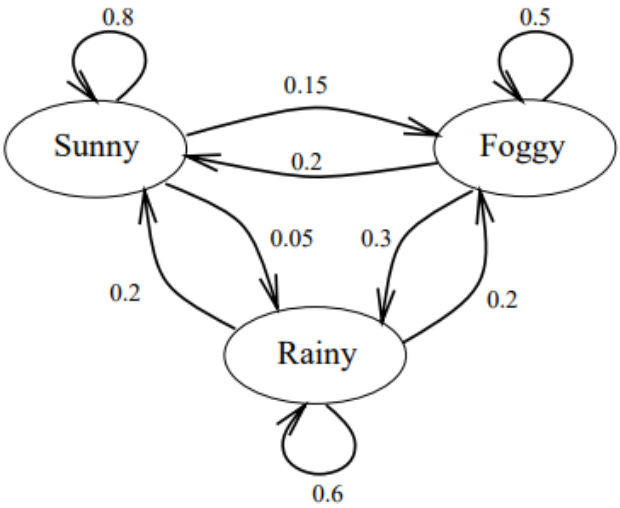


Figura 3 - Modelo de processo Markoviano.

Fonte: [9]

Através da análise do sistema de estudo, pode-se elaborar uma matriz quadrada com as probabilidades correspondentes a cada estado e sua transição. Como exemplo, a Tabela 1 é a matriz de probabilidade de transição para o sistema apresentado na Figura 3. Onde os valores da primeira coluna representam a probabilidade de transição de diferentes estados para o estado ensolarado. Seguindo a mesma lógica, os valores das colunas conseguintes são relacionados à transição dos diferentes tipos de tempo para nebuloso e chuvoso.

Estados		Estado de amanhã		
		Ensolarado	Nebuloso	Chuvoso
Estado atual	Ensolarado	80%	15%	5%
	Nebuloso	20%	50%	30%
	Chuvoso	20%	20%	60%

Tabela 1 - Exemplo de matriz de transição para a probabilidade do estado de amanhã baseado no estado atual.

Fonte: [9].

2.4 MODELO OCULTO DE MARKOV

Descrito em meados de 1960 por Leonard E. Baum, o Modelo Oculto de Markov (MOM) é uma extensão das Cadeias de Markov onde a observação é uma função probabilística do estado [5]. Ou seja, um processo estocástico não observável, o qual pode ser observado por outro processo estocástico que produza a sequência de observação.

De forma a exemplificar o MOM, pode-se estender o exemplo anterior assumindo que os estados temporais são não observáveis. Supondo que seja possível identificar a probabilidade de estar no estado chuvoso apenas através da observação da presença ou não de um guarda-chuva em um determinado local.

	Probabilidade na presença de guarda-chuva
Ensolarado	10%
Chuvoso	80%
Nebuloso	20%

Tabela 2 – Exemplo de probabilidade de emissão do Modelo Oculto de Markov.

Fonte: [9].

Na Tabela 2, é demonstrado o exemplo de probabilidade de emissão que a presença de um guarda-chuva implica sobre os estados temporais não observados.

Neste caso, a probabilidade de transição entre os estados não observáveis são as mesmas das já demonstradas.

O MOM possui elementos que compõem o processo, sendo eles:

1. Um espaço de estados não observáveis *N*= $\{n_1, n_2, \dots, n_n\}$. Mesmo que não seja possível observar os estados, para muitas aplicações há algum significado físico conectado a eles.
2. Um conjunto de observáveis distintos *M*= $\{m_1, m_2, \dots, m_n\}$, sendo estes o modelo da saída física do sistema.
3. Os valores de probabilidade de transição de estados de *N, A*= $\{a_{ij}\}$.
4. Os valores de distribuição de probabilidade de emissão no estado j, *B*= $\{b_j\ (k)\}$.
5. O estado inicial de distribuição *Π* = $\{\pi_i\}$.

Dado valores apropriados para as variáveis demonstradas, pode-se utilizar o Modelo Oculto de Markov para gerar a sequência de observação

$$O = O_1 O_2 \dots O_T \qquad (1)$$

Onde T é o número de observações na sequência.

A completa especificação do MOM exige, portanto, a definição dos parâmetros N e M, além das medidas de probabilidade A, B e *Π.* Podendo ser utilizada a notação (2) para a completa indicação dos parâmetros do modelo.

$$\lambda = (A, B, \Pi) \qquad (2)$$

2.4.1 Problemas e soluções do Modelo Oculto de Markov

De acordo com Rabiner [5], há três grandes problemas básicos no MOM que devem ser analisados para que o sistema seja eficientemente aplicado em situações reais:

1. *O cálculo da probabilidade de sequências observáveis;*

2. *A sequência ótima de estados;*
3. *O ajuste do modelo.*

Tais problemas são resolvidos através do uso de algoritmos que serão demonstrados.

2.4.1.1 Solução do problema 1

O cálculo da probabilidade de uma sequência de observação, como demonstrado em (1), a partir do modelo (2), ou P(O|λ), pode ser feito de maneira direta, enumerando todas as possibilidades de transição de estado, ou através do uso de algoritmos.

Considerando que se tenha a seguinte sequência de estados:

$$Q = q_1 q_2 \dots q_T \tag{3}$$

Onde q_1 é o estado inicial, e T é o número de estados analisados. A probabilidade da sequência de observações O para a sequência de estados (3) é

$$\mathrm{P}(\mathrm{O}|\mathrm{Q},\lambda) = \prod_{t=1}^{T} \mathrm{P}(O_t|Q_t,\lambda) \tag{4a}$$

Ou seja,

$$\mathrm{P}(\mathrm{O}|\mathrm{Q},\lambda) = b_{q_1}(O_1).b_{q_2}(O_2) \dots b_{q_t}(O_t) \tag{4b}$$

A probabilidade de sequência de estados Q é escrito como:

$$\mathrm{P}(\mathrm{Q}|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T} \tag{5}$$

E a probabilidade de ocorrência de Q e O simultaneamente é

$$\mathrm{P}(\mathrm{O},\mathrm{Q}|\lambda) = \mathrm{P}(\mathrm{O}|\mathrm{Q},\lambda)\mathrm{P}(\mathrm{Q},\lambda) \tag{6}$$

Assim, a probabilidade de O é obtida através do somatório de (6) em todas as possíveis sequências de estado q, ou seja:

$$\mathrm{P}(\mathrm{O}|\lambda) = \sum_{Q} \mathrm{P}(\mathrm{O}|\mathrm{Q},\lambda)\mathrm{P}(\mathrm{Q}|\lambda) \qquad (7)$$

$$\mathrm{P}(\mathrm{O}|\lambda) = \sum_{q_1,q_2,\ldots,q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \ldots a_{q_{T-1} q_T} b_{q_T}(Q_T) \qquad (8)$$

A interpretação das equações (7) e (8), nos mostra o seguinte: No tempo inicial (T=1), tem-se o estado q_1 com a probabilidade π_{q_1}, a qual gera o símbolo O_1 com a probabilidade $b_1(O_1)$. O tempo, então, passa para t=2, onde a probabilidade de transição do estado q_1 para o estado q_2 é $a_{q_1 q_2}$, gerando O_2 com a probabilidade $b_2(O_2)$. Esta lógica ocorre até o estado O_T.

O cálculo de $\mathrm{P}(\mathrm{O}|\lambda)$, utilizando esta definição direta, envolve operações na ordem 2T*$N^T - 1$, onde N são os estados e T as posições no tempo. Ou seja, o cálculo se torna possível, mas inviável até mesmo para pequenos valores de N e T.

De forma a calcular a probabilidade de forma mais eficiente, utiliza-se o algoritmo *forward-backward.* Sendo apenas a primeira parte (*forward)* necessária para o cálculo.

Considere a variável *forward* $\alpha_t(\mathrm{i})$, definida como:

$$\alpha_t(\mathrm{i}) = \mathrm{P}(O_1 O_2 \ldots O_t, q_t = S_i|\lambda) \qquad (9)$$

A resolução de $\alpha_t(\mathrm{i})$ é feita por três passos:

1) Inicialização: A qual inicia a probabilidade forward como a junção de probabilidades entre o estado S_1 e a observação inicial O_1. Representado pela equação (10).

$$\alpha_1(\mathrm{i}) = \pi_i b_i(O_1)\ , \qquad 1 \leq \mathrm{i} \leq \mathrm{N} \qquad (10)$$

2) Indução: Ilustrado pela equação (11). Visto que $\alpha_t(\mathrm{i})$ é a probabilidade de junção do evento em que as observações

parciais $O_1 O_2 \dots O_t$ são observadas e o estado no tempo t é S_i. Ao multiplicar $\alpha_t(i)a_{ij}$, calcula-se a transição do estado S_i para S_j no tempo t+1. Somando o produto por todas as N possibilidades, resulta na probabilidade do estado S_j no tempo t+1 acompanhado das observações parciais anteriores. Assim, $\alpha_{t+1}(j)$ é obtido através da multiplicação do valor somado obtido por $b_j(O_{t+1})$.

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)a_{ij}]\, b_j(O_{t+1}), \quad 1 \leq t \leq T-1 \qquad 1 \leq j \leq N \tag{11}$$

3) Terminação: O último passo é definido pela soma das variáveis *forward* finais $\alpha_T(i)$.

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{12}$$

O algoritmo *forward* requer cálculos de ordem N^2T, exigindo menor esforço computacional se comparado ao método demonstrado anteriormente.

Analogicamente, tem-se o algoritmo *backward* que é o reverso do algoritmo *forward*, definido de acordo com a equação (13).

$$\beta_t(i) = P(o_{t+1} o_{t+2} \dots o_T | q_t = s_i, \lambda) \tag{13}$$

Sendo $\beta_t(i)$ a probabilidade de observação parcial da sequência de t+1 à T, iniciando no estado s_i.

2.4.1.2 A solução do problema 2

Existem diversas formas de se encontrar a sequência ótima de estados de um sistema. No entanto, como demonstrado por Blunsom [12] e Rabiner [5], o algoritmo mais recomendado é o algoritmo de *Viterbi*. Sendo o critério mais utilizado, este método leva em conta casos de onde haja probabilidades de transição iguais à zero, além disso, busca pela melhor sequência (caminho) de estados.

Sendo Q a sequência de estados e O a sequência observável, a maior probabilidade em um único caminho é dada por $\delta_t(i)$, no tempo t, contabilizando as primeiras t observações e acabando no estado s_i.

$$\delta_t(i) = \max_{q_1,q_2 \dots q_{t-1}} P(q_1 q_2 \dots q_t = s_i,\ O_1\ O_2 \dots\ O_t |\ \lambda) \tag{14}$$

Por indução têm-se

$$\delta_{t+1}(i) = [\max_i \delta_t(i) a_{ij}] b_j(O_{t+1}) \tag{15}$$

Sendo ψ_t um vetor de armazenamento de estados maximizados, guardando nas posições t o índice j do estado S_j que maximiza a sequência para o próximo estado.

O algoritmo de *Viterbi* é dividido em:

1. Inicialização:

$$\delta_1(i) = \pi_i b_i(O_1) \qquad 1 \le i \le N \tag{16}$$

$$\psi_t(j) = 0 \tag{17}$$

2. Indução:

$$\delta_t(j) = \max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij}\right] b_j(O_t) \qquad 2 \le t \le T \quad 1 \le j \le N \tag{18}$$

$$\psi_t(j) = \operatorname*{argmax}_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij}\right], \qquad 2 \le t \le T \quad 1 \le j \le N \tag{19}$$

3. Finalização:

$$P^* = \max_{1 \le k \le N} [\delta_T(k)] \tag{20}$$

$$q_T^* = \underset{1 \le k \le N}{\operatorname{argmax}}[\delta_T(k)] \tag{21}$$

4. Recriação do caminho (sequência de estado):

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 1 \tag{22}$$

Este algoritmo tem certa semelhança com o *forward*, sendo sua maior diferença no passo de indução, onde há o uso dos valores maximizados.

2.4.1.3 A solução do problema 3

Para determinar um método que ajuste o modelo de parâmetros $\lambda = (A, B, \Pi)$ de modo a maximizar a probabilidade da sequência de observação P(O|λ), dado um modelo, recomenda-se o algoritmo *Bawn-Welch*, o qual faz atualizações e melhoria de forma iterativa nos parâmetros do HMM.

Considere a seguinte definição:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \tag{23}$$

Dado a sequência O de treinamento, a variável $\xi_t(i,j)$ é a probabilidade de estar no estado S_i em t e no estado S_j em t+1. Com os algoritmos *forward* e *backward* já demonstrados, pode-se escrever a equação (23) na forma

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \tag{24}$$

Definindo $\gamma_t(i)$ como

$$\gamma_t(i) = P(q_t = S_i | O, \lambda) \tag{25}$$

Ou seja, a probabilidade de estar no estado S_i no tempo t, dado um modelo e a sequência de observação. Pode-se expressar tal equação utilizando os algoritmos *forward* e *backward* da seguinte forma

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \quad (25)$$

A qual pode ser relacionada com $\xi_t(i,j)$ da seguinte maneira

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \quad (26)$$

Somando $\gamma_t(i)$ sobre o tempo T, obtêm-se a estimativa do número de vezes que é atingido o estado S_i. O número de transições sobre S_i, deve-se levar o somatório até o tempo T-1. Assim, pode-se definir formalmente

$$\sum_{t=1}^{T-1} \gamma_t(i) = número\ esperado\ de\ transições\ de\ S_i \quad (27)$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = número\ esperado\ de\ transições\ de\ S_i\ para\ S_j \quad (28)$$

Através das fórmulas acima, pode-se reestimar os parâmetros de um MOM para as seguintes variáveis:

$$\bar{\pi} = Frequência\ esperada\ no\ estado\ S_i\ no\ tempo\ t = 1 = \gamma_t(i) \quad (29)$$

$$\overline{a_{ij}} = \frac{Número\ esperado\ de\ transições\ de\ S_i\ para\ S_j}{Número\ esperado\ de\ transições\ de\ S_i} \quad (30)$$

$$\overline{b_j}(k) = \frac{Número\ esperado\ de\ transições\ no\ estado\ j\ e\ observação\ v_k}{Número\ esperado\ no\ estado\ j} \quad (31)$$

Assim, definindo o modelo atual como $\lambda = (A, B, \Pi)$ e utilizando as equações (29)-(31) para criar um novo modelo $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\Pi})$, Baum e Baker provam que:

1. O modelo inicial é um ponto crítico, onde a função de probabilidade é um ponto de máximo, ou seja $\lambda = \bar{\lambda}$;
2. O novo modelo $\bar{\lambda}$ é mais provável que o antigo, sendo esse o novo modelo uma sequência mais provável de observação.

O processo é executado até o encontro do ponto de limite, onde $\lambda = \bar{\lambda}$.

3 MATERIAIS E MÉTODOS

Nesta sessão são apresentados os materiais necessários para elaboração dos experimentos, assim como a definição, tratamento e interpretação dos sinais adquiridos e sua aplicação no sistema de aprendizado.

3.1 MONITORAMENTO DA REDE CAN

Para análise das redes CAN disponíveis no veículo, utilizou-se a *CANcaseXL* da *Vector Informatik.* Este *hardware* é uma interface USB com duas entradas DB9 (dois canais), as quais possuem canais CAN independentes. Assim, para leitura do barramento, deve-se ramificar o par trançado da CAN do veículo e colocar um conector DB9 para integração com a interface. Com ela, é possível ler e transmitir mensagens CAN 11bits e 29bits, além de leitura e geração de códigos de falha.

O programa utilizado para a interpretação dos dados lidos é o CANalyzer 9.0 também da *Vector Informatik.* Este programa exibe as mensagens CAN do veículo de maneira que seja possível analisar o que se passa no barramento, como os identificadores e seus bits sendo alterados de acordo com a informação presente no veículo. A Figura 4 demonstra a leitura de um barramento, onde a coluna “Data” exibe na cor preta os hexadecimais alterados recentemente.

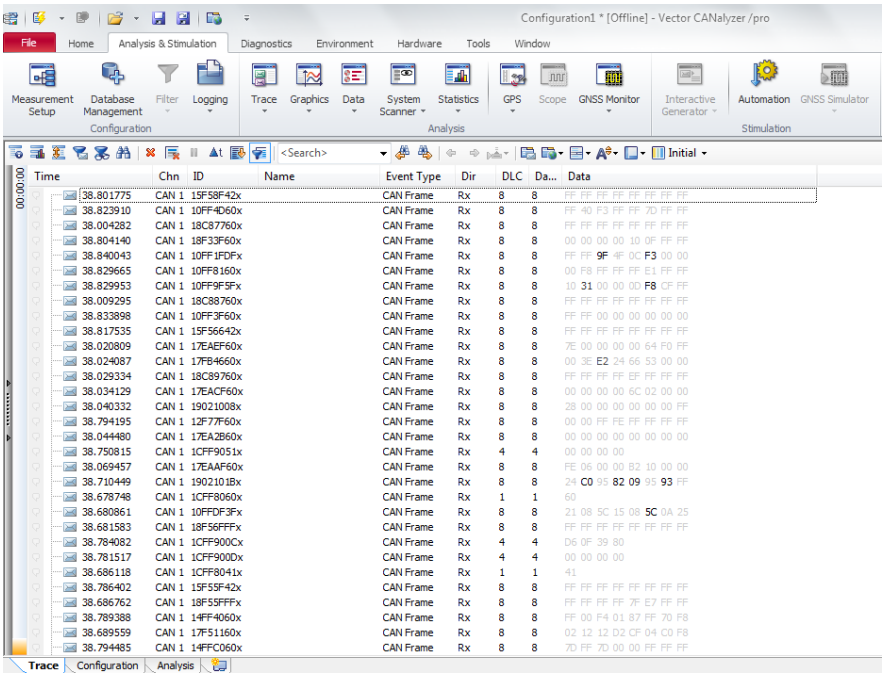


Figura 4 - Exemplo de funcionamento do programa CANalyzer.

Fonte: O autor.

Para possível compreensão do barramento lido, é necessária a aplicação de uma data-base proprietária do veículo disponibilizada pela montadora, que contém a informação dos identificadores como nome da mensagem, os sinais presentes neles e seus valores correspondentes.

Neste estudo, consideram-se três possíveis estados comportamentais do motorista: direção, repouso e aguardo/abastecimento.

Para análise do comportamento pela rede CAN, escolheu-se o barramento que possui mensagens tanto de interface com o motorista quanto informações do motor e sensores. Após análise dos sinais disponíveis no barramento por meio da data-base, foram selecionados aqueles que se supõem ser de grande impacto sobre o comportamento do motorista, sendo eles:

- Porcentagem da pressão sobre o pedal do acelerador;
- Porcentagem da pressão sobre o pedal do freio;
- Estado da tranca da porta;
- Mudança no nível de combustível;
- Volume do rádio;
- Posição da chave de ignição;

- Velocidade acima de zero.

Além da definição dos sinais, deve-se analisar sua variação em função do tempo. Para tanto, utilizou-se novamente das informações presentes na data-base, a qual informa o *startbit* de cada sinal sobre a mensagem e seu tamanho. Deste modo, ao executar uma gravação da rede CAN, pode-se identificar na mensagem a variação dos sinais presentes no arquivo de registro em função do tempo e seu conteúdo.

Ao registrar um arquivo de *log* como exemplo para análise, identificou-se a possibilidade de exportar o arquivo como uma tabela no formato ASCII, o qual torna fácil a compreensão do texto presente no documento gerado. A Tabela 3 contém um exemplo de como é formada a tabela no arquivo de *log*.

Tempo	Canal	ID Hex	Direção	DLC	Data field								Comprimento	Contagem de bits	ID Dec
					0	1	2	3	4	5	6	7			
0.000000	1	FFFFFFFFx	Rx	d 8	FF	FF	FF	FF	FF	FF	FF	FF	Length = 273910	BitCount = 141	ID = 4294967295x

Tabela 3 - Estrutura da mensagem CAN exportada pelo CANalyzer em ASCII.

Tal tabela contém informações a respeito do momento de leitura, como o tempo, o identificador e o *data field*, o qual é o conteúdo dos sinais. Com essas três informações, pode-se interpretar o momento em que o sinal foi variado no tempo.

3.2 AQUISIÇÃO DE DADOS

Após a instrumentação do veículo com o *CANalyzer*, iniciou-se a aquisição de dados. Para tanto, solicitou-se que um motorista qualificado executasse a direção do veículo.

Foram executados os seguintes registros para treinamento:

- Repouso: 447 segundos;
- Direção do ponto de início ao posto de abastecimento: 385 segundos;
- Abastecimento: 815 segundos;
- Direção do posto de abastecimento ao posto de carga: 411 segundos;
- Direção do posto de carga ao ponto de início: 408 segundos;
- Repouso: 665 segundos.

Em outro momento, executou-se mais uma quantidade de medições:

- Do ponto de início ao posto de abastecimento: 614 segundos;
- Abastecimento: 776 segundos;
- Do ponto de abastecimento ao ponto de início: 1386 segundos;
- Repouso: 489 segundos.

Totalizando 6596 segundos de dados de treinamento.

De forma a validar o sistema, registrou-se uma viagem contendo todos os estados de estudo, repouso, direção e abastecimento totalizando 7553 segundos de validação.

3.3 TRATAMENTO DE DADOS

Para que a informação presente no veículo seja interpretada com melhor precisão pelo software de análise estatística, é necessário que o registro do barramento *CAN* seja tratado de acordo com a particularidade de cada sinal. Assim, com a aquisição de um registro exemplo, foram elaborados *scripts* para tratamento dos sinais:

1. Filtragem por mensagem:

   Após a exportação do documento puro vindo do *CANalyzer*, obtém-se uma tabela onde os sinais são listados em função do tempo, como já demonstrado. Aplicou-se um filtro sobre as mensagens, de modo a gerar um arquivo com apenas a mensagem pertinente ao estudo.

2. Seleção de sinal:

   Utilizando o arquivo contendo a mensagem filtrada e, com a informação da *data field* contida na data-base da *CAN*, foi selecionada a coluna correspondente ao sinal a ser analisado. Aplicou-se, portanto, outro filtro sobre o arquivo de modo a obter apenas a coluna referente ao sinal.

3. Média Móvel Exponencial:

Devido ao fato de as mensagens conterem distintas taxas de amostragem, é necessário aplicar a média móvel exponencial sobre as mensagens com maior taxa de amostragem, de modo a aumentar o tempo de resposta dos sinais.

Para definição do alfa, variável de definição do peso do sinal mais recente, utilizou-se novamente a data-base da rede de estudo para analisar qual o ciclo de cada mensagem. Assim, a partir de tal informação, definiu-se um alfa para cada valor de taxa de amostragem, sendo ele medido de maneira empírica até que a média exponencial mantenha a variação do sinal por um segundo.

4. Discretização e amostragem:

   Através da interpretação dos sinais avaliados, aplicou-se a discretização do sinal, onde, novamente, utilizou-se da data-base para interpretação da relação entre sinal e evento.

   Além disso, para os sinais com alta taxa de amostragem, foi aplicado um filtro de modo a adquirir amostras do sinal de acordo com sua variação no tempo de maneira inversamente proporcional a sua taxa de amostragem. Ou seja, quanto maior sua taxa, menor o número de mensagens adquiridas, de modo a igualar a taxa de amostragem de todos os sinais.

Todos os códigos aplicados estão presentes no anexo I. Após o tratamento dos sinais, os valores obtidos foram inseridos em uma tabela onde cada coluna corresponde aos sinais de estudo. Um *software* de manipulação de dados de forma a implementar a lógica por trás do MOM no sistema de estudo.

3.4 R PROJECT

Para aplicação do MOM, utilizou-se de um *software* voltado para manipulação de dados estatísticos e análise de gráficos. Chamado de *R,* este *software* é uma ferramenta de uso livre desenvolvida com base no *software S* desenvolvido por John Chambers da *Bell Laboratories*.

Para aplicação do modelo de estudo, executou-se a instalação de pacotes de análise de dados no *software* R, os quais aplicam médias móveis, análise gráfica e o próprio HMM. Os pacotes utilizados são:

- DepmixS4 - é um pacote desenvolvido por Ingmar Visser e Maarten Speekenbrink adaptado para aplicação do Modelo Oculto de Markov sobre dados categóricos mistos e contínuos (séries temporais), conhecido como “modelos de mistura dependente”.
- QCC - desenvolvido por Luca Scrucca, Greg Snow e Peter Bloomfield, este pacote faz aplicação da média móvel exponencial sobre os dados.
- Cluster – desenvolvido por Martin Maechler, Peter Rousseeuw, Anja Struyf, Mia Hubert, Kurt Hornik, Matthias Studer, Pierre Roudier, Juan Gonzalez e Kamil Kozlowski, este pacote executa a clusterização dos dados inseridos para análise de grupos.

Para aplicação do MOM foram seguidos os seguintes passos:

1. Leitura de tabela de sinais tratados para aprendizado;
2. Aplicação de média móvel exponencial sobre cada coluna;
3. Aplicação da função *depmix,* a qual gera um modelo especificando a distribuição observada. Ou seja, contendo os sinais, seu tipo (contínuo ou discreto) e o número de estados esperados;
4. Aplicação da função *fit*, gerando os dados de convergência de modo a otimizar os parâmetros;
5. Função *posterior* a qual aplica o algoritmo de *Viterbi.*

Assim, já se obtém o modelo contendo suas probabilidades de transição e emissão. Para que o modelo seja validado, foram executados os seguintes passos:

1. Leitura de tabela de sinais tratados para validação;
2. Aplicação de média móvel exponencial sobre cada coluna;
3. Aplicação da função *depmix*;
4. Função *setpars*, de modo a aplicar os parâmetros do modelo de aprendizado;

5. Função *viterbi,* a qual, utilizando os parâmetros e dados, fará uma estimativa da probabilidade de transição de estados para cada linha da tabela.

Os dados obtidos foram analisados e comparados com o estado efetivo do motorista.

Para análise, foram estudados sistemas de taxas de amostragem diferentes, de forma a analisar a alteração do valor da probabilidade de transição de estados.

Além disso, analisou-se a correlação entre os sinais de estudo, de forma a verificar o impacto da remoção de sinais que possuem alta correlação entre si. A Tabela 4 contém os valores obtidos.

	Acelerador	Combustive	Freio	Porta	Volume	Ignicao	Movimento
Acelerador	-						
Combustivel	-0,1208601	-					
Freio	-0,1263976	-0,0451053	-				
Porta	-0,161913	-0,0722326	-0,1287206	-			
Volume	0,11603749	0,0054679	0,10315182	-0,4752261	-		
Ignicao	0,32995345	0,09497573	0,12568397	-0,5613783	0,13234538	-	
Movimento	0,80265128	-0,1273716	0,12938613	-0,1607456	0,10999687	0,3414233	-

Tabela 4 - Correlação entre sinais de estudo.

Fonte: O autor.

Assim, outra situação de estudo foi executada onde os sinais de “Movimento” e “Porta” foram removidos, devido à alta correlação com outros sinais.

De modo a analisar possíveis padrões nos dados inseridos e a presença de grupos distintos, aplicou-se a função *cluster*. Em seguida, aplicou-se o treinamento e sua validação. Nos resultados, têm-se os dados das matrizes de probabilidade de transição, matrizes de probabilidade de emissão. Em seguida, têm-se um gráfico comparativo entre estado efetivo e estado estimado pelo *software*. O estado não foi informado ao sistema de treinamento, apenas a quantidade esperada de estados. Assim, os números entre ambos os gráficos não devem ser levados em consideração na análise, apenas a transição entre estados. Após, analisa-se a precisão do sistema através da comparação do estado efetivo do motorista no arquivo de validação e a estimativa do sistema através do mesmo arquivo.

4 RESULTADOS

Os resultados apresentados são divididos de acordo com sua taxa de amostragem e quantidade de sinais presentes.

4.1 PRIMEIRA ANÁLISE

Taxa de amostragem: 1 amostra por segundo.

Sinais: Acelerador, combustível, freio, porta, volume, chave de ignição, e movimento.

Resultados:

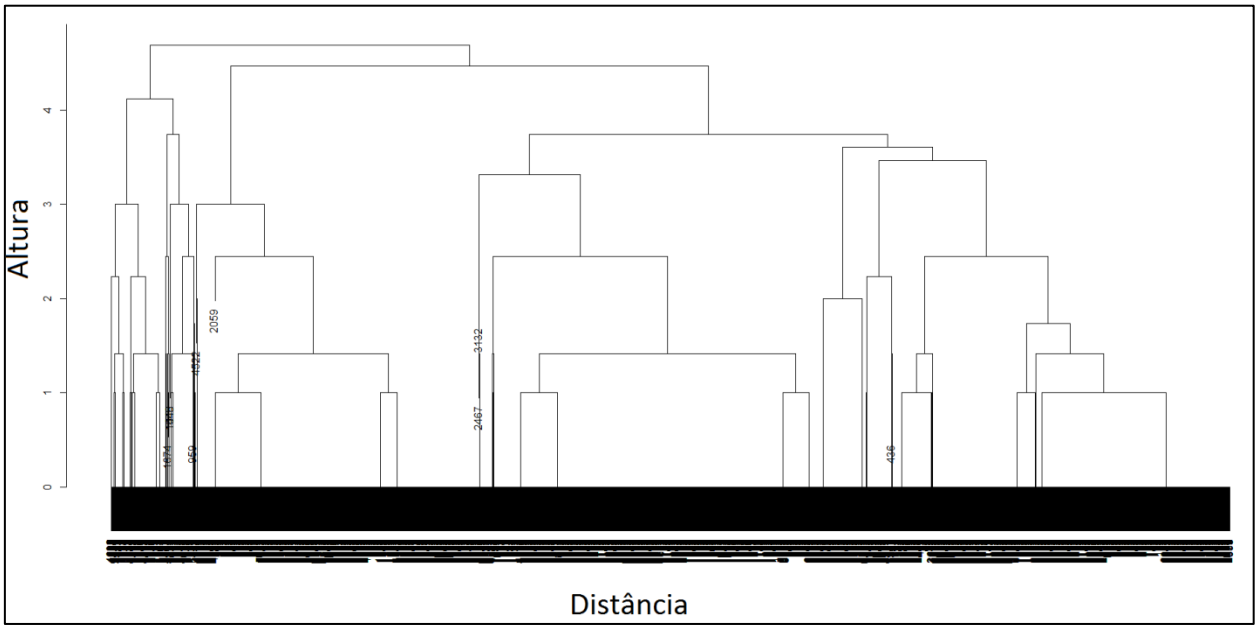


Gráfico 1 - Dendrograma da primeira análise.

Fonte: O autor.

	Rep.	Abas.	Dir.
Rep.	0.996	0.001	0.003
Abas.	0.004	0.996	0.000
Dir.	0.002	0.000	0.998

Tabela 5 - Matriz de probabilidade de transição de estado da primeira análise.

Fonte: O autor.

	Aceleração	Combustível	Freio	Porta	Volume	Ignição	Movimento
Repouso	0.078	0.201	0.020	0.703	0.599	0.396	0.087
Abastecimento	0.005	0.361	0.001	0.436	0.944	0.006	0.006
Dirigindo	0.524	0.015	0.175	0.425	0.589	0.192	0.581

Tabela 6 - Matriz de probabilidade de emissão por estado da primeira análise.

Fonte: O autor.

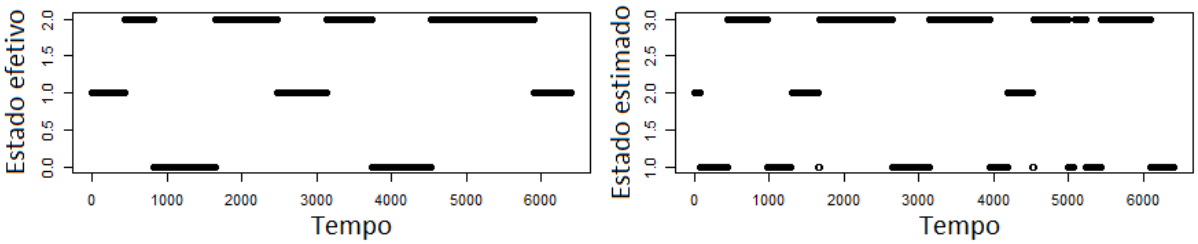


Gráfico 2 - Comparativo entre estado efetivo e estimado da primeira análise.

Fonte: O autor.

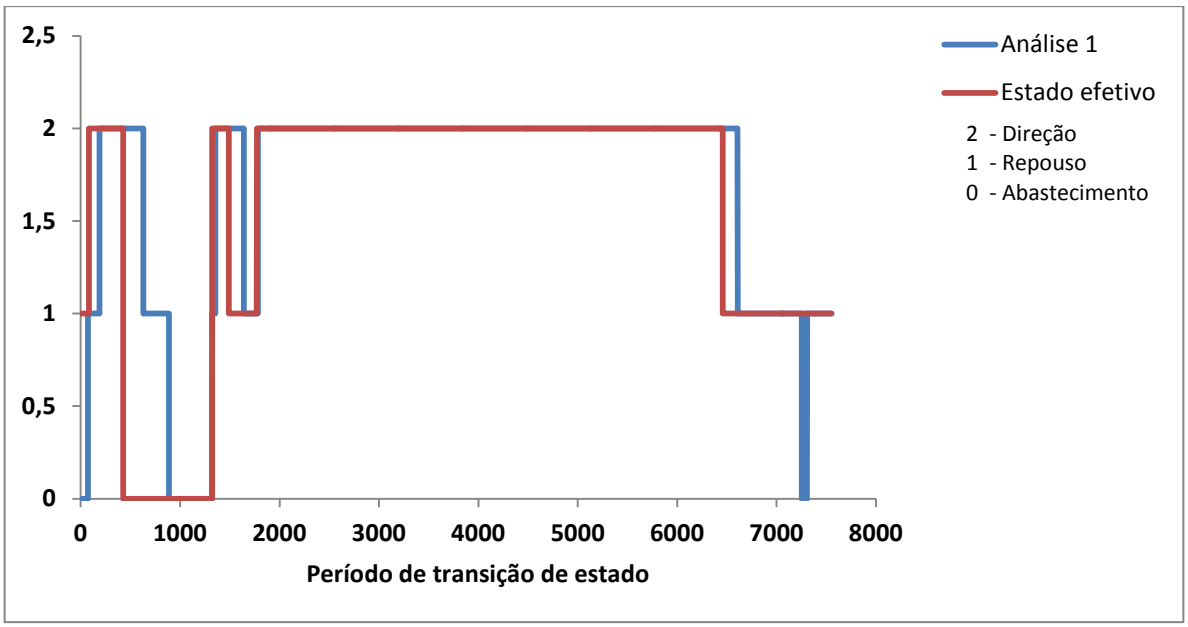


Gráfico 3 - Comparativo entre estado efetivo e estado definido pela primeira análise.

Fonte: O autor.

Tempo médio de transição: 103 períodos.

Porcentagem de acerto: 86,3%.

4.2 SEGUNDA ANÁLISE

Taxa de amostragem: 1 amostra a cada 6 segundos.

Sinais: Acelerador, combustível, freio, porta, volume, chave de ignição, e movimento.

Resultados:

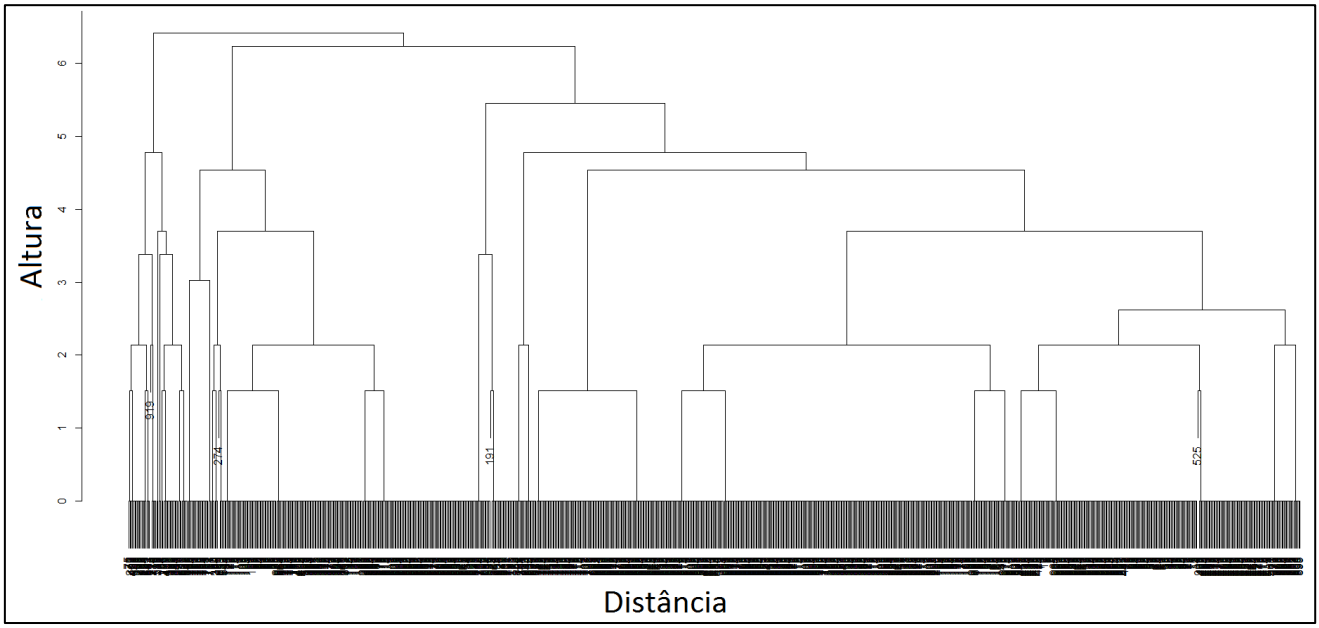


Gráfico 4 - Dendrograma da segunda análise.

Fonte: O autor.

	Rep.	Abas.	Dir.
Rep.	0.992	0.000	0.008
Abas.	0.003	0.992	0.006
Dir.	0.003	0.008	0.989

Tabela 7 - Matriz de probabilidade de transição de estado da segunda análise.

Fonte: O autor.

	Aceleração	Combustível	Freio	Porta	Volume	Ignição	Movimento
Repouso	0.162	0.012	0.042	0.219	0.176	0.132	0.178
Abastecimento	0.085	0.072	0.060	0.223	0.585	0.077	0.097
Dirigindo	0.193	0.030	0.057	0.193	0.422	0.107	0.189

Tabela 8 - Matriz de probabilidade de emissão por estado da segunda análise.

Fonte: O autor.

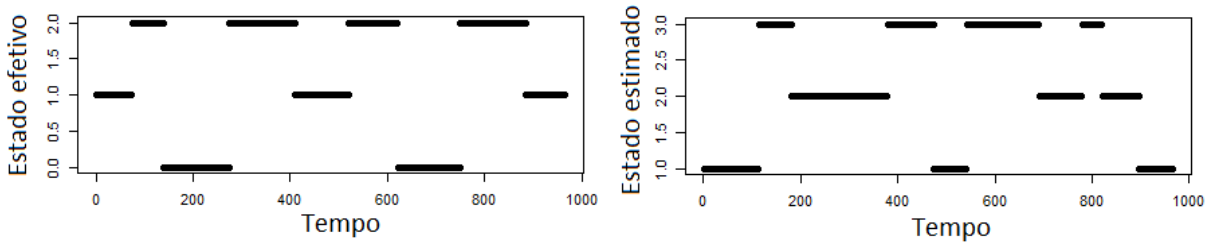


Gráfico 5 - Comparativo entre estado efetivo e estimado da segunda análise.

Fonte: O autor.

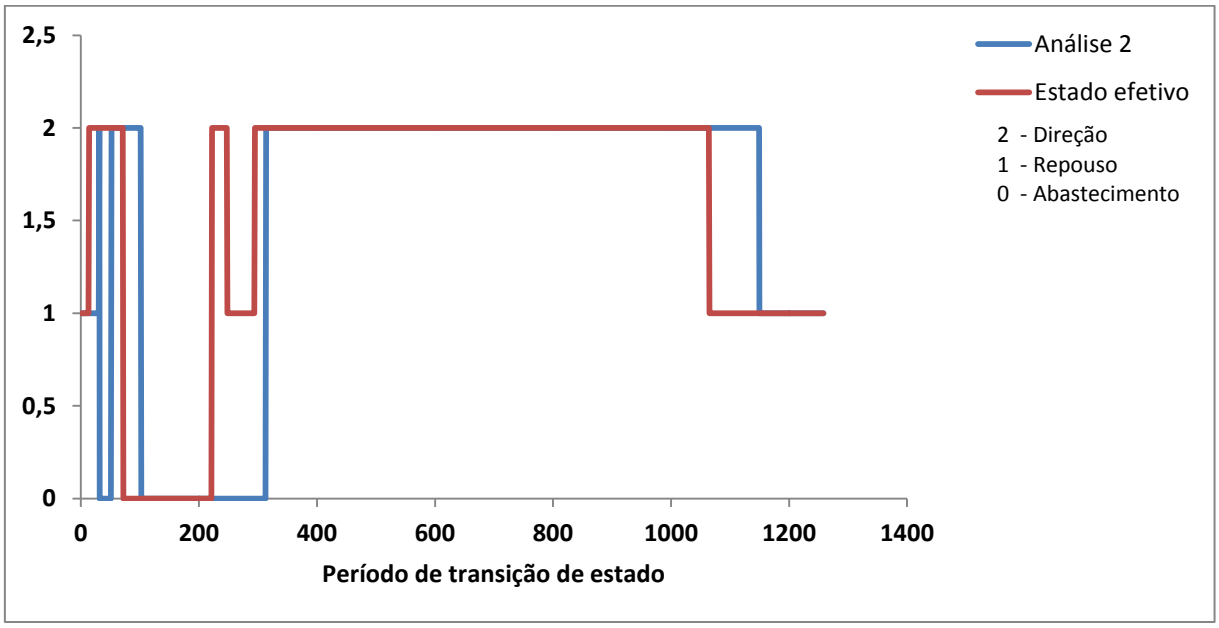


Gráfico 6 - Comparativo entre estado efetivo e estado definido pela segunda análise.

Fonte: O autor.

Tempo médio de transição de estado: 291 períodos.

Porcentagem de acerto: 80,1%.

4.3 TERCEIRA ANÁLISE

Taxa de amostragem: 1 amostra por segundo.

Sinais: Acelerador, combustível, freio, volume e chave de ignição.

Resultados:

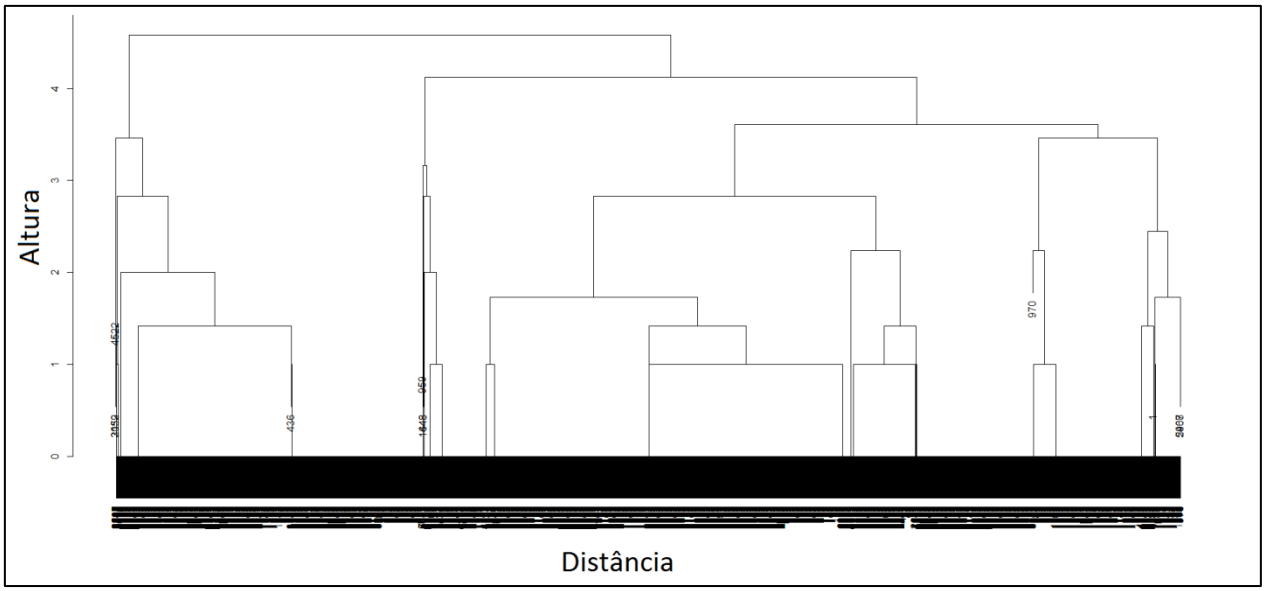


Gráfico 7 - Dendrograma da terceira análise.

Fonte: O autor.

	Rep.	Abas.	Dir.
Rep.	0.999	0.001	0.000
Abas.	0.002	0.997	0.002
Dir.	0.000	0.002	0.998

Tabela 9 - Matriz de probabilidade de transição de estado da terceira análise.

Fonte: O autor.

	Aceleração	Combustível	Freio	Volume	Ignição
Dirigindo	0.517	0.008	0.212	0.706	0.000
Abastecimento	0.050	0.320	0.009	0.701	0.397
Repouso	0.597	0.025	0.094	0.414	0.297

Tabela 10 - Matriz de probabilidade de emissão por estado da terceira análise.

Fonte: O autor.

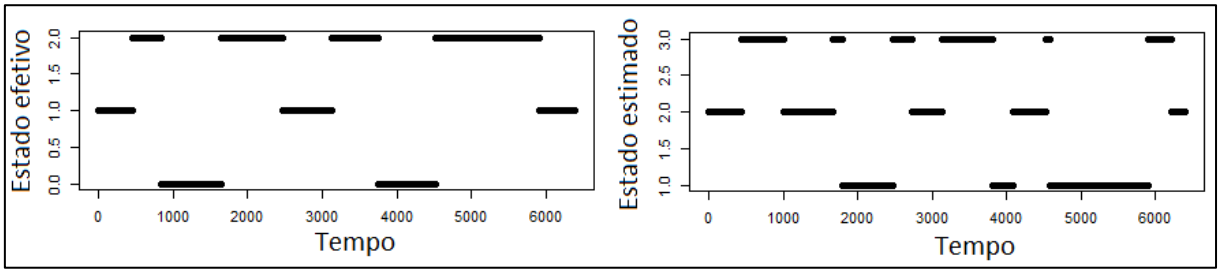


Gráfico 8 - Comparativo entre estado efetivo e estimado da terceira análise.

Fonte: O autor.

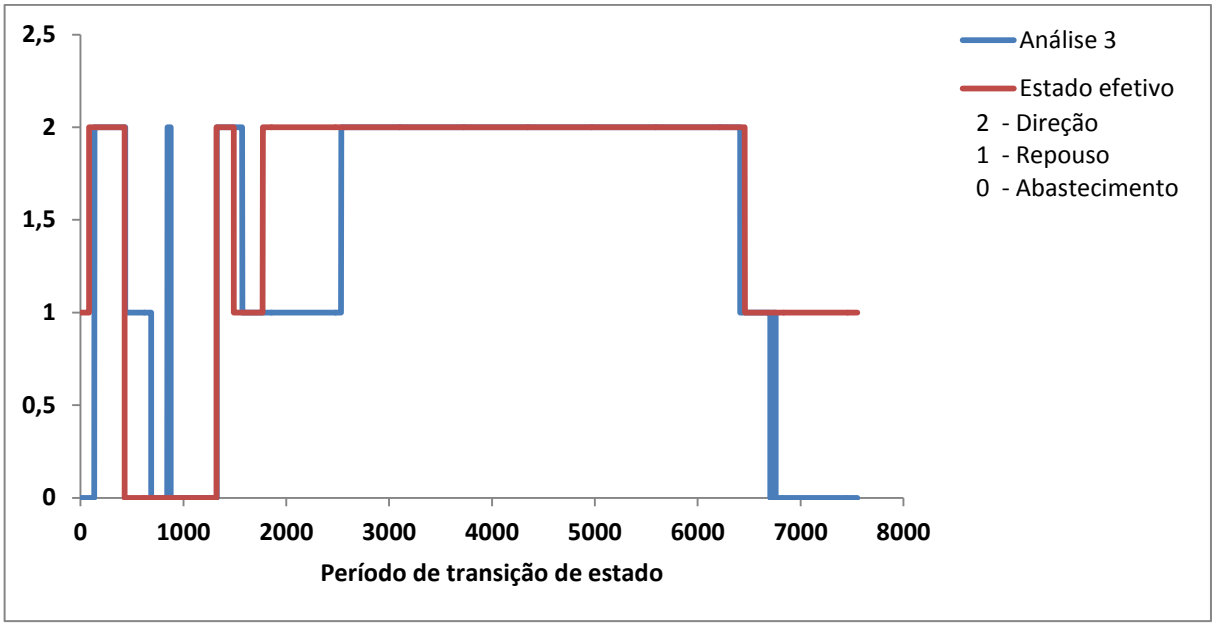


Gráfico 9 - Comparativo entre estado efetivo e estado definido pela terceira análise.

Fonte: O autor.

Tempo médio de transição: 37 períodos.

Porcentagem de acerto: 71,8%.

4.4 QUARTA ANÁLISE

Taxa de amostragem: 1 amostra a cada 6 segundos.

Sinais: Acelerador, combustível, freio, volume e chave de ignição.

Resultados:

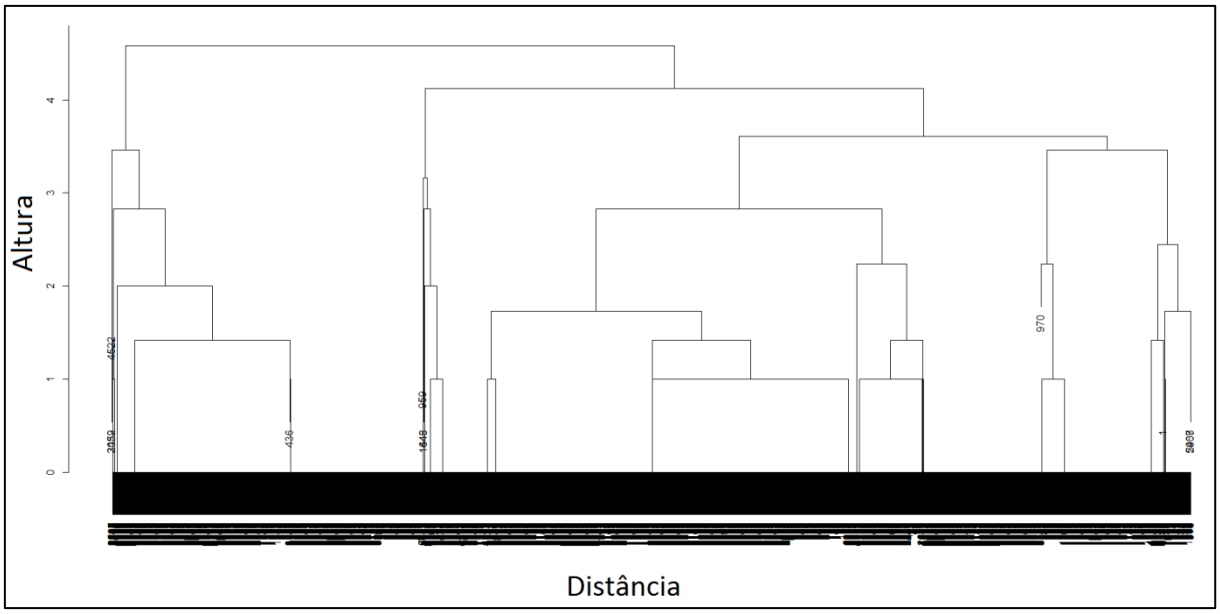


Gráfico 10 - Dendrograma da quarta análise.

Fonte: O autor.

	Dir.	Rep.	Abas.
Dir.	0.995	0.002	0.002
Rep.	0.005	0.990	0.005
Abas.	0.006	0.000	0.994

Tabela 11 - Matriz de probabilidade de transição de estado da quarta análise.

Fonte: O autor.

	Aceleração	Combustível	Freio	Volume	Ignição
Repouso	0.117	0.046	0.061	0.647	0.026
Abastecimento	0.201	0.112	0.017	0.322	0.110
Dirigindo	0.306	0.014	0.047	0.231	0.121

Tabela 12 - Matriz de probabilidade de emissão por estado da quarta análise.

Fonte: O autor.

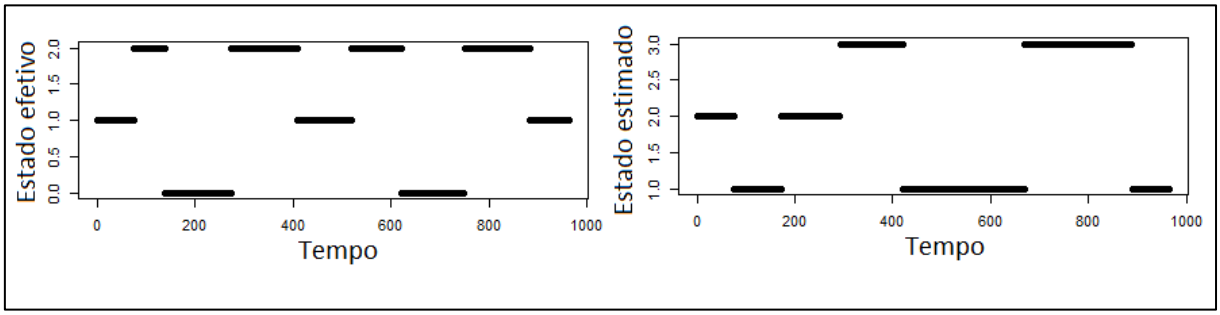


Gráfico 11 - Comparativo entre estado efetivo e estimado da quarta análise.

Fonte: O autor.

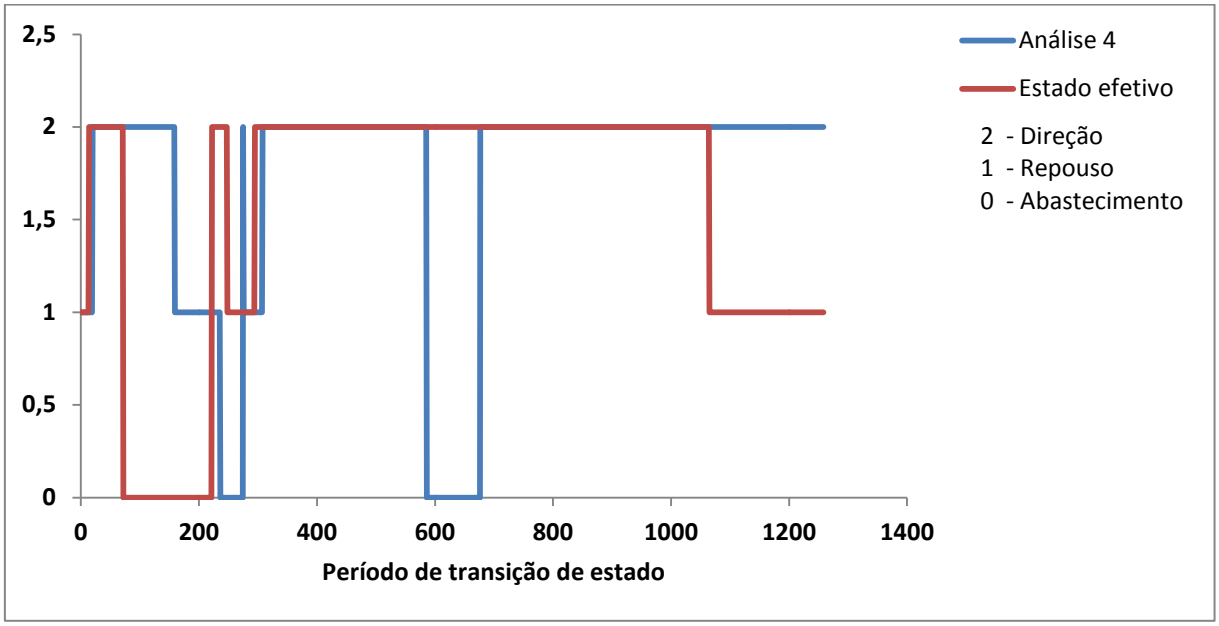


Gráfico 12 - Comparativo entre estado efetivo e estado definido pela quarta análise.

Fonte: O autor.

Tempo médio de transição: 33 períodos.

Porcentagem de acerto: 59,6%.

Os dados obtidos nas tabelas de probabilidade de transição de estado foram truncados pelo *software*. Devido a este fato, os campos de valor igual à zero, são valores com probabilidade próximos de zero.

Através da análise dos resultados, identifica-se que a primeira análise, contendo maior taxa de amostragem, demonstrou maior acerto. No entanto, identifica-se também que há um maior tempo entre troca de estados, o qual ocorre devido à alta taxa de amostras.

CONSIDERAÇÕES FINAIS

O comportamento de um motorista de caminhão em seu trabalho tem maior dinâmica do que apenas os estados que foram definidos no estudo. No entanto, identificou-se que, com poucos sinais, é possível obter uma alta taxa de acerto sobre o seu comportamento.

Analisando os sinais, os estados do motorista em repouso e o reabastecimento do veículo têm pouca diferença entre si, caso o sinal de combustível não esteja presente ou demore a alterar. Devido a este fato, verifica-se que o sistema definiu o estado erroneamente em três das quatro análises antes da informação de combustível ser alterada.

4.5 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

De modo a aprofundar o estudo, pode-se utilizar uma maior quantidade de amostras tanto para treinamento quanto validação do sistema.

O estudo efetuado contempla apenas três dos mais diversos estados comportamentais de um motorista com seu caminhão. Portanto, outros estados como manutenção, carga e descarga, podem ser analisados.

Os mais diversos sinais e redes CAN podem ser adicionados ao estudo de modo a aprimorar o resultado, tanto na velocidade de identificação de troca de estado quanto na precisão do sistema.

Podem ser aplicados diversos métodos de aprendizado de máquina para a definição do seu estado comportamental, de modo a definir qual é o mais indicado para esta situação.

REFERÊNCIAS

[1] Lei do motorista, Nº 13.103, DE 2 DE MARÇO DE 2015. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13103.htm

[2] J. S. Espindola. Um Estudo sobre Modelos Ocultos de Markov HMM - Hidden Markov Model. 2009

[3] L. R. Rabiner e B. H. Juang. An introduction to hidden markov models. IEEE ASSP Magazine, 3(1):4–16, 1986.

[4] H. T. Junior. Estudo dos protocolos de comunicação das arquiteturas eletroeletrônicas automotivas, com foco nas suas características e respectivas aplicações, visando o direcionamento para o uso adequado e customizado em cada categoria de veículo. 2010

[5] L. R. Rabiner. A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. 1989

[6] A. A. Guimarães. O Protocolo CAN: Entendendo e Implementando uma Rede de Comunicação Serial de Dados baseada no Barramento “Controller Area Network”. 2002

[6] Vector. Introduction to SAEJ1939. Disponível em: https://vector.com/portal/medien/cmc/application_notes/AN-ION-1-3100_Introduction_to_J1939.pdf. Acessado em 14/08/18.

[7] http://www.portalaction.com.br/processo-estocastico/cadeia-de-markov

[8] https://pt.wikipedia.org/wiki/Modelo_oculto_de_Markov

[9] Lussier E. F. Markov Models and Hidden Markov Models: A Brief Tutorial. 1998

[10] CSS Electronics, CAN Bus explained – A simple intro - https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en

[11] C. M. Grinstead; J. L. Snell. Introduction to Probability.

[12] Blunsom P. Hidden Markov Models. 2004.

[13] http://www.ti.com/lit/an/sloa101b/sloa101b.pdf

[14] Visser, I. e Speekenbrink, M. depmixS4: An R Package for Hidden Markov Models. Agosto 2010

[15] stockcharts.com/school/doku.php?id=chart_school:technical_indicators:moving_averages

[16] Vakati K. Driver Telematics Analysis. 2015

[17] C. Hwang, M. Chen, C. Shih, H. Chen and W. K. Liu, "Apply Scikit-Learn in Python to Analyze Driver Behavior Based on OBD Data," 2018

5 ANEXO

Código em shell para aplicação dos códigos de tratamento:

```
#!/bin/sh

grep 10FF1FDFx c.asc > sinais/Acelerador.asc
grep 10FF1FDFx c.asc > sinais/Freio.asc
grep 17F51166x c.asc > sinais/Porta.asc
grep 17FF5F9Fx c.asc > sinais/Combustivel.asc
grep 18F06A82x c.asc > sinais/Radio_volume.asc
grep 10FF1FDFx c.asc > sinais/Vehicle_mode.asc
grep 10FF9F5Fx c.asc > sinais/Velocidade.asc

./Filtro_acelerador > filtro/Acelerador_filtrado.asc
./Filtro_freio > filtro/Freio_filtrado.asc
./Filtro_porta > filtro/Porta_filtrado.asc
./Filtro_combustivel > filtro/Combustivel_filtrado.asc
./Filtro_radio_volume > filtro/Radio_volume_filtrado.asc
./Filtro_vehicle_mode > filtro/Vehicle_mode_filtrado.asc
./Filtro_velocidade > filtro/Velocidade_filtrado.asc

./Media_movel_exp_acelerador > mme/Med_exp_acelerador.asc
./Media_movel_exp_freio > mme/Med_exp_freio.asc
./Media_movel_exp_porta > mme/Med_exp_porta.asc
./Media_movel_exp_combustivel > mme/Med_exp_combustivel.asc
./Media_movel_exp_radio_volume > mme/Med_exp_radio_volume.asc
./Media_movel_exp_vehicle_mode > mme/Med_exp_vehicle_mode.asc
./Media_movel_exp_velocidade > mme/Med_exp_velocidade.asc

./Dis_acelerador > dis/dis_acelerador.asc
./Dis_freio > dis/dis_freio.asc
./Dis_porta > dis/dis_porta.asc
./Dis_combustivel > dis/dis_combustivel.asc
./Dis_radio_volume > dis/dis_radio_volume.asc
./Dis_vehicle_mode > dis/dis_vehicle_mode.asc
./Dis_velocidade > dis/dis_velocidade.asc
```

Códigos em C para tratamento dos sinais:

Filtro Acelerador:

```
#include <stdio.h>
```

```
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
   int d[8];
   char outros[9][100];
   int interessa;

   if ((a=fopen("./sinais/Acelerador.asc","r")) == NULL) {
       printf("\nErro lendo Leitura1.asc");
       exit (0);
   }

   while (!feof(a))
   {
       fgets(str, sizeof(str), a);

       //printf("%s",str);

       sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);

      printf("%d\n", interessa);

      interessa=0;

      interessa=d[7];

   }

}
```

Média Móvel Exponencial Acelerador:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
   float i;
   int j=0;
   float mme[1000000];
   float buff=0;

   if ((a=fopen("./filtro/Acelerador_filtrado.asc","r")) == NULL) {
       printf("\nErro lendo Leitura1.asc");
       exit (0);
   }
   while (!feof(a)){
```

```
        fgets(str, sizeof(str), a);
            sscanf(str, "%e ", &i);
        mme[j] = (i-buff)*0.18 + buff;
   buff=mme[j];
   printf("%f\n", mme[j]);
  j=j+1;
   }

}
```

Discretização Acelerador:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
   float i;
   int j=0;
   int b=0;
   int contador=0;
   int dis[1000000];

   if ((a=fopen("./mme/Med_exp_acelerador.asc","r")) == NULL) {
       printf("\nErro lendo Leitura1.asc");
       exit (0);
   }
   while (!feof(a)){
        fgets(str, sizeof(str), a);
            sscanf(str, "%f ", &i);
            if(i== 229.5){
            dis[j]= 0;
            }
            if(i== 0){
        dis[j]= 0;
        }
            if(i>0 && i<229.5f){
        dis[j]= 2;
        }
   if(contador == 100){
   printf("%d\n", dis[j]);
   contador = 0;
}
  contador = contador + 1;
  j=j+1;
   }
}
```

Filtro Combustível:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[10000];
```

```
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
   int d[8];
   char outros[9][100];
   int interessa;

   if ((a=fopen("./sinais/Combustivel.asc","r")) == NULL) {
       printf("\nErro lendo Leitura1.asc");
       exit (0);
   }

   while (!feof(a))
   {
       fgets(str, sizeof(str), a);

       //printf("%s",str);

       sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);

      printf("%d\n", interessa);

      interessa=0;

      interessa=d[0];

   }

}
```

Média Móvel Exponencial Combustível:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
   int i;
   int j=0;
   float mme[1000000];

   if ((a=fopen("./filtro/Combustivel_filtrado.asc","r")) == NULL) {
       printf("\nErro lendo Leitura1.asc");
       exit (0);
   }

   while (!feof(a))
   {
       fgets(str, sizeof(str), a);

            sscanf(str, "%i ", &i); //converte para um valor numérico
        mme[j] = i;
   printf("%f\n", mme[j]);
```

```
  j=j+1;

}
}
```

Discretização Combustível:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
   float i;
   int j=0;
   float buff=0;
   int contador=0;
   int dis[1000000];

   if ((a=fopen("./mme/Med_exp_combustivel.asc","r")) == NULL) {
       printf("\nErro lendo Leitura1.asc");
       exit (0);
   }
   while (!feof(a)){
        fgets(str, sizeof(str), a);
            sscanf(str, "%f ", &i);
            if(i == buff){
            dis[j]= 0;
            }
            if(i != buff){
        dis[j]= 2;
            }
   if(contador == 1){
   printf("%d\n", dis[j]);
   contador = 0;
   }
  j=j+1;
  contador=contador+1;
  buff=i;
   }
}
```

Filtro Freio:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[10000];
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
   int d[8];
   char outros[9][100];
```

```
  int interessa;

  if ((a=fopen("./sinais/Freio.asc","r")) == NULL) {
     printf("\nErro lendo Leitura1.asc");
     exit (0);
  }

  while (!feof(a))
  {
     fgets(str, sizeof(str), a);

     //printf("%s",str);

     sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);

     printf("%d\n", interessa);

     interessa=0;

     interessa=d[6];
     }

}
```

Média Móvel Exponencial Freio:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  float buff=0;
  int j=0;
  float mme[1000000];

  if ((a=fopen("./filtro/Freio_filtrado.asc","r")) == NULL) {
     printf("\nErro lendo Leitura1.asc");
     exit (0);
  }

  while (!feof(a))
  {
     fgets(str, sizeof(str), a);
          sscanf(str, "%e ", &i); //converte para um valor numérico
      mme[j] = (i-buff)*0.18 + buff;
  buff=mme[j];
  printf("%f\n", mme[j]);
 j=j+1;
  }
}
```

Discretização Freio:

```
#include <stdio.h>
```

```
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
   float i;
   int j=0;
   int contador=0;
   int dis[1000000];

   if ((a=fopen("./mme/Med_exp_freio.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }
   while (!feof(a)){
        fgets(str, sizeof(str), a);
            sscanf(str, "%f ", &i);
            if(i== 229.5){
            dis[j]= 0;
            }
            if(i== 0){
        dis[j]= 0;
        }
            if(i>0 && i<229.5f){
        dis[j]= 2;
        }
   if(contador == 100){
   printf("%d\n", dis[j]);
   contador = 0;
}
  contador = contador + 1;
  j=j+1;
   }
}
```

Filtro Porta:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[10000];
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
   int d[8];
   char outros[9][100];
   int interessa;

   if ((a=fopen("./sinais/Porta.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }
```

```
  while (!feof(a))
  {
     fgets(str, sizeof(str), a);

     //printf("%s",str);

     sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);

     printf("%d\n", interessa);

     interessa=0;

     interessa=d[5];

  }
}
```

Média Móvel Exponencial Porta:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  float buff = 0;
  int j=0;
  float mme[1000000];

  if ((a=fopen("./filtro/Porta_filtrado.asc","r")) == NULL) {
     printf("\nErro lendo Leitura1.asc");
     exit (0);
  }

  while (!feof(a))
  {
     fgets(str, sizeof(str), a);
          sscanf(str, "%e ", &i); //converte para um valor numérico
      mme[j] = (i-buff)*0.9 + buff;
  buff=mme[j];
  printf("%f\n", mme[j]);
 j=j+1;
  }
 }
```

Discretização Porta:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
```

```
   int j=0;
   int contador =0;
   int dis[1000000];

   if ((a=fopen("./mme/Med_exp_porta.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }
   while (!feof(a)){
       fgets(str, sizeof(str), a);
           sscanf(str, "%f ", &i);
           if(i < 140){ // abertas
           dis[j]= 0;
           }
           if(i>140 && i<145){ // uma aberta
       dis[j]= 1;
       }
           if(i > 145){ // trancadas
       dis[j]= 2;
       }
   if(contador == 2){
   printf("%d\n", dis[j]);
   contador = 0;
}
  contador = contador + 1;
  j=j+1;
   }
}
```

Filtro Volume:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[10000];
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
   int d[8];
   char outros[9][100];
   int interessa;

   if ((a=fopen("./sinais/Radio_volume.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }

   while (!feof(a))
   {
      fgets(str, sizeof(str), a);

      //printf("%s",str);
```

```
      sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);

     printf("%d\n", interessa);

     interessa=0;

     interessa=d[2];

  }
}
```

Média Móvel Exponencial Volume:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  float buff =0;
  int j=0;
  float mme[1000000];

  if ((a=fopen("./filtro/Radio_volume_filtrado.asc","r")) == NULL) {
     printf("\nErro lendo Leitura1.asc");
     exit (0);
  }

  while (!feof(a))
  {
     fgets(str, sizeof(str), a);

          sscanf(str, "%e ", &i); //converte para um valor numérico
          mme[j] = (i-buff)*0.7 + buff;
  buff=mme[j];
  printf("%f\n", mme[j]);
 j=j+1;
  }
}
```

Discretização Volume:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  int j=0;
  int contador=0;
  int dis[1000000];

  if ((a=fopen("./mme/Med_exp_radio_volume.asc","r")) == NULL) {
```

```
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }
   while (!feof(a)){
       fgets(str, sizeof(str), a);
           sscanf(str, "%f ", &i);
           if(i < 30){
           dis[j]= 0;
           }
           if(i > 30 && i < 160){
       dis[j]= 1;
       }
           if(i > 160){
       dis[j]= 2;
       }
   if(contador == 5){
   printf("%d\n", dis[j]);
   contador = 0;
}
  contador = contador + 1;
  j=j+1;
   }
}
```

Filtro Ignição:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[10000];
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
   int d[8];
   char outros[9][100];
   int interessa;

   if ((a=fopen("./sinais/Vehicle_mode.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }

   while (!feof(a))
   {
      fgets(str, sizeof(str), a);

      //printf("%s",str);

      sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);

     printf("%d\n", interessa);
```

```
    interessa=0;

    interessa=d[3];

    }

}
```

Média Móvel Exponencial Ignição:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  float buff =0;
  int j=0;
  float mme[1000000];

  if ((a=fopen("./filtro/Vehicle_mode_filtrado.asc","r")) == NULL) {
     printf("\nErro lendo Leitura1.asc");
     exit (0);
  }

  while (!feof(a))
  {
     fgets(str, sizeof(str), a);

          sscanf(str, "%e ", &i); //converte para um valor numérico
      mme[j] = (i-buff)*0.18 + buff;
  buff=mme[j];
  printf("%f\n", mme[j]);
 j=j+1;
  }
}
```

Discretização Ignição:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  int j=0;
  int contador=0;
  int dis[1000000];

  if ((a=fopen("./mme/Med_exp_vehicle_mode.asc","r")) == NULL) {
     printf("\nErro lendo Leitura1.asc");
     exit (0);
  }
  while (!feof(a)){
      fgets(str, sizeof(str), a);
```

```
            sscanf(str, "%f ", &i);
            if(i<=42){
            dis[j]= 0; // desligado
            }
            if(i>42 && i<65){
        dis[j]= 1; //acessorios
        }
            if(i>65){
        dis[j]= 2; // ligado
        }
   if(contador == 100){
   printf("%d\n", dis[j]);
   contador = 0;
}
  contador = contador + 1;
  j=j+1;
   }
}
```

Filtro Velocidade:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[10000];
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
   int d[8];
   char outros[9][100];
   int interessa;

   if ((a=fopen("./sinais/Velocidade.asc","r")) == NULL) {
       printf("\nErro lendo Leitura1.asc");
       exit (0);
   }

   while (!feof(a))
   {
       fgets(str, sizeof(str), a);

       //printf("%s",str);

       sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);

      printf("%d\n", interessa);

      interessa=0;

      interessa=d[2]<<8 | d[3];
      }
}
```

Média Móvel Exponencial Velocidade:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
   float i;
   float buff = 0;
   int j=0;
   float mme[1000000];

   if ((a=fopen("./filtro/Med_velocidade_filtrado.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }

   while (!feof(a))
   {
      fgets(str, sizeof(str), a);

          sscanf(str, "%e ", &i); //converte para um valor numérico
       mme[j] = (i-buff)*0.5 + buff;
   buff=mme[j];
   printf("%f\n", mme[j]);
  j=j+1;
   }
 }
```

Discretização Velocidade:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
   float i;
   int j=0;
   int contador=0;
   int dis[1000000];

   if ((a=fopen("./filtro/Velocidade_filtrado.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }
   while (!feof(a)){
       fgets(str, sizeof(str), a);
           sscanf(str, "%f ", &i);
           if(i== 65535){
           dis[j]= 0;
           }
           if(i== 0){
        dis[j]= 0;
        }
```

```
            if(i>0 && i<65535){
        dis[j]= 2;
        }
   if(contador == 33){
   printf("%d\n", dis[j]);
   contador = 0;
}
  contador = contador + 1;
  j=j+1;
   }
}
```

Dendrograma em R:

```
D <- read.table("dadosm1s.csv", sep=";",header=TRUE)
d<-D[,-9]
d.use<-d[,-1]

d.dist=dist(d.use)
d.hclust=hclust(d.dist)
plot(d.hclust)
```

Treinamento de HMM em R:

```
library(depmixS4)
library(quantmod)
library(qcc)
D <- read.table("dadosm1s.csv", sep=";",header=TRUE)
a<-D$Acelerador
c<-D$Combustivel
f<-D$Freio
p<-D$Porta
v<-D$Volume
i<-D$Ignicao
m<-D$Movimento
a<-ewmaSmooth(x=D$Time, y=D$Acelerador, lambda=0.01) # Aplicação da MME
c<-ewmaSmooth(x=D$Time, y=D$Combustivel, lambda=0.01)
f<-ewmaSmooth(x=D$Time, y=D$Freio, lambda=0.01)
p<-ewmaSmooth(x=D$Time, y=D$Porta, lambda=0.01)
v<-ewmaSmooth(x=D$Time, y=D$Volume, lambda=0.01)
i<-ewmaSmooth(x=D$Time, y=D$Ignicao, lambda=0.01)
m<-ewmaSmooth(x=D$Time, y=D$Movimento, lambda=0.01)
a<-a$y
c<-c$y
f<-f$y
p<-p$y
v<-v$y
i<-i$y
m<-m$y
HMM<-depmix(list(a~1,c~1,f~1, p~1, v~1, i~1,
m~1),nstates=3,family=list(gaussian(),gaussian(),gaussian(), gaussian(), gaussian(), gaussian(),
gaussian()),ntimes=6396) # Aplicação da função depmix
HMMfit<-fit(HMM, verbose = FALSE)
HMMpost<-posterior(HMMfit)
par(mfrow=c(3,3))
plot (a, col=2)
plot(c, col=3)
plot(f, col=4)
```

```
plot(p, col=5)
plot(v, col=6)
plot(i, col=7)
plot(m, col=8)
plot(HMMpost$state) # estado mais provável de acordo com o HMM
plot(D$Estado) # estado "real" para comparação da eficiência do método
print(HMMfit)
```

Validação do sistema em R:

```
F <- read.table("dadosm1sfinal.csv", sep=";",header=TRUE)
a1<-F$Acelerador
c1<-F$Combustivel
f1<-F$Freio
p1<-F$Porta
v1<-F$Volume
i1<-F$Ignicao
m1<-F$Movimento
a1<-ewmaSmooth(x=F$Time, y=F$Acelerador, lambda=0.01)
c1<-ewmaSmooth(x=F$Time, y=F$Combustivel, lambda=0.01)
f1<-ewmaSmooth(x=F$Time, y=F$Freio, lambda=0.01)
p1<-ewmaSmooth(x=F$Time, y=F$Porta, lambda=0.01)
v1<-ewmaSmooth(x=F$Time, y=F$Volume, lambda=0.01)
i1<-ewmaSmooth(x=F$Time, y=F$Ignicao, lambda=0.01)
m1<-ewmaSmooth(x=F$Time, y=F$Movimento, lambda=0.01)
a1<-a1$y
c1<-c1$y
f1<-f1$y
p1<-p1$y
v1<-v1$y
i1<-i1$y
m1<-m1$y
HMMfinal<-depmix(list(a1~1,c1~1,f1~1, p1~1, v1~1, i1~1,
m1~1),nstates=3,family=list(gaussian(),gaussian(),gaussian(), gaussian(), gaussian(), gaussian(),
gaussian()),ntimes=7553)
HMMfinal <- setpars(HMMfinal,getpars(HMMfit)) #aplicação dos parâmetros de aprendizado
viterbi(HMMfinal) #Gerando uma tabela de valores
```