


Car-Finding System with CouchDB-Based Sensor Management Platform



Banco de dados NoSQL - CouchDB

Luciano Brum
Rebeca Fiss



Introdução

Desenvolver uma ferramenta que faça o reconhecimento automático das placas dos veículos que circulam pela cidade, com o objetivo de diminuir o número de furtos de veículos.

Introdução

- Utilização de dispositivos móveis
 - Gerenciar redes sem fio
 - Otimização de protocolos
 - Gestão de energia
- Inviável a centralização do processamento
 - Os dispositivos devem realizar o processamento e enviar apenas resultados relevantes
- Reconhecimento de placas
 - Extração e reconhecimento de caracteres

Arquitetura da Plataforma

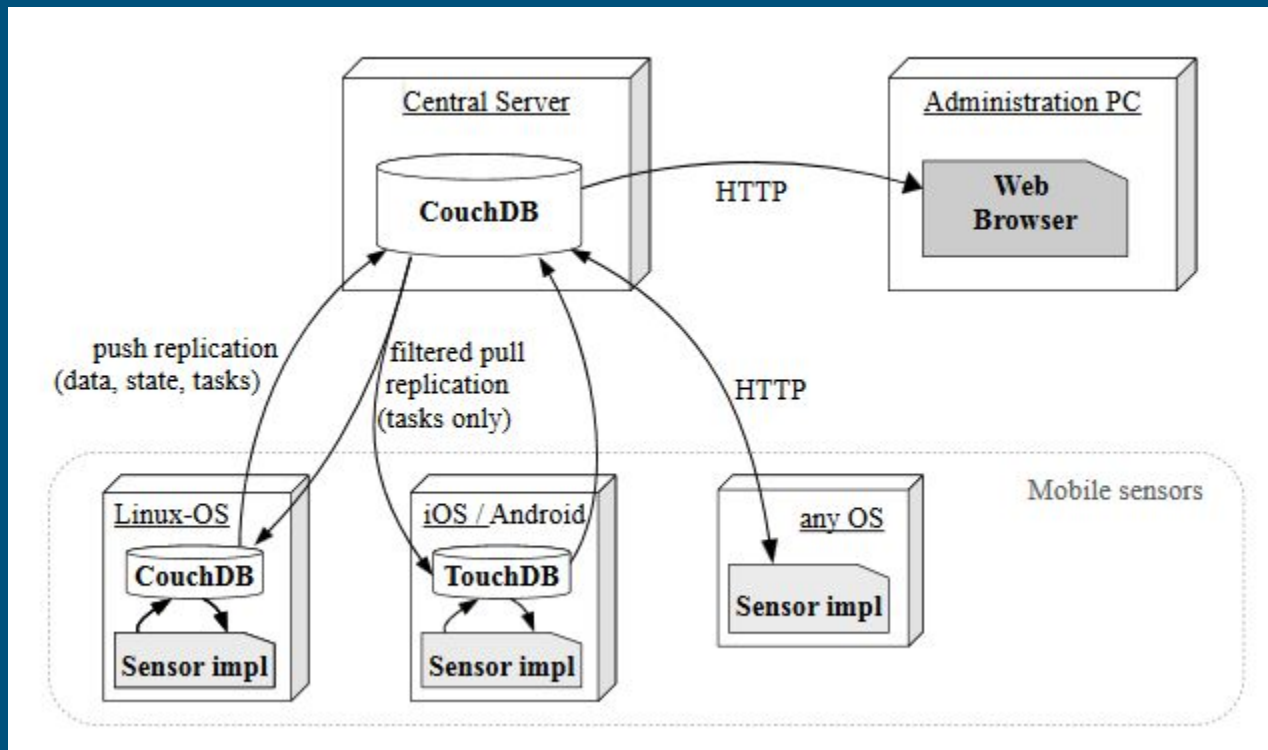


Figura 1: Arquitetura da Plataforma: Fonte: NOCUN et al., 2017.

Arquitetura da Plataforma

- Sistema de gerenciamento de sensores:
 - Monitoramento estado dos sensores
 - Definir tarefas a serem executadas
 - Armazenar informações
- Sistema baseado em CouchDB

CouchDB

- Banco de Dados NoSQL, Schema-free
- Banco de dados orientado a documentos
 - Formato JSon
 - Outros dados são armazenados como anexo
- Cada documento possui um ID e não possui relação com os demais
- Para se recuperar os dados é preciso criar Views, com JavaScript
 - O arquivo é inspecionado e atribuído ou não à view
 - A view é armazenada no banco, também como um arquivo, mas em outro formato
 - MapReduce

CouchDB

- Mecanismo de replicação de dados
- Tolerância a falhas
- Suporta diversas instâncias geograficamente distribuídas
 - Ou apenas uma
- Interação é feita via HTTP

CouchDB

- Foi escolhido para o sistema por
 - Banco orientado a documentos
 - Replicação de dados nativa
 - Interface HTTP
- Utilizado para
 - Fornecer aplicação web
 - Gerenciamento de sensores com uma instância local do banco, e replicação no servidor web
 - Armazenar e distribuir tarefas dos sensores
 - Coletar as informações dos sensores
 - No servidor web ou no banco local

Sensores

- Grupo de Sensores, que capturam dados , realizam análise, e guardam as informações em um banco de dados.
- Por se comunicar com o servidor via HTTP, é possível utilizar diversas plataformas (IOS, Android, Windows, Linux).
- Os dados são armazenados primeiramente no banco local do dispositivo, por conta da dificuldade de conexão com a internet.
- A replicação automática para um servidor é configurada.
 - Sistema tolerante à falhas
 - Testes comprovaram 100% dos dados replicados

Sensores

- Instalação do CouchDB em dispositivos móveis é limitada, quando não se tem acesso root ao sistema.
- Duas possibilidades:
 - Couchbase Mobile
 - ThouchDB
- ThouchDB é uma implementação diferente
- Mais rápido na inicialização
- Utiliza menos memória e CPU

Sensores

- A cada 30 segundos o sensor cria um novo arquivo para guardar os dados coletados.
- Quando existe a informação da localização, estes dados são adicionados à um mapa.
- As tarefas que serão executadas pelo sensor podem variar, e são armazenadas em um arquivo, que é verificado periodicamente.
- Tarefas estão armazenadas no servidor remoto, em CouchDB, são filtradas, e armazenadas no banco local do dispositivo.
- Para diminuir o número de requisições, um sistema de feed avisa os sensores quando uma nova tarefa está disponível.

Sensores

- Inicialmente foram criados 4 sensores
 - Um em Python, para monitorar a carga da CPU
 - Demais plataforma Android
- Dados obtidos pelos sensores
 - Localização GPS
 - Imagens da câmera
 - Amostras de Som
- Dados são armazenados em um arquivo JSON, com imagens e sons como anexo.

Cauchapp – Interface Gráfica

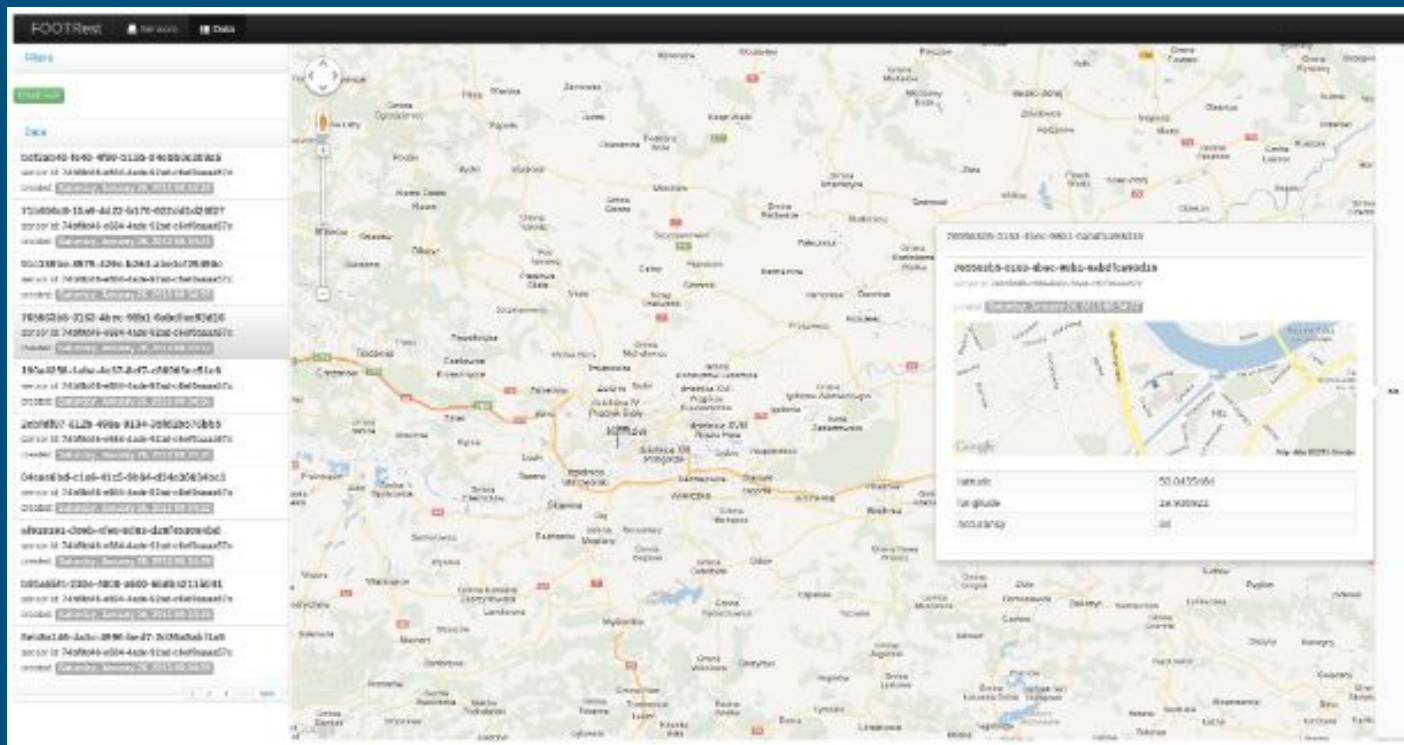


Figura 2: Interface gráfica CauchApp: Fonte: NOCUN et al., 2017.

Cauchapp – Interface Gráfica

- Interface desenvolvida para interação com o usuário
 - Monitor de estado de sensores
 - Gerenciar sensores (tarefas)
 - Visualizar dados coletados pelos sensores
- Sensores são marcados no mapa, e sempre que seus dados são atualizados, estas informações ficam disponíveis na interface.
- Usuário pode gerenciar os sensores, definir tarefas, verificar estado e ver dados coletados.
- Apresenta também os arquivos no formato JSON, com imagens e sons.
- Usuário pode criar filtros.

Recursos da Plataforma

- **Multiplataforma**
 - Único requisito é executar uma instância do CouchDB
 - Para a visualização basta um navegador web
- **Suporte a dispositivos com sensores diferentes**
 - Por utilizar interface HTTP não é necessário criar ferramenta para comunicação entre sensor e o CouchDB
- **Extensibilidade**
 - Por ser NoSQL orientado a documentos, os dados podem ter vários formatos diferentes
- **Resistência à má qualidade de rede**
 - Utilização da instância local do CouchDB
 - Replicação dos dados é tratada direto pelo CouchDB

Desempenho da Plataforma

- A plataforma foi testada para verificar sua performance sob condições de carga pesada.
- Carga Pesada:
 - Milhares de sensores enviando dados simultaneamente.
 - Vários envios por segundo para o servidor.

Desempenho da Plataforma

- O CouchDB foi inicializado no Amazon Elastic Compute Cloud (serviço que provê computação escalável e armazenamento em nuvem).
- A máquina foi paravirtualizada.

Desempenho da Plataforma

Parâmetros		Valores
Cores	4	
Ram	16 GB	
Storage	Amazon EBS (cloud)	
Operating System	Ubuntu Server 12.04 (Linux)	
CouchDB	1.2.1 (Erlang OTP R15B03-1)	
Network Interfaces	1	

Figura 3: Parâmetros da máquina utilizada para a instância do CouchDB: Fonte: NOCUN et al., 2017.

Desempenho da Plataforma

- Os sensores foram simulados por duas outras instâncias ec2 da amazon com os parâmetros descritos na figura abaixo.

Parâmetros	Valores
Cores	2
Ram	4 GB
Storage	Amazon EBS (cloud)
Operating System	Amazon Linux
<u>Tsung</u>	1.4.1 (Erlang OTP R15B03-1)
Network Interfaces	1

Figura 4: Parâmetros da máquina utilizada para simular os sensores. Fonte: NOCUN et al., 2017.

Desempenho da Plataforma

- Cenários de teste:
 - 5000 sensores concorrentes escrevendo no banco de dados (gap 0.03s).
 - 5000 sensores concorrentes escrevendo no banco de dados enquanto usuários efetuam consultas em tempo-real.
 - Sensores carregando grandes quantidades de dados enquanto o usuário consulta os resultados em tempo real.
 - Carga máxima (12000 sensores, gap 0.01s).

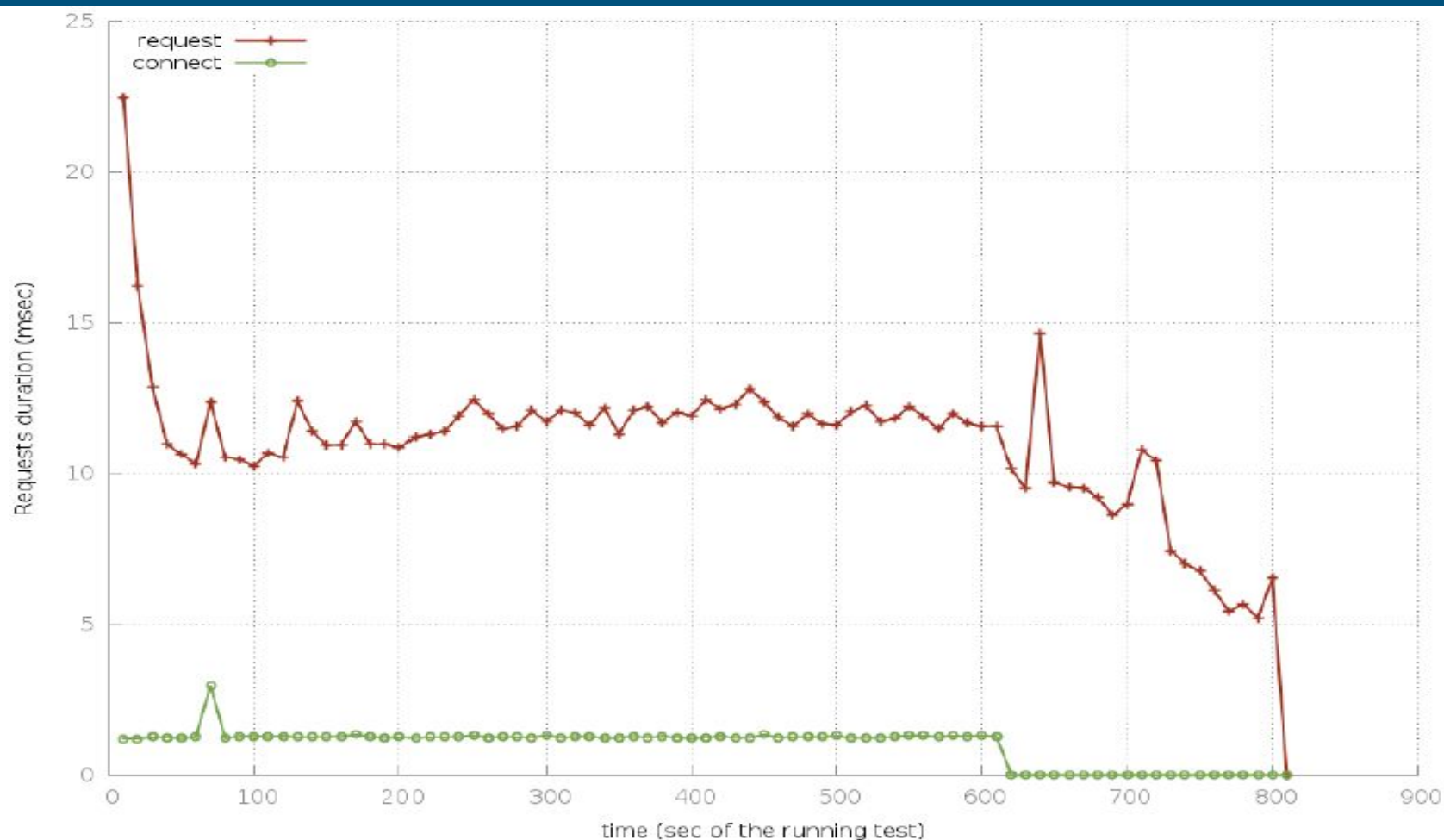


Figura 5: Duração média de conexão e requisição. Fonte: NOCUN et al., 2017.



Figura 6: Taxa de conexões TCP/UDP e conexões. Fonte: NOCUN et al., 2017.

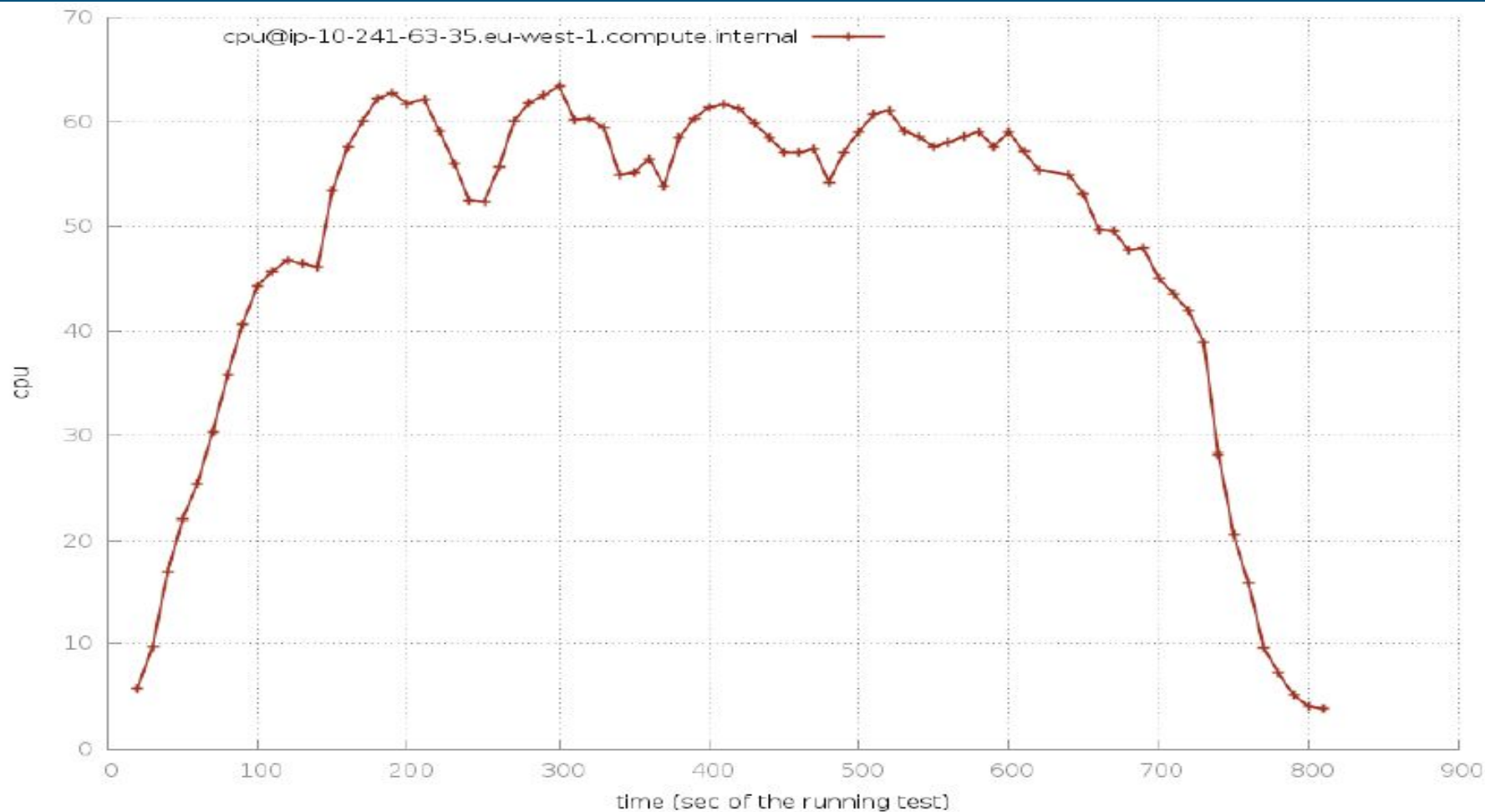


Figura 7: Porcentagem de uso da CPU no primeiro teste. Fonte: NOCUN et al., 2017.

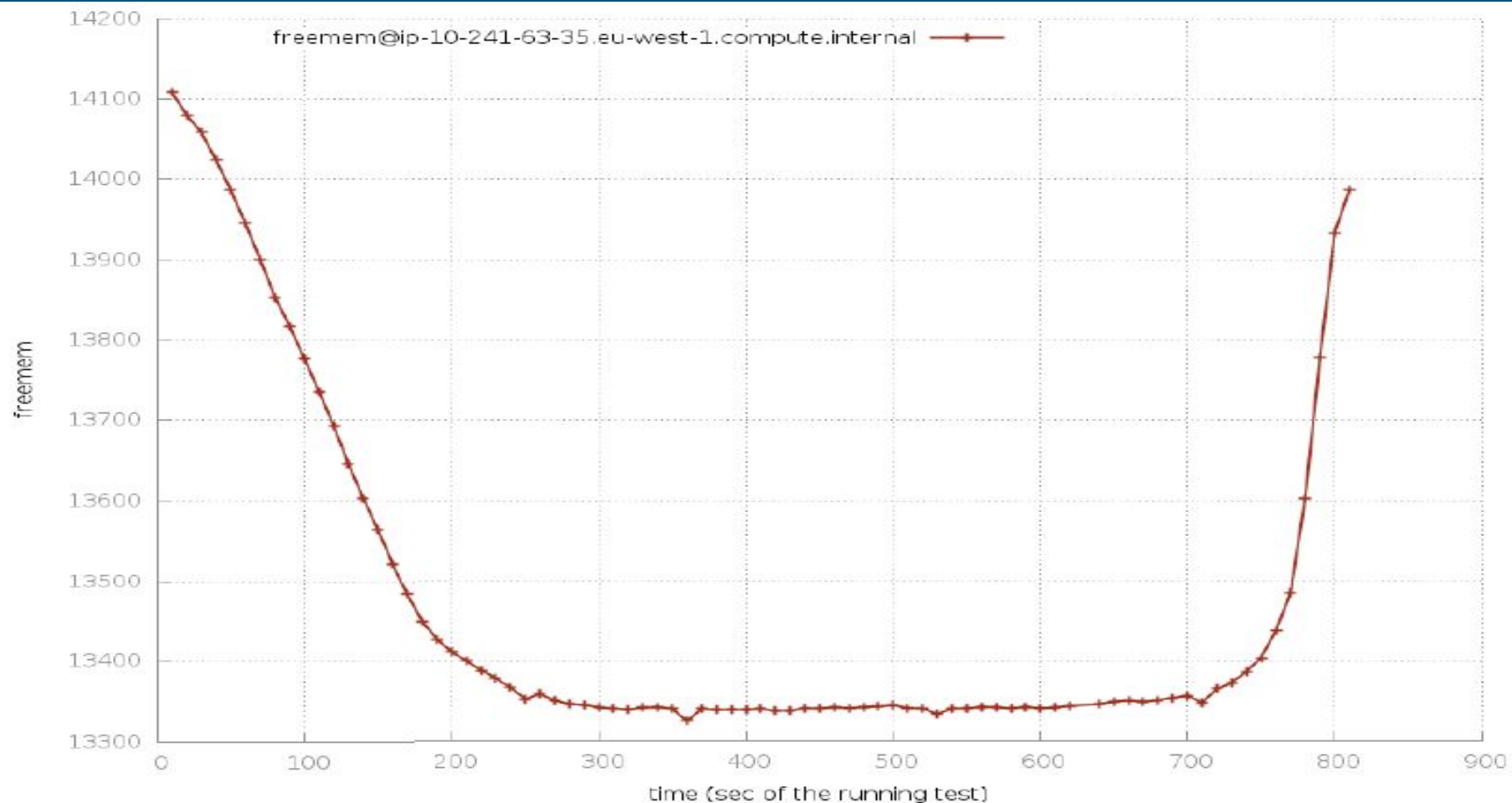


Figura 8: Consumo de memória em KB no primeiro teste. Fonte: NOCUN et al., 2017.

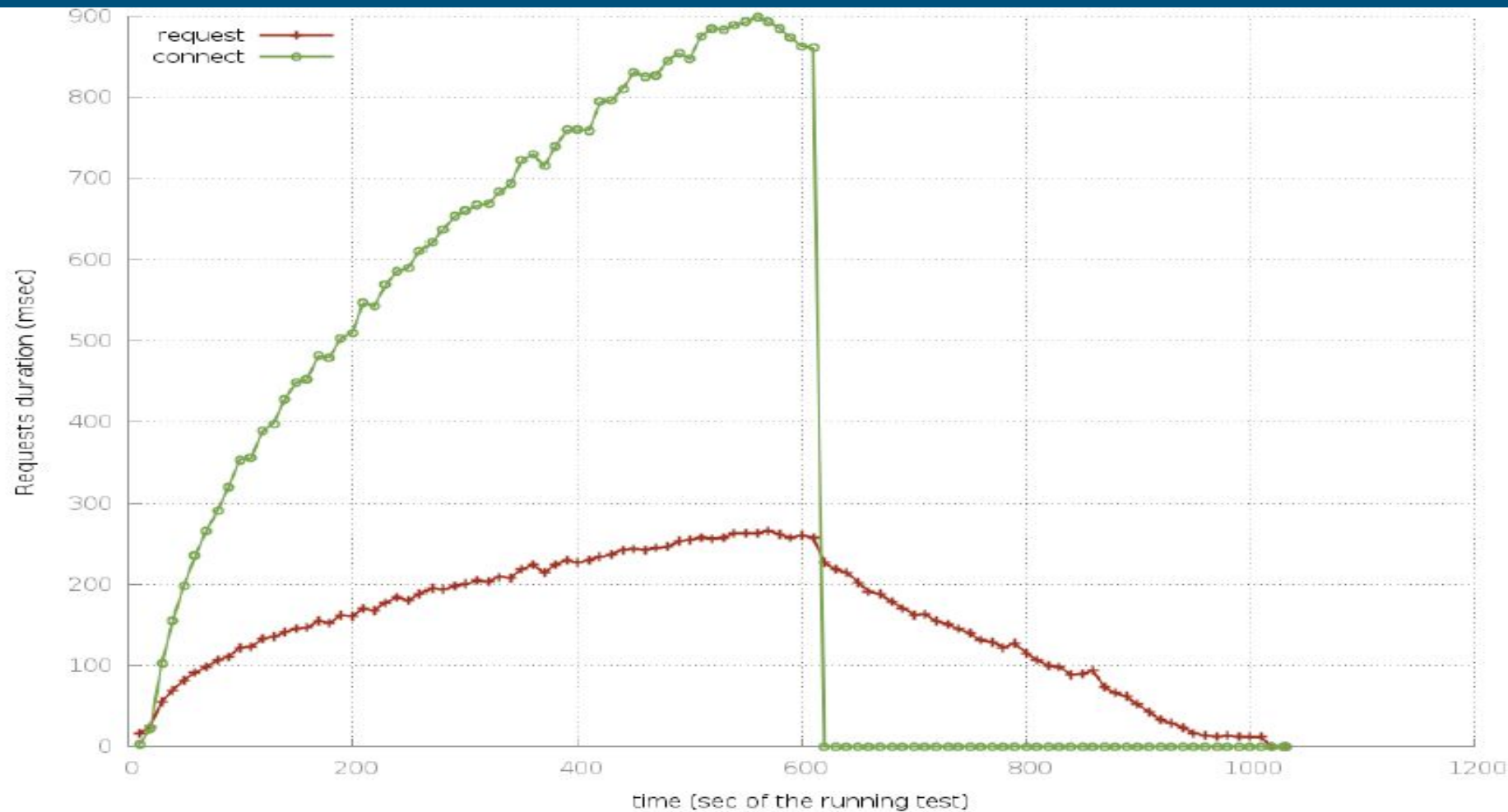


Figura 9: Média de tempo para salvar documentos no banco no terceiro teste. Fonte: NOCUN et al., 2017.

Desempenho da Plataforma

- Resumo dos testes:
 - Durante o terceiro teste, a conectividade de rede foi o gargalo.
 - A CPU e a memória quase não foram usadas.
 - A carga muito alta no quarto teste não travou o banco de dados completamente.
 - O servidor recuperou-se dos erros que ocorreram quando mais de 12.000 sensores submeteram dados para o servidor.

Algoritmo de Reconhecimento de Placas

- A tarefa do sistema projetado é encontrar uma placa em uma determinada imagem, reconhecer seus caracteres, e comparar os resultados com cada entrada da lista de carros roubados.

Algoritmo de Reconhecimento de Placas

- 1º passo: pré-processamento.



Figura 10: Pré-processamento inicial do algoritmo. Fonte: NOCUN et al., 2017.

Algoritmo de Reconhecimento de Placas

- 2º passo: Localização da placa.

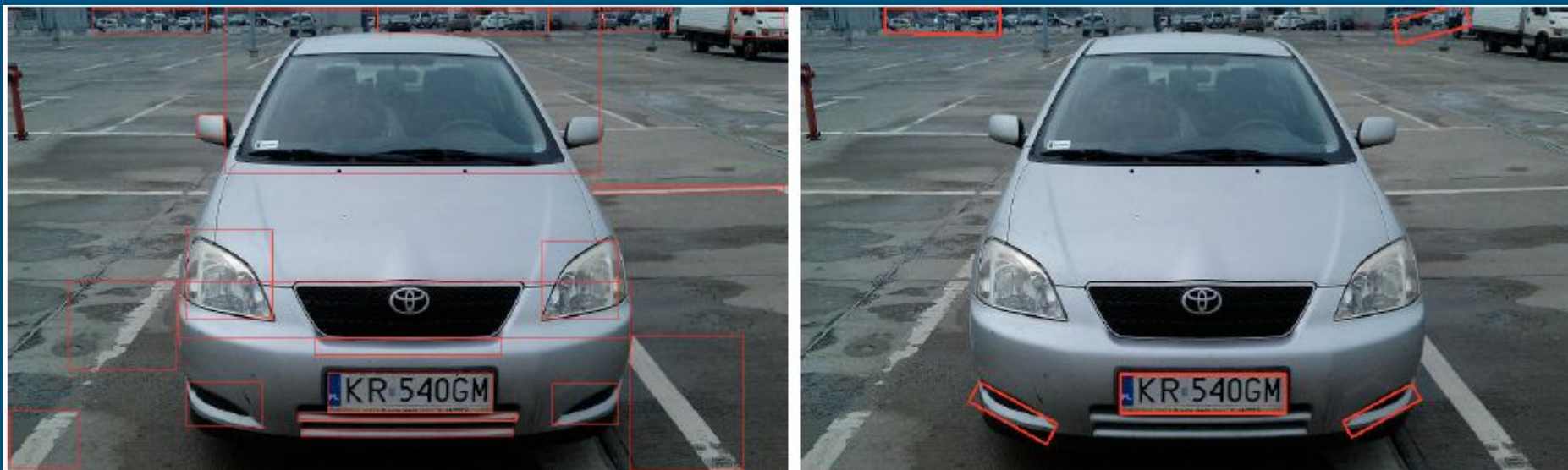


Figura 11: Localizando os padrões de placas. Fonte: NOCUN et al., 2017.

Algoritmo de Reconhecimento de Placas

- 3º passo: Separação dos caracteres.



Figura 12: Estágios de Segmentação dos caracteres. Fonte: NOCUN et al., 2017.

Algoritmo de Reconhecimento de Placas

- 4º passo: Reconhecimento de caracteres (kNN).

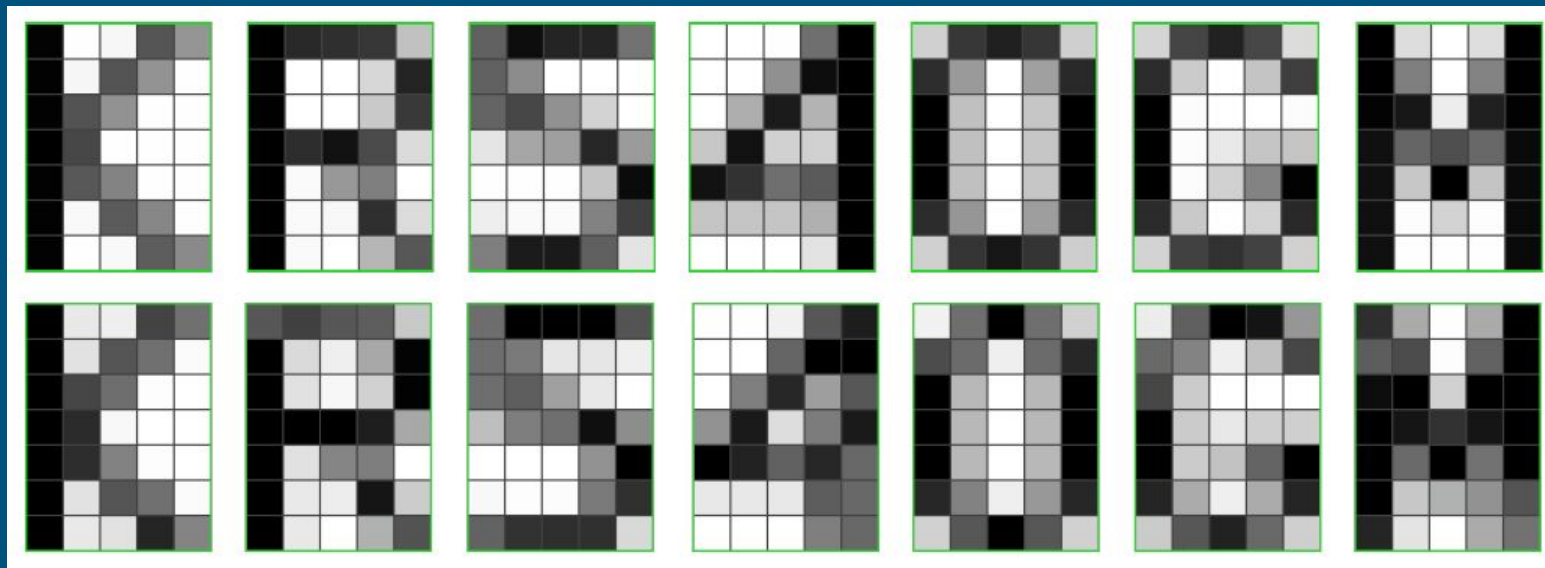


Figura 13: Vetor de amostras de placas de carro e caracteres reconhecidos. Fonte: NOCUN et al., 2017.

Algoritmo de Reconhecimento de Placas

- 5º passo: Interpretação dos resultados.
 - É possível calcular um nível de similaridade com números de placas que estão sendo pesquisados.
 - A solução proposta é rodar uma palavra sobre outra, calculando cada vez a soma dos níveis de aptidão.
 - O melhor resultado possível é então retornado, levando em conta o fator de penalidade para os caracteres que não foram encontrados.

Algoritmo de Reconhecimento de Placas

- Testes em um laptop mostrou que o processamento de imagens de cerca de 1M pixel levou 100-250ms dependendo do número de placas na imagem.
- O desempenho de um dispositivo móvel equipado com uma CPU ARM Cortex A9 com FP foi pior; 2-4 fps foram processados .
- O desempenho, portanto, é suficiente para esta aplicação.

Integração e Testes

- A plataforma foi integrada com sucesso em dispositivos móveis.
- Os algoritmos de localização de placas de carros eram executados de forma autônoma.
 - Hardware: PandaBoard ES2, baseado na arquitetura ARM.
 - Imagens eram obtidas via USB externo.
 - Sistema Operacional: ArchLinux.

Integração e Testes

- Havia três segmentos de operação principais: heartbeat, gerenciamento de tarefas e processamento de imagem.
- Após a inicialização, o programa foi controlado apenas através da página da web, respondendo às tarefas do usuário e enviando dados com os resultados adequados para o banco de dados da plataforma.

Integração e Testes

- O sensor da placa do carro não usou instância local do CouchDB.
- Dados eram enviados diretamente para o banco de dados principal usando solicitações HTTP remotas.
- A aplicação era responsável por manter seus dados de configuração (lista de placas de carro).
- O usuário poderia manipular os dados usando as tarefas adequadas no sistema de gerenciamento da plataforma on-line.
- A lista de placas foi persistente em toda a vida do sistema.

Integração e Testes

- Dos experimentos:
 - Foram realizados duas vezes no mesmo local:
 - No estacionamento da Universidade.
 - Ambos envolveram movimentações de alguns minutos com uma câmera instalada em um carro.

Integração e Testes

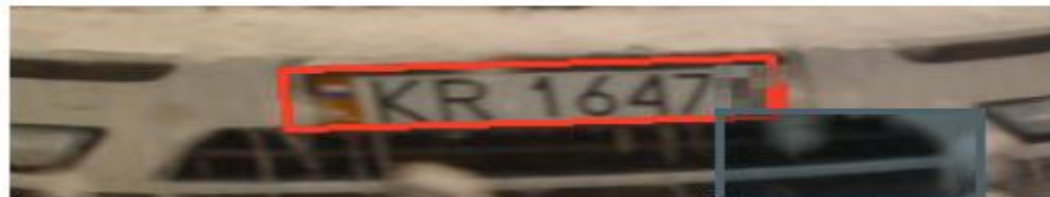
- No experimento realizado no primeiro dia:
 - Condições climáticas ideais para o uso do algoritmo de reconhecimento.
 - Dia de sol e as placas dos carros não estavam sujas.

Integração e Testes

- No experimento realizado no segundo dia:
 - Condições climáticas completamente diferentes.
 - Neve pesada, sujeira nos carros e gotículas de água no parabrisa do carro utilizado no experimento prejudicou a performance do reconhecimento.

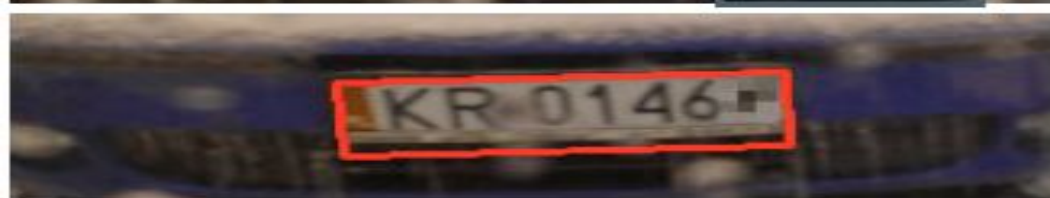
Integração e Testes

- Em ambos experimentos, carros foram escolhidos com antecedência e o algoritmo foi configurado para procurá-los.
- Sob condições ideais, o algoritmo obteve uma performance de 92%.
- Sob condições impróprias, entre 25% e 30% das placas foram reconhecidas.



KR1647?

81%



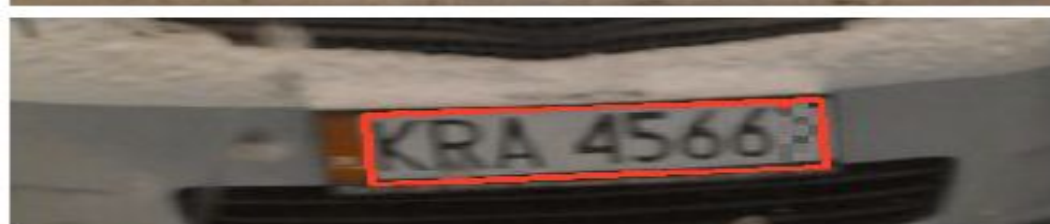
KR0146?

81%



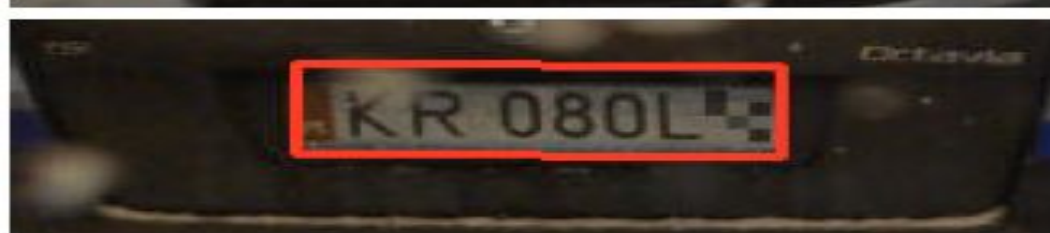
KR609G?

84%



2A48

—



KR380C?

79%

Integração e Testes


- A Quarta placa não foi reconhecida devido a imagem fonte estar embaçada.
- A última placa foi classificada incorretamente.
- A qualidade global de reconhecimento e correspondência foi satisfatória, considerando-se as condições climáticas.

Considerações Finais


- O couchDB se mostrou ótimo para uso de redes de sensores em larga-escala processando ativamente em dispositivos móveis.
- O algoritmo utilizado não é dos melhores no domínio, havendo espaço para melhorias.
- O algoritmo de processamento de imagem funciona de forma confiável em dispositivos móveis, proporcionando um desempenho suficiente.

Referências Bibliográficas

- NOCUN, Ł; NIE, C. M; PIKULA, P; MAMLA, A. TUREK, W. (2013). Car-finding system with couchdb-based sensor management platform. *AGH University of Science and Technology Department of Computer Science*, 2013. 14(3):403–422. Polônia.



Car-Finding System with CouchDB-Based Sensor Management Platform



Banco de dados NoSQL - CouchDB



Luciano Brum
Rebeca Fiss