PLATAFORMA PARA O MONITORAMENTO DE IRREGULARIDADE VEICULAR EM CIDADES INTELIGENTES

Vitor Greati*, Vinicius Ribeiro*, Ivanovitch Silva*, Allan Martins†, Eduardo Freitas*

*Instituto Metrópole Digital
Universidade Federal do Rio Grande do Norte
Natal, RN, Brasil

†Departamento de Engenharia Elétrica
Universidade Federal do Rio Grande do Norte
Natal, RN, Brasil

Emails: greati@ufrn.edu.br, viniciuscampos120@gmail.com, ivan@imd.ufrn.br, allan@dee.ufrn.br, eduardo.freitas360@gmail.com

Abstract— This paper presents an IoT platform for Smart Cities designed for the integration of security agents in the task of locating vehicles of interest through license plate recognition and queries to external databases. The platform is composed by an Android application capable of performing real time license plate recognition and a server executing a RESTful API designed to query external databases and integrate the application users. The adopted recognition method uses the text recognizer implemented in the Mobile Vision API by Google, and specific heuristics for results adequacy and refinement. Experiments were conducted considering four situations of movement of the users with respect to the vehicles in the field of view, represented by photographs and frames from video streams recorded in real open urban scenarios, revealing 86.6% of average accuracy. The phases of license plate recognition in the smartphone, communication with the API and notification of the interested users showed satisfactory efficiency, demonstrating the feasibility of the platform concept.

Keywords— License Plate Recognition, Mobile Vision API, Smart Cities, Android

Resumo— Este artigo apresenta uma plataforma de IoT para Cidades Inteligentes destinada à integração de agentes de segurança na tarefa de localização de veículos de interesse por intermédio do reconhecimento automático de suas placas de licenciamento e consulta a bases externas. A plataforma é composta por uma aplicação Android capaz de reconhecer placas em tempo real e um servidor executando uma API RESTful responsável por consultar bases externas e integrar os usuários do aplicativo. O método de reconhecimento adotado utiliza o reconhecedor de textos da Mobile Vision API, da Google, e heurísticas específicas para adequação e refinamento dos resultados. Os experimentos conduzidos consideraram quatro situações de movimento do usuário com respeito aos veículos no campo de visão, representadas em fotografias e *frames* de *streams* de vídeo obtidos em cenário urbano aberto, e revelaram acurácia média de 86,6%. O conjunto das etapas de reconhecimento das placas no *smartphone*, de comunicação com a API e de notificação apresentaram eficiência satisfatória, demonstrando a viabilidade do conceito da plataforma.

Palavras-chave— Reconhecimento Automático de Placas, Mobile Vision API, Cidades Inteligentes, Android

1 Introdução

Cidades Inteligentes são cenários urbanos comprometidos a aproveitar os mais recentes avanços em Tecnologias da Informação e Comunicação (TICs) para melhorar a qualidade de vida dos cidadãos, ao mesmo tempo que reduz os custos operacionais de governança (Pellicer et al. 2013).

A abordagem tecnológica mais proeminente para implementar Cidades Inteligentes atualmente é a da Internet das Coisas (IoT), a qual prevê meios de adquirir, transmitir, processar e publicar dados a partir de objetos de várias naturezas interconectados. Nesse paradigma, a interoperabilidade e interação entre esses elementos são favorecidas por uma completa infraestrutura de comunicação (Vashi et al. 2017).

A colaboração entre esses dois conceitos (IoT para Cidades Inteligentes) constitui, portanto, um meio eficaz de gerar respostas para desafios importantes do meio urbano (Zanella et al. 2014). Nesse contexto, a segurança pública é um campo estratégico para o desenvolvimento de soluções. Mais especificamente, aplicações adeptas de técnicas de Visão Computacional demonstram-se promissoras devido à tendência crescente de instalação de câmeras de vigilância em pontos estratégicos das cidades. Adicionalmente, a possibilidade de disseminação desses sistemas entre os cidadãos comuns, por meio de dispositivos móveis potentes e equipados com câmeras de alta qualidade, cria um cenário fértil para as Cidades Inteligentes.

Um alvo importante para essas aplicações, em particular, consiste na recuperação de veículos roubados, uma necessidade eminente devido aos altos índices desse crime a nível mundial (Interpol 2017). O Brasil, por exemplo, destaca-se negativamente nesse contexto: 552 mil veículos foram roubados em 2016, número maior do que o registrado em 2015 (Amâncio 2017). As taxas de recuperação, entretanto, não acompanham esse aumento, como observado em São Paulo, onde o número foi 12,3% menor em 2017 quando comparado ao contabilizado em 2016 (Vieira 2018).

À relevância do problema, soma-se o cená-

rio favorável no campo de Visão Computacional: existem muitos algoritmos de Reconhecimento Automático de Placas (RAP) com taxas de acerto e eficiência satisfatórios (Du et al. 2013), os quais podem ser úteis na localização de veículos roubados, desde que complementados com outros recursos, como uma infraestrutura de comunicação sem fio e mecanismos de geolocalização.

Diante disso, este trabalho propõe uma plataforma de IoT para Cidades Inteligentes destinada à detecção e à comunicação em tempo real de dados sobre características e localização de veículos de interesse por intermédio do reconhecimento de suas placas de licenciamento e consulta a bases externas. O objetivo é promover a ação integrada de agentes de segurança, bem como da própria população, na detecção de veículos roubados ou sob investigação.

O sistema prevê que cada usuário instale uma aplicação Android responsável por reconhecer as placas de licenciamento dispostas no campo de visão da câmera do dispositivo móvel, e enviar as sequências de caracteres contidas nas placas reconhecidas e a sua localização atual para o servidor da plataforma. Este é capaz de consultar bases externas, com informações sobre a situação legal dos veículos, e comunicar, a todos os demais agentes, os veículos de interesse avistados. Em um mapa exibido pela aplicação, caso o usuário esteja próximo ao local em que o veículo fora avistado, um ícone exibe a localização aproximada, agilizando tomadas de decisão.

Na literatura especializada, não se encontram propostas de aplicações como a apresentada neste trabalho. Não obstante, algumas se assemelham por apresentarem esquemas e arquiteturas voltadas à segurança veicular e de tráfego em Cidades Inteligentes (Chen et al. 2016, Agarwal et al. 2018).

Apesar dessa ausência no meio acadêmico, sistemas semelhantes funcionam em algumas cidades dos Estados Unidos há anos[1]. As viaturas são equipadas com várias câmeras, as quais captam imagens de diferentes ângulos para serem processadas e as placas dos veículos, reconhecidas. Um banco de dados contendo registros de placas de veículos roubados é embarcado também nas viaturas, permitindo rápida identificação. Não estão disponíveis, entretanto, informações sobre o poder de processamento e o custo exigidos por essas soluções.

Assim, este trabalho se destaca por propor um sistema de detecção de carros roubados em tempo real que demanda apenas dispositivos Android conectados à Internet e um servidor. As principais contribuições deste artigo podem ser resumidas nos seguintes tópicos:

1. Uma plataforma de segurança para Cidades Inteligentes no contexto de IoT que integra agentes e cidadãos para agilizar a recuperação de veículos roubados;
2. Reconhecimento Automático de Placas (RAP) eficiente em dispositivos Android por intermédio da tecnologia Mobile Vision, desenvolvida pela Google, aliada a heurísticas e técnicas complementares para reduzir os erros cometidos;
3. Avaliação desse método em *frames* capturados de *streams* de vídeo em diferentes situações reais de trânsito e estacionamento.

[1] http://goo.gl/Y3XnCe

2 Sistema proposto

O sistema está arquitetado em duas camadas: a do servidor, que coordena a integração entre os dispositivos e as bases de dados externas, e a de aplicação, caracterizada pelos usuários do aplicativo *Android*.

Tabela 1: URIs disponibilizadas pela API.

URI	Método	Descrição
/vehicles/appearances	POST	Submeter uma aparição de veículo.
	GET	Consultar aparições por filtros de data e placa.
/topics/subscriptions	POST	Inscrição em um tópico do FCM.
	DELETE	Cancelar inscrição em um tópico do FCM.

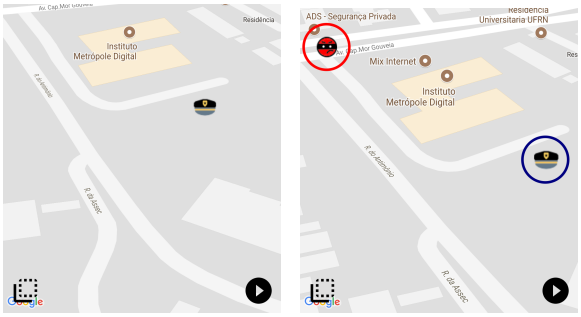
2.1 Servidor

O servidor da arquitetura é responsável por captar, por meio de requisições HTTP, os números das placas processadas nos aplicativos e as coordenadas geográficas; consultar bases de dados externas sobre dados de irregularidades; armazená-las em um banco de dados e notificar todos os demais usuários sobre a ocorrência e localização dos veículos de interesse.

O atendimento às requisições é orquestrado por uma API RESTful implementada em Node.js[2], tecnologia lançada em 2009 com o objetivo de aplicar a linguagem Javascript em aplicações no lado do servidor. Ela disponibiliza URIs (*Unique Resource Identifiers*), resumidas na Tabela 1, para se submeterem e consultarem as aparições de veículos, e para a inscrição de usuários em geradores de notificações.

As notificações são gerenciadas pela ferramenta *Firebase Cloud Messaging*[3] (FCM), a qual admite a criação de tópicos de interesse e a associação de usuários (aplicativos) a eles, através de *tokens*. Por exemplo, uma vez que certa placa é reconhecida como sendo de um carro roubado, a

[2] https://nodejs.org/en/
[3] http://goo.gl/3kfaYq



(a) Interface da aplicação mostrando a localização do usuário em tempo real.

(b) Interface da aplicação mostrando a localização após o recebimento das notificações do servidor.

Figura 1: Interfaces da aplicação antes e depois da atualização.

Figura 2: Interface da aplicação mostrando o reconhecimento automático das placas em tempo real.

FCM envia uma mensagem para todos os aplicativos registrados no tópico de veículos roubados, permitindo a integração em tempo real dos agentes de segurança na tarefa de localização e recuperação do veículo (Figura 1 (b)).

2.2 Aplicação

O aplicativo Android da plataforma foi desenvolvido diretamente na linguagem Java, sendo compatível com as versões *Jelly Bean* (4.1) e superiores. Ele realiza duas operações fundamentais: o reconhecimento automático de placas e o rastreamento das viaturas policiais e veículos de interesse em tempo real, a partir do envio e recebimento de dados por meio da API executando no servidor.

A aplicação pode ser utilizada de maneira passiva ou ativa. Na primeira, o usuário tem a visualização de um mapa (Figura 1), no qual é possível acompanhar a sua movimentação, assim como a de outros agentes e veículos de interesse encontrados por outros usuários. A forma ativa envolve alimentar o sistema com o reconhecimento de placas em tempo real, podendo ser usado com a mesma interface mencionada anteriormente, porém com o reconhecimento acontecendo em segundo plano, ou pode-se habilitar a visualização da câmera (Figura 2), pela qual é possível observar o cenário e as detecções sendo realizadas em tempo real.

O rastreamento dos veículos é realizado por intermédio do Sistema de Posicionamento Global (GPS), já integrado aos *smartphones*, utilizando a *Google Maps Android API*[4] para manipular a interface do mapa e obter as coordenadas geográficas. O mapa apresenta marcadores para cada tipo de indivíduo rastreado, como definido na Tabela 2.

Marcador	Especificação
	Usuário da aplicação.
	Outros usuários próximos à localização atual.
	Veículos em algum estado de irregularidade.

Tabela 2: Tabela de marcadores da aplicação.

O processo de atualização de coordenadas geográficas dos agentes ocorre quando se distanciam em pelo menos 10 m da posição atual, ou caso já se tenha passado 1 min desde a última localização. As localizações dos veículos de interesse são atualizadas à medida que suas placas são identificadas pela aplicação, considerando-se a última localização conhecida do policial como a nova posição geográfica desses veículos.

2.3 Método de reconhecimento de placas

A abordagem tradicional para lidar com o reconhecimento automático de placas de licenciamento baseia-se em três etapas: detecção da placa na cena, segmentação dos caracteres de cada placa e reconhecimento ótico de cada caractere. A performance desse fluxo de execução geralmente encontra, na primeira etapa, um gargalo que prejudica a execução em dispositivos de capacidade limitada, como são os *smartphones*.

Tendo isso em vista, decidiu-se por adotar outra estratégia: dada uma cena, aplicar um algoritmo eficiente de reconhecimento de textos – neste caso, o provido pela Mobile Vision API – para capturar todas as cadeias de caracteres; em seguida, selecionar, utilizando um filtro específico, apenas as que conformam com o padrão esperado para o código de licenciamento dos veículos brasileiros; e, finalmente, refinar o resultado aplicando técnicas complementares. O diagrama presente na Figura 3 ilustra o método proposto, e as subseções seguintes detalham cada etapa.

2.3.1 Mobile Vision API

Desenvolvida pela Google, a Mobile Vision API fornece um *framework* para detecção de objetos em fotos e vídeos em aplicativos Android. Três detectores prontos são distribuídos com o *fra-*

[4] http://goo.gl/9qq7X1

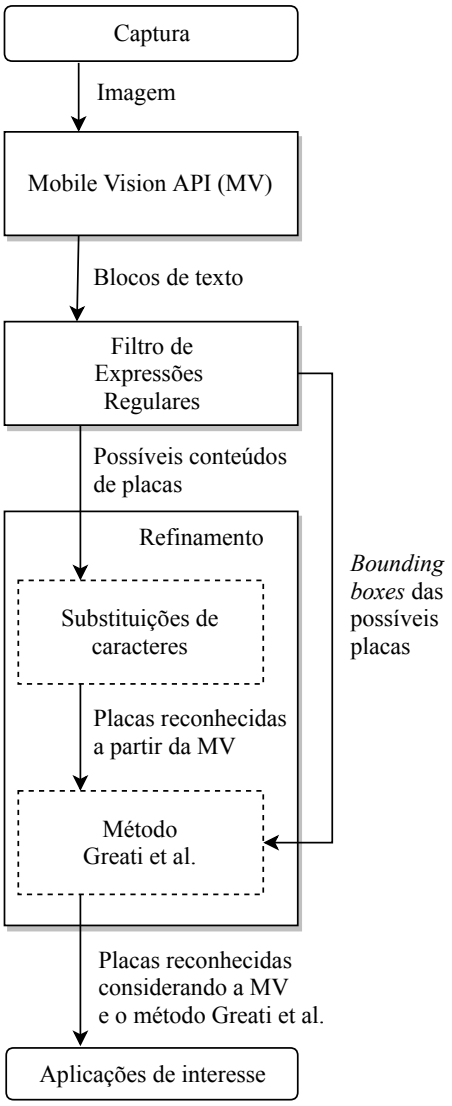


Figura 3: Método de reconhecimento de placas proposto.

mework: um para faces, outro para códigos de barra e outro para textos.

O último, de interesse para a aplicação proposta, reconhece textos de qualquer língua latina, e utiliza três níveis de segmentação: bloco, como um conjunto de linhas; linha, como um conjunto de palavras no mesmo eixo vertical; e palavra, como uma sequência de caracteres também no mesmo eixo vertical.

Neste trabalho, a aplicação Android da plataforma utiliza o detector de textos a nível de bloco sobre imagens capturadas diretamente da câmera do dispositivo hospedeiro. Para cada imagem, obtém-se um conjunto de blocos, cada qual contendo uma sequência de linhas. Cada linha é composta de uma sequência de palavras que demanda ser filtrada, objetivando identificar apenas as cadeias representantes de placas de licenciamento.

2.3.2 Filtro

Seja N o número de blocos obtidos a partir do algoritmo de reconhecimento de textos da Mobile Vision API para determinada imagem e N_i a quantidade de linhas no bloco i, com $i = 1, 2, \ldots, N$. Aplica-se uma substituição de cada caractere hífen presente em cada linha por um caractere espaço. Considere $\mathcal{F}_{ij} = \{c_{ijk}\}_{k=1}^{N_{ij}}$ a sequência das N_{ij} cadeias, obtidas após essas substituições, dentro da j-ésima linha do i-ésimo bloco, $j = 1, 2, \ldots, N_i$, sobre o alfabeto $\Sigma = \{0, 1, \ldots, 9\} \cup \{A, B, \ldots, Z\}$, tal que c_{ijk} aparece imediatamente antes de $c_{ij(k+1)}$, com $k = 1, 2, \ldots, (N_{ij} - 1)$, na linha j do bloco i.

O filtro utilizado para selecionar os membros de cada $\mathcal{F}_{ij}$ que conformam com o padrão das placas brasileiras aplica as expressões regulares $\phi_1 = [a - zA - Z0158]\{3\}$, $\phi_2 = [0 - 9iIoOqQdDsSbB]\{4\}$ e $\phi_3 = \phi_1\phi_2$, pertencentes ao conjunto das expressões regulares interpretáveis pela linguagem Java. Isso é feito por meio da função $f_{ij} : \{1, \ldots, N_{ij} + 2\} \to \mathcal{P}(\mathcal{F}_{ij}^2)$, assim definida:

$$f_{ij}(k) = \begin{cases} \emptyset & \text{se } k > N_{ij}, \\ \{c_{ijk}\} \cup f_{ij}(k+1) & \text{se } k \leq N_{ij} \wedge \\ & \phi_3(c_{ijk}), \\ \{c_{ijk}c_{ij(k+1)}\} \cup f_{ij}(k+2) & \text{se } k < N_{ij} \wedge \\ & \phi_1(c_{ijk}) \wedge \\ & \phi_2(c_{ij(k+1)}), \\ f_{ij}(k+1) & \text{caso contrário.} \end{cases} \tag{1}$$

onde c_ic_j é a concatenação da cadeia c_i com a cadeia c_j, $\mathcal{F}_{ij}^2 = \{xy \mid x, y \in \mathcal{F}_{ij}\}$, $\mathcal{P}(\mathcal{F}_{ij}^2)$ é o conjunto das partes de $\mathcal{F}_{ij}^2$, e $\phi_j(c_i)$ é verdadeiro quando c_i casa com a expressão regular ϕ_j. A Figura 4 exemplifica o uso do filtro.

O conjunto das cadeias consideradas como placas pelo algoritmo de filtragem é, então, dado por

$$\mathcal{P} := \bigcup_{i=1}^{N} \bigcup_{j=1}^{N_i} f_{ij}(1). \tag{2}$$

2.3.3 Refinamento

Esta etapa existe para lidar com as possíveis confusões entre caracteres cometidas pelo algoritmo de reconhecimento de textos, e ocorre em duas fases.

A primeira lida com os casos em que, sob o pressuposto de que $\mathcal{F}_{ij}$ contenha conteúdos de placas brasileiras, haja ocorrência(s) de número(s) nas três primeiras posições ou de letras nas quatro restantes, o que caracterizaria certamente um engano. A fim de lidar com essas situações, são aplicadas funções de substituição de caracteres $\sigma_L : \Sigma \to \Sigma$ e $\sigma_N : \Sigma \to \Sigma$, definidas por:

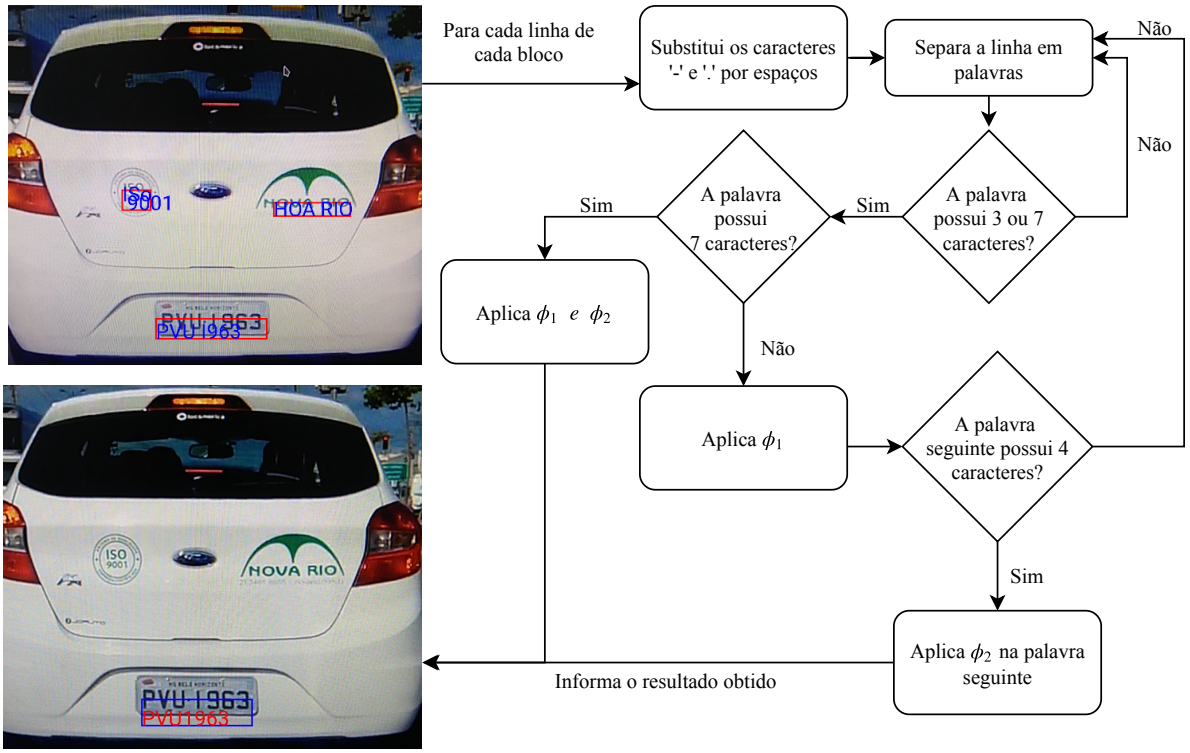


Figura 4: Exemplo de funcionamento do filtro de expressões regulares.

$$\sigma_N(\alpha) = \begin{cases} O & \text{se } \alpha = 0, \\ I & \text{se } \alpha = 1, \\ S & \text{se } \alpha = 5, \\ B & \text{se } \alpha = 8, \\ \alpha & \text{caso contrário.} \end{cases} \tag{3}$$

$$\sigma_L(\alpha) = \begin{cases} 1 & \text{se } \alpha \in \{I, L\}, \\ 0 & \text{se } \alpha \in \{O, D, Q\}, \\ 5 & \text{se } \alpha = S, \\ 8 & \text{se } \alpha = B, \\ \alpha & \text{caso contrário.} \end{cases} \tag{4}$$

Como, por definição, todas as cadeias em $\mathcal{P}$ possuem sete caracteres necessariamente, considere cada $c = \alpha_1\alpha_2\ldots\alpha_7 \in \mathcal{P}$ e seja $\sigma_1 : \Sigma^7 \to \Sigma^7$ definida por

$$\sigma_1(c) = \sigma_N(\alpha_1)\sigma_N(\alpha_2)\sigma_N(\alpha_3)\sigma_L(\alpha_4)\ldots\sigma_L(\alpha_7)$$

Assim, o resultado desse passo é o conjunto

$$\mathcal{P}^* := \{\sigma_1(c) \mid c \in \mathcal{P}\} \tag{5}$$

das cadeias resultantes das substituições especificadas acima.

O segundo passo do refinamento é motivado pela observação de que as confusões entre os caracteres O e Q ocorrem com muita frequência. A estratégia adotada para lidar com esse fato consiste em aplicar as etapas de segmentação e reconhecimento de caracteres do método descrito em Greati et al. (2017), de abordagem clássica, considerando a Mobile Vision API como responsável pela primeira etapa, a de detecção da placa. Ou seja, obtém-se o *bounding box* da placa detectada e executa-se tal método sobre ele. Não é esperada uma queda de performance considerável, porque as duas etapas aproveitadas são muito eficientes.

Considerando $c = \alpha_1\alpha_2\ldots\alpha_7 \in \mathcal{P}^*$, a substituição baseada nesse outro método, denominada aqui de $\sigma_2^c : \{\alpha_1, \alpha_2, \alpha_3\} \to \{A, B, \ldots, Z\}$, é definida por

$$\sigma_2^c(\alpha_i) = \begin{cases} \psi_c(\alpha_i) & \text{se } \alpha_i \in \{O, Q\}, \\ \alpha_i & \text{caso contrário.} \end{cases} \tag{6}$$

onde $\psi_c(\alpha_i)$ corresponde à resposta da abordagem clássica para o caractere na posição i da cadeia c, quando ela existe, ou o próprio α_i, caso contrário. Isso significa que a decisão da Mobile Vision API no que tange aos caracteres Q e O sempre é ignorada caso o outro método tenha reconhecido a placa.

O conjunto final das placas reconhecidas é, portanto, definido por

$$\mathbb{P} := \{\sigma_2^c(\alpha_1)\sigma_2^c(\alpha_2)\sigma_2^c(\alpha_3)\alpha_4\ldots\alpha_7 \mid \alpha_1\alpha_2\ldots\alpha_7 \in \mathcal{P}^*\}.$$

3 Experimentos

As seguintes circunstâncias possíveis no cenário de execução do aplicativo guiaram os experimentos:

1. Quando o usuário e os demais veículos estão parados, caso interpretável como uma simples fotografia;

(a) Situação 1.

(b) Situação 2.

(c) Situação 3.

(d) Situação 4.

Figura 5: Exemplos de imagens dos cenários considerados.

2. Quando o usuário está em movimento e os demais veículos estão parados, tipicamente situação de vistoria em estacionamentos;

3. Quando o usuário está parado e os demais veículos estão em movimento, caso tradicional de aplicações policiais de reconhecimento de placas, em que o agente aponta o *smartphone* para os veículos durante uma *blitz*;

4. Quando tanto o usuário, quanto os demais veículos estão em movimento, caso em que o agente está em patrulha enquanto trafega em meio ao trânsito.

A fim de se verificar a factibilidade do uso da plataforma no contexto pretendido, foram criados quatro cenários de testes para representar os casos apresentados anteriormente, um utilizando fotografias e os demais, gravações de vídeo. Ao invés de se processarem os vídeos diretamente, extraíram-se os *frames* previamente, de modo a facilitar a execução dos experimentos. A Tabela 3 apresenta as características dos artefatos utilizados em cada cenário de teste.

Tabela 3: Especificações do conjunto de testes.

Sit.	Duração	Fps	Resolução	Frames
1	-	-	$3264\,px \times 1836\,px$	481
2	$7'29''$	30	$1920\,px \times 1080\,px$	13470
3	$16'50''$	5	$1600\,px \times 1200\,px$	5050
4	$3'06''$	30	$1920\,px \times 1080\,px$	5580

Na primeira situação foi adotado um conjunto de imagens obtidas no estacionamento do Instituto Metrópole Digital (IMD), situado na Universidade Federal do Rio Grande do Norte (UFRN), por um *tablet* Samsung Galaxy Tab 10 - 2014 Edition, o mesmo *dataset* utilizado nos experimentos de Greati et al. (2017). Na segunda, utilizou-se um vídeo obtido no mesmo local e pelo mesmo *tablet* da situação anterior. Na terceira, foi utilizado um vídeo gravado por uma câmera pública de Porto Alegre[5] (Rio Grande do Sul, Brasil), onde cada cena possui resolução de $800\,px \times 600\,px$, e que, para se adequar ao sistema, teve sua dimensão dobrada de tamanho. Por fim, na última situação, utilizou-se uma filmagem[6] a partir de um veículo em movimento. As Figuras 5 (a), 5 (b), 5 (c), 5 (d) mostram os exemplos de placas existentes em cada *dataset*.

Vale salientar que as duas últimas situações poderiam ser reproduzidos com uma câmera comum de um *smartphone* devidamente configurada, o que não foi realizado por questões de logística. Além disso, para a segunda e a quarta situações, apenas $\frac{1}{6}$ dos *frames* foram considerados, para simular uma taxa de $5\,fps$ nas respectivas gravações a $30\,fps$ originalmente.

Para cada uma dessas situações, a sequência de imagens foi processada pelo método de reconhecimento em um *smartphone* Moto G5S Plus, da fabricante Motorola, o qual conta com um processador octa-core de velocidade $2\,GHz$ e $3\,GB$ de memória RAM. A contabilização das placas foi feita manualmente, bastando que cada placa esperada aparecesse ao menos uma vez entre as reconhecidas.

Em alguns desses experimentos, uma vez que uma placa fosse reconhecida, outro dispositivo executando a aplicação aguardava para receber

[5] http://bit.ly/2GzSas1
[6] http://goo.gl/SWoBxs

Tabela 4: Resultados dos experimentos para cada situação.

Situação	Acurácia	Porcentagem
1	167/198	84,34%
2	83/86	96,51%
3	65/77	84,41%
4	60/72	83,30%
Total	375/433	86,61%

uma notificação se o veículo fosse classificado como "roubado" após a consulta a uma base externa, almejando verificar a dinâmica esperada para a plataforma.

4 Resultados

A Tabela 4 mostra os resultados obtidos nos experimentos para cada uma das quatro situações consideradas. Nela, é mostrada a acurácia do sistema em razão do número de placas distintas acertadas pelo número total de placas presente nos conjuntos de *frames*, além da porcentagem que esses dados representam.

O método de reconhecimento aplicado à primeira situação resultou em uma acurácia de 84,34%, superior à obtida em Greati et al. (2017), o qual utilizou o mesmo conjunto de imagens, mas aplicando o método aproveitado para etapa de refinamento deste trabalho (ver Seção 2.3.3). É importante ressaltar que muitas das placas apresentam inclinação considerável, uma propriedade pouco observada no cenário de aplicação pretendido.

Os resultados de maior destaque foram alcançados na segunda situação, com 97% de acurácia, evidenciando a validade do método para a aplicação em vistorias de estacionamentos.

Para a terceira situação, apesar de os *frames* apresentarem menor resolução em comparação aos outros casos e mesmo tendo sua resolução aumentada e perdendo um pouco de qualidade, apresentou a segunda melhor acurácia, com 84,41%. Isso revela que a plataforma também é aplicável às situações de *blitz* e monitoramento do trânsito.

A última, e mais problemática, por geralmente apresentar *frames* mais ruidosos e borrados, também mostrou-se adequada à solução proposta, com acurácia de 83,3%. Essa situação é típica das patrulhas policiais, ações frequentemente realizadas e com grande potencial de localização de um elevado volume de veículos.

Constatou-se que o tempo de resposta do método para cada *frame* foi de aproximadamente $300\,ms$. Quanto à dinâmica de comunicação com a API do servidor, percebeu-se um tempo de resposta satisfatório. Este, entretanto, pode variar de acordo com a velocidade de resposta das bases externas, condição a ser cuidadosamente considerada no contexto de aplicação real.

Outra observação relevante é a de que a plataforma, por implementar uma API RESTful, pode aceitar requisições não apenas de *smartphones* Android, mas de dispositivos de quaisquer naturezas conectados à Internet. Esse fato demonstra a flexibilidade do sistema proposto e ressalta sua imersão no contexto de IoT.

5 Conclusões

Este artigo apresenta uma plataforma para Cidades Inteligentes composta por dois componentes: uma aplicação Android, responsável principalmente por reconhecer as placas dos veículos por meio de imagens; e uma API RESTful que, ao ser informada sobre essas placas, armazena as ocorrências, verifica fontes externas e notifica outros usuários sobre a localização dos veículos de interesse encontrados.

O método de reconhecimento automático de placas executado no aplicativo combina o reconhecedor de textos da Mobile Vision API, produzida pela Google, com filtro por expressões regulares e refinamentos específicos ao padrão brasileiro, incluindo o uso de um segundo reconhecedor eficiente para os casos em que a Mobile Vision API se equivoca com frequência. Considerando todas as situações de experimentação, a acurácia média do método foi de 86,6%. Além disso, o algoritmo se mostrou eficiente mesmo executando em *smartphones* e o tempo de resposta da API implementada foi satisfatório, validando a viabilidade da plataforma.

Os trabalhos futuros se concentrarão na apuração da quantidade de falsos positivos do método e no desenvolvimento de uma estratégia de processamento dos *frames* de vídeo considerando intervalos de tempo, de forma que sejam detectadas aparições sucessivas de um mesmo veículo, para evitar tráfego excessivo de dados e reduzir o número de classificações incorretas.

Agradecimentos

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), à Secretaria da Segurança Pública e da Defesa Social do Rio Grande do Norte (Sesed), à UFRN, ao IMD e ao projeto Smart Metrópolis, por todo o aporte financeiro e estrutural.

Referências

Agarwal, Y., Jain, K. & Karabasoglu, O. (2018). Smart vehicle monitoring and assistance using cloud computing in vehicular Ad Hoc networks, *International Journal of Transportation Science and Technology* **7**(1): 60–73.

Amâncio, T. (2017). Brasil tem 1 roubo ou furto de veículo a cada minuto; Rio lidera o ranking, Folha de São Paulo.

Chen, N., Chen, Y., You, Y., Ling, H., Liang, P. & Zimmermann, R. (2016). Dynamic Urban Surveillance Video Stream Processing Using Fog Computing, *2016 IEEE Second International Conference on Multimedia Big Data (BigMM)*, IEEE, pp. 105–112.

Du, S., Ibrahim, M., Shehata, M. & Badawy, W. (2013). Automatic License Plate Recognition (ALPR): A State-of-the-Art Review, *IEEE Transactions on Circuits and Systems for Video Technology* **23**(2): 311–325.

Greati, V. R., Ribeiro, V. C. T., Silva, I. M. D. & Martins, A. d. M. (2017). Método para Reconhecimento Automático de Placas Brasileiras em ambientes não controlados, *Simpósio Brasileiro de Automação Inteligente (SBAI 2017)*.

Interpol (2017). Database statistics, vehicle crimes, crime areas, https://www.interpol.int/Crime-areas/Vehicle-crime/Database-statistics. Accessed: 2017-05-19.

Pellicer, S., Santa, G., Bleda, A. L., Maestre, R., Jara, A. J. & Skarmeta, A. G. (2013). A Global Perspective of Smart Cities: A Survey, *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013*, IEEE, pp. 439–444.

Vashi, S., Ram, J., Modi, J., Verma, S. & Prakash, C. (2017). Internet of Things (IoT): A vision, architectural elements, and security issues, *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp. 492–496.

Vieira, A. (2018). Recuperação de veículos roubados bate recorde negativo em São Paulo, Notícias Band.

Zanella, A., Bui, N., Castellani, A., Vangelista, L. & Zorzi, M. (2014). Internet of Things for Smart Cities, *IEEE Internet of Things Journal* **1**(1): 22–32.