A Contextual Data Offloading Service With Privacy Support

Francisco Gomes
almada@crateus.ufc.br
Universidade Federal do Ceará
Fortaleza - CE - Brasil

Lincoln Rocha
lincoln@great.ufc.br
Universidade Federal do Ceará
Fortaleza - CE - Brasil

Fernando Trinta
fernandotrinta@great.ufc.br
Universidade Federal do Ceará
Fortaleza - CE - Brasil

ABSTRACT

Mobile and context-aware applications are now a reality thanks to the increased capabilities of mobile devices. In the last twenty years, researchers had proposed several software infrastructures to help the development of context-aware applications. We verified that most of them do not store contextual data history and that few of these infrastructures take into account the privacy of contextual data. This article presents a service named COP (Contextual data Offloading service with Privacy support) to mitigate these problems. Its foundations are: (i) a context model; (ii) a privacy policies; and (iii) a list of synchronization policies. The COP aims at storing and processing the contextual data generated from several mobile devices, using the computational power of the cloud. We have implemented one experiment evaluated the impact of contextual filter processing in the mobile device and the remote environment. In this experiment, we measured the processing time and the energy consumption of COP approach. The analysis detected that the migration of data from mobile device to a remote environment is advantageous.

KEYWORDS

Mobile Cloud Computing, Context-Aware, Middleware, Mobile Device, Offloading, Privacy

1 INTRODUÇÃO

Este artigo apresenta um resumo dos resultados obtidos na dissertação defendida no Programa de Pós-graduação em Ciências da Computação em 08 de fevereiro de 2017. O trabalho foi desenvolvido pelo 1o autor, num período de 23 meses, sob a orientação do(s) último(s) autor(es).

Nas últimas décadas, os dispositivos móveis tornaram-se ferramentas essenciais para a sociedade moderna. *Smartphones, tablets e smartwatches* estão agora difundidos em todo o mundo [2]. Esses dispositivos têm vários sensores (por exemplo, acelerômetro, giroscópio e GPS) que permitem reunir informações sobre o ambiente ao redor, bem como dados de seus usuários [7]. Esse tipo de sistema interpreta e processa dados contextuais para caracterizar a situação do usuário (por exemplo, localização, humor). Os aplicativos sensíveis ao contexto monitoram as mudanças de contexto para se adaptarem a essas novas condições e, consequentemente, melhoram a experiência do usuário. O conhecimento do contexto é uma característica essencial no cenário de Computação Ubíqua preconizado por Mark Weiser.

Um dos maiores desafios para a adoção da Computação Ubíqua está relacionado à infraestrutura necessária para suportar esses aplicativos [1]. Apesar dos avanços tecnológicos, os dispositivos móveis têm recursos restritos em comparação com as máquinas tradicionais de servidores e *desktops*. Os dispositivos móveis têm poder de computação restrito devido ao tamanho reduzido, custos mais baixos e eficiência de energia. Uma solução possível para superar esse problema é o uso de serviços em nuvem em um paradigma chamado *Mobile Cloud Computing* (MCC) [4, 9]. Resumidamente, a MCC concentra-se no *offloading* de tarefas de processamento ou armazenamento de dados do dispositivo móvel para um ambiente remoto com maior capacidade computacional. De acordo com a literatura científica, o termo " offloading " refere-se à técnica usada para migrar armazenamento e processamento de um dispositivo fraco para um mais poderoso [4, 9]. Embora a maioria das infraestruturas sensíveis ao contexto se concentre no gerenciamento de contexto do usuário e na transparência da complexidade, há poucos trabalhos que abordam o *offloading* de dados e a integração com técnicas de MCC.

A ideia de superar problemas de aplicação sensíveis ao contexto com técnicas de *offloading* é bastante promissora, mas também apresenta muitos desafios. Por exemplo, o *offloading* de dados deve levar em conta possíveis dados confidenciais do usuário (por exemplo, localização), que os usuários nem sempre se sentem à vontade para compartilhá-los com outros usuários. Portanto, uma abordagem de *offloading* para aplicativos com reconhecimento de contexto deve abordar questões como: Quais dados devem ser anonimizados? Quem pode acessar os dados do usuário? Esses tipos de problemas, quando negligenciados, podem levar à divulgação pública dos dados contextuais do usuário sem o devido consentimento, abrindo caminho para disputas legais dispendiosas. Para resolver essas questões, propusemos um serviço de descarregamento de dados com suporte à privacidade chamado COP (*Contextual data Offloading service with Privacy support*) [5].

O COP é baseado em duas soluções: *(i)* LoCCAM [6], um *middleware* que ajuda os desenvolvedores de aplicativos sensíveis ao contexto a obterem informações dos sensores, bem como separar questões relacionadas à aquisição de informações contextuais a partir da lógica de negócios das aplicações; e *(ii)* MpOS [3], uma arquitetura orientada a serviços que permite aos desenvolvedores marcar métodos em seus aplicativos usando anotações, a fim de definir quais métodos devem ser transferidos para servidores em nuvem em vez de ser executado no dispositivo móvel. O COP visa (*i*) disseminar dados contextuais entre o dispositivo móvel e o ambiente de nuvem de uma maneira transparente e configurável; e (*ii*)

In: I Concurso de Teses e Dissertações (CTD 2019), Rio de Janeiro, Brasil. Anais Estendidos do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia). Porto Alegre: Sociedade Brasileira de Computação, 2019.
ISSN 2596-1683

fornecer uma política de privacidade ajustável, com a qual os usuários podem restringir a disseminação e difusão de suas informações contextuais.

O restante deste artigo está organizado da seguinte forma. Na Seção 2, apresentamos o COP, bem como o modelo de contexto, políticas de privacidade e políticas de sincronização propostas. A Seção 3 apresenta uma prova de conceito que desenvolvemos mostrando o uso do serviço COP. Na Seção 4, descrevemos experimentos de avaliação que mediram o desempenho do COP e seu consumo de energia. Finalmente, a última seção conclui o artigo.

2 COP

Buscando mitigar os problemas apresentados anteriormente, propomos uma solução para permitir o descarregamento de dados contextuais. Esta solução é um serviço chamado COP, que é baseado em: (*i*) um modelo de contexto; (*ii*) uma política de privacidade; e (*iii*) políticas de sincronização.

2.1 Arquitetura

A solução proposta tem uma arquitetura de três camadas, mostrada na Figura 1. O primeiro nível é representado pelo dispositivo móvel, onde os dados contextuais são capturados de seus sensores e armazenados localmente. A segunda camada é representada por um *cloudlet* [8], que recebe dados contextuais de dispositivos móveis conectados à mesma WLAN. O último nível representa um espaço de tupla hospedado em um serviço de nuvem, que agrega dados de diferentes *cloudlets* e usuários que compartilham seus dados contextuais. No lado do dispositivo móvel, foi adicionado um banco de dados NoSQL, que atua como um cache local de informações de contexto, até que ocorra o processo de *offloading* de dados.

No lado móvel, o componente *Synchronizer* é responsável por recuperar dados contextuais do banco de dados em execução no dispositivo móvel e enviá-lo para o ambiente remoto. *Synchronizer* usa diferentes estratégias definidas pelas políticas de sincronização (para mais detalhes, veja Seção 2.4). Sempre que as tuplas são enviadas para o ambiente remoto, o banco de dados local é esvaziado para evitar o estouro de dados no dispositivo móvel. A camada de nuvem e de *cloudlet* fornece componentes responsáveis pelo gerenciamento de dados contextuais compartilhados. No caso do *cloudlet*, esse componente é chamado *Cloud SysSU*, enquanto o serviço que agrega dados de todos os *cloudlets* e usuários é chamado *Global SysSU*. Ambos possuem um banco de dados NoSQL, semelhante ao que é executado no dispositivo móvel, que visa persistir dados contextuais de dispositivos móveis que usam o sistema.

O COP permite que dispositivos móveis descarreguem tuplas em *cloudlets*, ou no *Global SysSU*, permitindo que dados contextuais de múltiplos dispositivos móveis sejam reunidos em um só lugar. No lado do *cloudlet*, o componente *Synchronizer* envia seus dados periodicamente para o *Global SysSU*. É importante observar que, se nenhum *cloudlet* for encontrado, os dados contextuais poderão ser enviados diretamente do dispositivo móvel para *Global SysSU*. O COP acessa os dados contextuais através de filtros contextuais de forma transparente com uma API. A API é usada para integrar o COP com o aplicativo. Ele é responsável, entre outros recursos, por direcionar o filtro para o dispositivo móvel (*Local SysSU*), no cloudlet (*Cloud SysSU*) ou na nuvem (*Global SysSU*). Para realizar essa tarefa, o COP usa o suporte de *offloading* do MpOS [3]. Entre outros recursos, o MpOS decide se o processamento de *offloading* deve ser executado ou não no ambiente remoto para melhorar o desempenho do aplicativo. As métricas de rede são usadas para a tomada de decisões, especificamente em relação à qualidade da conexão entre dispositivos móveis e o servidor remoto (por exemplo, latência de conexão).

O COP realiza o mecanismo de descoberta do *cloudlet*, que realiza um *broadcast* na rede para a localização de um *cloudlet* ativo. Se essa decisão de realizar o offloading indicar que o melhor local é o dispositivo móvel, o filtro contextual será executado no SysSU local. Se valer a pena executar esses filtros em um *cloudlet* ou em uma nuvem, esses filtros são descarregados para esses locais. Quando esses filtros são executados (no REE), ele usa as informações de contexto agregadas e armazenadas em *Cloud SysSU* ou *Global SysSU*.

2.2 Modelo de Contexto

O COP estende o modelo de contexto do LoCCAM [6]. No COP, os dados contextuais de vários usuários são armazenados em um espaço de tupla compartilhado. Assim, novas informações são necessárias para individualizar os dados, o que indica que as tuplas COP exigem mais campos para garantir a identificação da fonte de dados do contexto, lidar com problemas de relógio em um ambiente distribuído e com privacidade de dados, de acordo com a definição 1.

Definição 1. *Uma tupla COP é representada por nove elementos, sendo cinco herdados do LoCCAM e quatro novos campos destacados a seguir:* t = (`iddevice`, "?"), (`idapp`, "?"), (`visible`, "?"), (`globaltimestamp`, "?")⟩*, onde* `iddevice` *representa o ID do dispositivo móvel;* `idapp` *representa o id do aplicativo que está usando o serviço;* `visible` *é usado para identificar a visibilidade dos dados (mais detalhes na seção 2.3); e* `globaltimestamp` *representa o instante, em milissegundos, no qual os dados contextuais são inseridos no ambiente remoto.*

A estrutura apresentada em (1) caracteriza uma amostra dos dados de contexto "temperatura ambiente"no formato da tupla COP. Os campos sublinhados foram adicionados aos campos existentes. Neste exemplo, o dispositivo móvel possui o ID '`0424033418` " e usa um aplicativo com identificação' '`br.casa.temp` ".

$$
\begin{aligned}
t = \langle & (\texttt{contextkey}, \text{“}\texttt{context.ambient.temperature}\text{”}), \\
& (\texttt{source}, \text{“}\texttt{physical}\text{”}), \\
& (\texttt{values}, \text{“}\texttt{80}\text{”}), \\
& (\texttt{accuracy}, \text{“}\texttt{0}\text{”}), \\
& (\texttt{timestamp}, \text{“}\texttt{239459060969}\text{”}) \\
& (\underline{\texttt{iddevice}}, \text{“}\texttt{0424033418}\text{”}) \\
& (\underline{\texttt{idapp}}, \text{“}\texttt{br.casa.temp}\text{”}) \\
& (\underline{\texttt{visible}}, \text{“}\texttt{0}\text{”}) \\
& (\underline{\texttt{globaltimestamp}}, \text{“}\texttt{239459780932}\text{”})\rangle
\end{aligned}
\tag{1}
$$

2.3 Políticas de Privacidade

Para lidar com a privacidade, no COP foi estabelecido dois conceitos específicos: dados contextuais sensíveis e *cloudlet* confiável. No

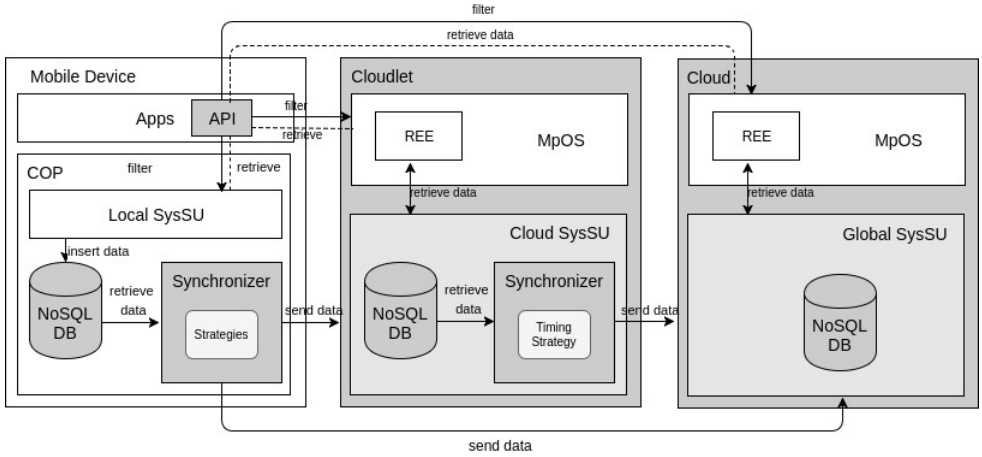


Figura 1: Arquitetura do COP

COP, os usuários podem declarar explicitamente quais dados eles autorizam (ou não) a enviar para um ambiente remoto. A privacidade de dados contextuais está relacionada à sua visibilidade, que é representada pelo campo "visible" da estrutura da tupla COP. Se um usuário não tiver nenhum problema em divulgar seus dados com outros usuários, o valor será "1" para indicar que a visibilidade é pública, caso contrário, o valor é "0" para indicar privado. Em (1), o dado é considerado sensível e não será enviado para o ambiente remoto.

Cloudlet confiável é um *cloudlet* definido pelo usuário que pode receber todos os dados contextuais coletados, inclusive os sensíveis. Pode ser, por exemplo, um *desktop* na casa desse usuário. No entanto, os dados privados armazenados em *cloudlets* confiáveis não são encaminhados para *Global SysSU*. Assim, de acordo com a política de privacidade do COP, se um dispositivo móvel não estiver conectado a um *cloudlet* confiável quando ocorrer a migração contextual de dados, apenas as tuplas públicas serão migradas. Caso contrário, se um dispositivo estiver conectado a um *cloudlet* confiável, todos os dados contextuais serão enviados.

2.4 Políticas de Sincronização

Outro ponto importante é definir quando os dados contextuais devem ser migrados de dispositivos móveis para um ambiente remoto. Estabelecemos diferentes estratégias para essa proposta, que chamamos de políticas de sincronização.

Três estratégias de envio foram especificadas, a saber: (*i*) tempo, (*ii*) quantidade de tuplas e (*iii*) orientado para conexão *Wi-Fi*. A estratégia de tempo significa que, periodicamente, as tuplas são enviadas para o ambiente remoto. O período para o envio dos dados é configurável. O valor padrão é 30 segundos e é empírico. A estratégia quantidade de tuplas implica que, se um número pré-estabelecido de tuplas for atingido, essas tuplas serão enviadas para a nuvem. Este número também é configurável. Por fim, a estratégia orientada à conexão *Wi-Fi* define que somente quando uma conexão *Wi-Fi* é estabelecida, as tuplas do dispositivo móvel serão enviadas para a infraestrutura da nuvem. Essas estratégias podem ser combinadas simultaneamente, onde aquela que acontece primeiro será executada.

3 AVALIAÇÃO

Esta Seção apresenta experimentos realizados para avaliar o desempenho de aplicativos móveis e o consumo de energia de dispositivos móveis ao usar o COP. Será apresentado um dos aplicativos dos nossos experimentos chamado de *Integers*.

O aplicativo *Integers* foi projetado para fornecer dados contextuais contendo valores inteiros. Para esta aplicação, foi desenvolvido um sensor lógico, que recebe dois limiares, um inferior e um superior, e gera tuplas de dados contextuais dentro desse intervalo. Por exemplo, se o sensor receber o valor 0 e 100, ele gerará as tuplas contendo os valores inteiros do intervalo (0, 1, 2, ..., 100). No final, o COP, mais especificamente o SysSU, terá 101 tuplas de dados contextuais contendo valores inteiros. O usuário pode controlar um número de tuplas geradas pelo COP, bem como controlar o número de tuplas que serão recuperadas dos filtros contextuais. A partir da execução dos filtros contextuais, foram medidos o tempo de processamento e o consumo de energia. As subseções a seguir apresentam mais detalhes sobre os experimentos.

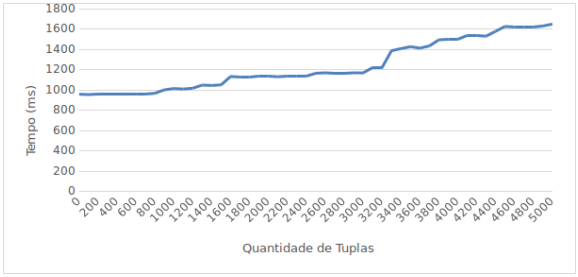
O experimento foi realizado em duas etapas. A primeira etapa consistiu em medir o tempo e o consumo de energia gastos no processamento de filtros contextuais no dispositivo móvel. A segunda etapa executou o mesmo experimento utilizando o dispositivo e o *cloudlet*. O dispositivo móvel usado foi um *smartphone* LG G3 Beat com Qualcomm Snapdragon 400 MSM8226 Cortex-A7 de 1,2 GHz Quad Core, memória interna de 8GB e 1 GB de RAM, rodando o Android 5.0.2. O *cloudlet* era um *laptop* rodando o sistema operacional Linux Mint 17.2 de 64 bits, com 8 GB de RAM e processador Core i5-4200U (1.6 GHz Quad Core). Para calcular o tempo de execução dos filtros contextuais, medimos o tempo gasto desde a chamada do método responsável pela recuperação das tuplas, até o retorno de sua execução. Esse tempo decorrido foi calculado em milissegundos. Um dispositivo especializado chamado *Power Monitor*[1] foi usado para medir o consumo de energia durante esses experimentos. Uma das métricas analisadas pelo *Power Monitor* é a energia (em Watts). O consumo de energia foi calculado multiplicando a potência e o

[1] https://www.msoon.com/LabEquipment/PowerMonitor/

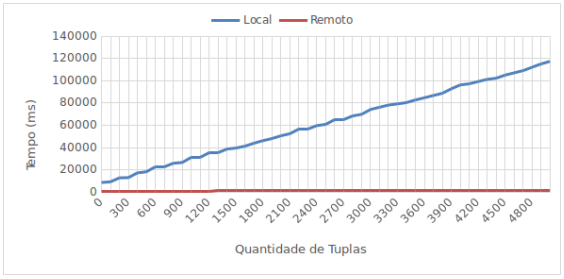
tempo de processamento, o que equivale à energia em milijoules (mJ).

Foi realizado a criação de 10000 tuplas na faixa de 1 a 10000. Para a execução de filtros contextuais, a filtragem foi realizada em passos de de 100, do limite inferior 1 até o limite superior de 5000. Assim, os filtros foram os seguintes: 1 a 100, 1 a 200, 1 a 300, ..., 1 a 5000. Para cada filtro, o tempo de processamento e o consumo de energia foram medidos. Os experimentos foram realizados 30 vezes e os dados coletados foram armazenados em uma planilha para posterior análise.

Figura 2 (a) mostra um gráfico com o resultado do processamento do filtro contextual no *cloudlet*. O gráfico mostra o tempo de processamento em milissegundos por um número de tuplas recuperadas. A partir deste gráfico, é possível verificar que o tempo de processamento tem uma pequena variação com o aumento da quantidade de tuplas recuperadas. Assim, pode-se afirmar que a qualidade da conexão tem maior influência no serviço do que o número de tuplas processadas. A figura 2 (b) mostra um gráfico comparativo do tempo de processamento do filtro contextual no dispositivo móvel e no *cloudlet*. A partir deste gráfico, é possível ver que o tempo de processamento do dispositivo móvel é maior que o *cloudlet*.



(a) Cloudlet



(b) Dispositivo Móvel x Cloudlet

Figura 2: Desempenho de processamento de filtro contextual

Com relação ao consumo de energia, a Figura 3 (b) mostra um gráfico comparativo do consumo de energia do processamento de filtro contextual no dispositivo móvel e no *cloudlet*. A partir deste gráfico, é possível verificar que o custo energético do processamento de filtro contextual no dispositivo móvel é bastante superior ao custo do *cloudlet*. Assim, pode-se confirmar que a transferência de dados de dispositivos móveis para um ambiente remoto tem vantagens em relação ao armazenamento no dispositivo. Também oferece maior velocidade e menor consumo de energia ao recuperar informações de todo o sistema. O consumo energético depende do tempo, assim este gráfico tem o mesmo comportamento que o gráfico da Figura 2 (b).

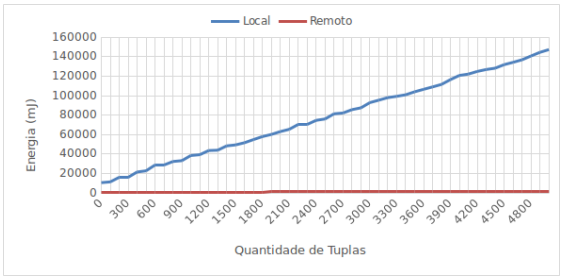


Figura 3: Consumo energético do processamento de filtro contextual

4 CONSIDERAÇÕES FINAIS

Este artigo apresenta o COP, uma proposta de um serviço de migração de dados contextual de dispositivos móveis para uma nuvem, que lida com políticas de privacidade de dados e sensibilidade contextual aos dados. Nossa pesquisa implementa técnicas de descarregamento de MCC usando um ambiente cloudlet. Reduzimos os desafios de privacidade, modificando o modelo de contexto do LoCCAM e estabelecendo uma política de privacidade, que trata a visibilidade de dados contextuais e sua migração para um cloudlet confiável.

REFERÊNCIAS

[1] R. Caceres and A. Friday. 2012. Ubicomp Systems at 20: Progress, Opportunities, and Challenges. *IEEE Pervasive Computing* 11, 1 (January 2012), 14–21. https://doi.org/10.1109/MPRV.2011.85

[2] Cisco Visual Networking Index Cisco. 2017. Global mobile data traffic forecast update, 2016–2021. *white paper* (2017).

[3] Philipp B. Costa, Paulo A. L. Rego, Lincoln S. Rocha, Fernando A. M. Trinta, and Jose N. de Souza. 2015. MpOS: A Multiplatform Offloading System. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC '15)*. ACM, New York, NY, USA, 577–584. https://doi.org/10.1145/2695664.2695945

[4] Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu. 2013. Mobile Cloud Computing. *Future Gener. Comput. Syst.* 29, 1 (Jan. 2013), 84–106. https://doi.org/10.1016/j.future.2012.05.023

[5] Francisco A.A. Gomes, Windson Viana, Lincoln S. Rocha, and Fernando Trinta. 2016. A Contextual Data Offloading Service With Privacy Support. In *Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web (Webmedia '16)*. ACM, New York, NY, USA, 23–30. https://doi.org/10.1145/2976796.2976860 doi: 10.1145/2976796.2976860.

[6] Marcio E. F. Maia, Andre Fonteles, Benedito Neto, Romulo Gadelha, Windson Viana, and Rossana M. C. Andrade. 2013. LOCCAM - Loosely Coupled Context Acquisition Middleware. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC '13)*. ACM, New York, NY, USA, 534–541. https://doi.org/10.1145/2480362.2480465

[7] Davy Preuveneers and Yolande Berbers. 2007. Towards Context-aware and Resource-driven Self-adaptation for Mobile Handheld Applications. In *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC '07)*. ACM, New York, NY, USA, 1165–1170. https://doi.org/10.1145/1244002.1244255

[8] Mahadev Satyanarayanan, P. Bahl, R Caceres, and N. Davies. 2009. The Case for VM-Based Cloudlets in Mobile Computing. *Pervasive Computing, IEEE* 8, 4 (Oct 2009), 14–23. https://doi.org/10.1109/MPRV.2009.82

[9] M. Shiraz, A. Gani, R.H. Khokhar, and R. Buyya. 2013. A Review on Distributed Application Processing Frameworks in Smart Mobile Devices for Mobile Cloud Computing. *Communications Surveys Tutorials, IEEE* 15, 3 (Third 2013). https://doi.org/10.1109/SURV.2012.111412.00045