DESENVOLVIMENTO DE UM SISTEMA DE SUPERVISÃO E CONTROLE RESIDENCIAL UTILIZANDO PROCESSAMENTO DE IMAGENS

Gessé Marchioro[1], Ricardo Antonello[2]
[1]Instituto Federal Catarinense, Campus Luzerna / gesse.marchioro@gmail.com
[2]Instituto Federal Catarinense, Campus Luzerna / ricardo.antonello@ifc.edu.br

Resumo: ***Este artigo teve origem no projeto de pesquisa intitulado "Desenvolvimento de um sistema de supervisão e controle residencial utilizando processamento de imagens" do Instituto Federal Catarinense – IFC, campus Luzerna. São tratados conceitos sobre automação residencial ou domótica, tais como visão computacional e processamento de imagem. Estas técnicas permitem captar os movimentos do usuário para gerar a interação com o ambiente como acender lâmpadas, abrir cortinas, ligar a televisão, cafeteira, entre outros. Essas funções são realizadas através de gestos feitos com a mão. O hardware utilizado para a captura de imagem é uma câmera RGB simples, de baixo custo. Os resultados demonstram que apesar de possível, não é viável utilizar câmeras de baixo custo sem um sensor de profundidade acoplado para a captura de movimento. Contudo, mesmo com hardware barato é possível capturar os gestos caso o usuário esteja próximo da câmera.***

Palavras-Chave: Domótica, Automação Residencial, Visão Computacional.

1. INTRODUÇÃO

Com o desenvolvimento da área da robótica e automação, diversos setores como indústrias, transportes e comunicações vêm sendo impactados. Além disso, nas residências já existe um alto grau tecnológico presente no desenvolvimento de eletrodomésticos, porém uma área nova está iniciando, a automação residencial também conhecida como domótica (ADAMI, 2013).

A domótica, associação das palavras "*domus*" (domicílio) e "robótica" (GIARETTA, 2011), é um conceito que denota a automação de processos em uma residência, ou seja, visa tornar automáticos processos dentro da residência do usuário. Apesar de algumas pequenas atividades já parecerem suficientemente simples, para pessoas com algum tipo de limitação física ainda existem dificuldades. Além disso, a automação de processos gera um maior conforto para o usuário. De acordo com Adami (2013), "esta tecnologia pode ser atribuída a questões e itens de segurança, comunicação, energia ou conveniência e conforto".

Devido às pesquisas nesta área ainda serem recentes e muitos conceitos ainda estarem no "estado da arte" da tecnologia, existem muitos recursos e ideias a serem explorados, como o uso da visão computacional para auxiliar nos comandos a serem acionados, como afirma Xavier (2014). Neste contexto, este artigo aborda o estudo para realizar o controle residencial comandado por gestos visando o conforto dos usuários e uma maior acessibilidade.

2. REFERENCIAL TEÓRICO

2.1. Python

Para visão computacional a linguagem Python se destaca pela simplicidade (PYTHON, 2017).

Como define o site PyScience (2017), “Python é uma linguagem de programação criada por Guido van Rossum em 1991. Os objetivos do projeto da linguagem eram: produtividade e legibilidade. Em outras palavras, Python é uma linguagem que foi criada para produzir código bom e fácil de manter de maneira rápida”.

2.2. OpenCV

Para realizar operações com imagens, são necessárias bibliotecas que são importadas para o programa e assim é possível utilizar funções já criadas para manipular as imagens. Existe uma vasta lista dessas bibliotecas, porém a mais utilizada é o OpenCV - *Open Source Computer Vision Library* (2017). Cavalca (2015) define: “A biblioteca OpenCV possui mais de 500 funções. Foi idealizada com o objetivo de tornar a visão computacional acessível a usuários e programadores em áreas tais como a interação humano-computador em tempo real e a robótica”.

2.3. Haar cascades

Na visão computacional, muitas vezes deseja-se encontrar algum objeto específico na imagem. Para os humanos tal ação é simples, pois já se sabe as características de tal objeto, e seguindo essa ideia, Viola e Jones (2001) criaram o “*haar-like cascades features*”, baseado em características chamadas *haar*.

Essas características são consideradas analisando retângulos em regiões da imagem, em que se calcula a diferença entre as somas da intensidade dos pixels. Por exemplo, em um rosto humano a região dos olhos é mais escura que a região das bochechas, e isso é uma característica que pode ser entendida e “ensinada” ao classificador.

Figura 1 – Característica referente à intensidade dos pixels nas regiões das bochechas e dos olhos
Fonte: VIOLA-JONES (2001)

Para que o detector seja forte (realize as detecções precisamente) é necessário que sejam analisadas várias características, e que essa análise feita com várias imagens.

Esses classificadores são chamados *haar-cascades*, e são salvos em um arquivo XML. A biblioteca OpenCV tem funções prontas para realizar a procura de objetos, mas é necessário indicar o classificador. Apesar de existirem muitos *haar-cascades* prontos disponíveis, para aplicações mais específicas é necessário treinar o próprio.

2.4. Raspberry Pi

Segundo a Fundação Raspberry Pi (2017), a placa é um minicomputador do tamanho de um cartão de crédito, de baixo custo, e que desempenha grande parte das funções de um computador grande. Com a primeira placa lançada em 2012 no Reino Unido, atualmente existem disponíveis diversos modelos, sendo o Raspberry Pi 3 Modelo B o com maior número de recursos.

Dentre as vantagens de tal hardware em relação a usar um computador ou outras placas semelhantes, pode-se citar: grande número de GPIO (*General Purpose Input/Output*, usadas para realizar acionamentos), conectividade Wi-Fi e Bluetooth, processamento rápido, diversos sistemas operacionais compatíveis, possibilidade de conectar câmera, e vasto conteúdo e suporte disponível na internet referente ao seu uso.

Figura 2 – Versões da placa disponíveis até 2017
Fonte: Fundação Raspberry Pi (2017)

3. METODOLOGIA

Após o estudo da bibliografia relacionada à área para compreender conceitos básicos, foi possível realizar dezenas de pequenos programas relacionados ao projeto, como contador de dedos (pela quantidade de contornos em determinadas regiões), câmera com base móvel que busca centralizar um objeto (mira automática), interface controlada pelo posicionamento da mão, jogos, leitor das cores de um cubo mágico, entre outros. Esses pequenos projetos serviram como treinamento para desenvolver a prática da programação. Também foram feitos vídeos e apresentações explicando os objetivos do projeto (YOUTUBE, 2017).

Em certo momento, foi estudado o método de detecção de objetos por *haar-cascade*, e tentou-se desenvolver o próprio detector. Foram tiradas centenas de fotos dos objetos a serem detectados e executado um programa para gerar o arquivo XML. Tal programa necessitava de muita memória, funcionando apenas no computador de um bolsista, que teve que deixá-lo em funcionamento durante algumas madrugadas, e após semanas do programa em execução, não se obteve um resultado funcional, fazendo com que fosse necessário voltar a usar detectores prontos disponíveis nas bibliotecas.

Após a falha ao criar o detector, foi dada atenção para métodos mais comuns para realizar detecção da mão.

Ao assistir vídeos de projetos que obtiveram sucesso e analisar o código de Gazman (2014), foi percebido que o método para realizar a contagem de dedos seguia os passos:

1. Borrar levemente a imagem para eliminar ruídos;
2. Encontrar onde existe pele na imagem capturada pela câmera, comparando a dois tons de cor, e binarizá-la (deixar nas cores totalmente preto ou branco);

Figura 3 – Imagem da mão binarizada
Fonte: O autor (2017)

3. Contornar as supostas regiões onde existe pele, e considerar apenas a maior área (para isso, a área da mão na imagem deve ser maior que as demais partes do corpo detectadas);
4. Gerar um invólucro (*hull*) do contorno, formado por linhas retas que ligam as pontas do contorno. No caso da detecção da mão, a ponta de cada dedo é um vértice;

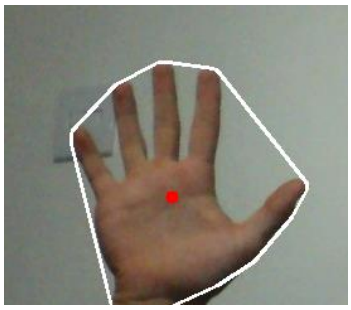

Figura 4 – Imagem da mão com invólucro
Fonte: O autor (2017)

5. Encontrar os pontos côncavos e convexos do contorno, e considerar apenas os com maior distância do centro da mão, que seriam supostamente as pontas dos dedos.

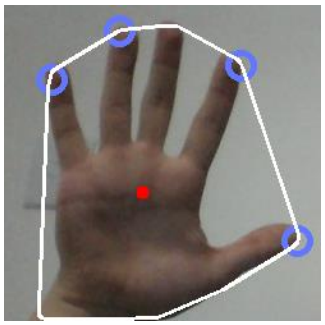

Figura 5 – Imagem da mão com invólucro e ponta dos dedos
Fonte: O autor (2017)

Feito isso, foram pensadas em três formas para determinar o número de dedos levantados, como a contagem de pontos suficientemente distantes do centro da mão, a análise da área capturada pela mão, e a contagem de contornos que passam por certas faixas na imagem.

Figura 6 – Métodos para realizar a contagem de dedos
Fonte: O autor (2017)

Ambos os três métodos não ofereceram resultados satisfatórios por baixa confiabilidade, portanto foram tomados novos rumos para reconhecer gestos das mãos.

Assim, foi retornado ao treinamento de detectores *haar-cascade*, porém, dessa vez com um método diferente (baseado no passo a passo de Rezaei (2017), "*Creating a Cascade of Haar-Like Classifiers: Step by Step*"), e com o intuito de criar um detector para cada gesto. Foram tiradas centenas de fotos de cada gesto (mão fechada, formando a letra "L", apenas um dedo levantado, dois dedos, entre outros), cerca de 600 fotos de cada, todas em fundo branco. Posteriormente, foi preciso selecionar a área de cada imagem em que se encontrava o gesto, e então aguardado o processo de treinamento, que compara conjuntos de imagens para analisar quais as semelhanças. Em todos os classificadores houve um ponto em que a análise não prosseguia, e nesse momento foram realizados testes. Desta vez o classificador não apresentou erro ao ser executado, e foi notável a detecção do objeto desejado, porém era necessário um fundo branco e iluminação moderada. Nessas condições, o desempenho foi melhor até mesmo que de alguns classificadores padrão, tendo cerca de 80% de acerto em testes.

Figura 7 – Seleção da área do objeto para treinar o *haar-cascade*
Fonte: O autor (2017)

4. RESULTADOS E DISCUSSÃO

Como já dito, inicialmente foi tentado realizar o treinamento de um detector para placas, para testar a técnica. Tal processo demandava muito processamento do computador, que ao treinar, se tornava inutilizável para outras atividades, portanto, o programa era executado durante as madrugadas (sendo um pouco incomodo devido aos ruídos produzidos pelo cooler). Após semanas, ao testar o arquivo gerado, ocorreu erro na execução, e assim foi abandonado tal método.

Alternativamente foi pensado no método dos contornos, e suas três maneiras que poderiam funcionar, porém nenhuma teve resultado confiável. Seus resultados foram:

- Número de pontas de dedos – O programa era incapaz de reconhecer o dedo do meio caso todos os dedos fossem levantados, pois o ângulo nesse ponto ficava próximo à 180°, não formando um vértice. Em outros dedos eram detectados mais de um ponto, tornando imprecisa a contagem.
- Área do invólucro – Para a mão fechada, apenas um dedo, ou todos os dedos levantados, funcionava bem, pois a variação de área era grande. Porém, a área para um ou dois dedos erguidos é semelhante, e havia interferência do espaçamento dado entre os dedos.
- Número de contornos que passam por certa área – Foi (entre os três) o método que melhor funcionou. Eram contados todos os contornos que passavam por “tiras” da imagem, e posteriormente dividido por 2 (pois cada dedo possui dois contornos), porém era preciso centralizar a mão e deixar de um modo que os contornos passassem pelas tiras.

Para os três métodos, houveram problemas com a iluminação, pois era detectada a cor da pele, e dependendo da luminosidade ambiente, o tom variava muito e a binarização não ocorria como desejado. Para solucionar tal problema, poderia ser utilizada uma câmera que captura imagens em 3 dimensões, e assim possibilita determinar a profundidade dos objetos, portanto seria mais fácil para detectar a mão.

Quanto ao uso de um classificador *haar-cascade* para cada gesto, que ainda estão sendo feitos, as preocupações são quanto à possibilidade de confundir gestos (poderá ser prevenido criando gestos com grandes diferenças), porém esta foi a alternativa mais simples encontrada.

5. CONSIDERAÇÕES FINAIS

Portanto, pode ser percebido que apesar de os componentes terem baixo custo, os resultados não são totalmente utilizáveis para a criação de um produto comercial devido principalmente à

câmera, que tanto a específica para Raspberry quanto a webcam simples, são incapazes de determinar a profundidade da imagem e assim dificultam a detecção dos gestos. Entretanto, para algumas outras aplicações e projetos que não denotem detecção de objetos a longa distância, o hardware utilizado supre a necessidade, como no leitor das cores do cubo mágico por exemplo.

Como sugestão de continuidade ou aperfeiçoamento, podem ser realizados os mesmos experimentos com uma câmera que captura imagens em três dimensões, como as produzidas pela Intel® RealSense™ (REALSENSE, 2017), e assim detectar e delimitar os gestos com maior facilidade e precisão.

AGRADECIMENTOS

Agradecemos ao IFC – Instituto Federal Catarinense pelo apoio via edital de fomento 162/2016 para a realização deste projeto.

REFERÊNCIAS BIBLIOGRÁFICAS

ADAMI, Anna. **Domótica.** Disponível em <http://www.infoescola.com/tecnologia/domotica/>. Acesso em 11 de julho de 2017.

CAVALCA, Diego. **Uma breve introdução sobre visão computacional.** Disponível em <http://diegocavalca.com/uma-breve-introducao-sobre-visao-computacional/>. Acesso em 11 de julho de 2017.

GAZMAN, Sasha. **Real-Time Hand Gesture Detection.** Disponível em <https://github.com/sashagaz/Hand_Detection>. Acesso em 12 de julho de 2017.

GIARETTA, Rafael. **DOMÓTICA.** Disponível em <http://portalarquitetonico.com.br/domotica/>. Acesso em 11 de julho de 2017.

OPEN SOURCE COMPUTER VISION. **OpenCV**. Disponível em: <http://opencv.org/>. Acesso em: 13 jul. 2017.

PYIMAGESEARCH. **Accessing the Raspberry Pi Camera with OpenCV and Python**. Disponível em: <http://www.pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/>. Acesso em: 12 jul. 2017.

PYSCIENCE-BRASIL. Python: O que é? Por que usar? [2017]. Disponível em: < http://pyscience-brasil.wikidot.com/python:python-oq-e-pq>. Acesso em: 12 jul. 2017.

PYTHON SOFTWARE FOUNDATION. Python [2017]. Disponível em: <https://www.python.org/>. Acesso em: 12 jul. 2017.

RASPBERRY PI FOUNDATION. Raspberry Pi [2017]. Disponível em: <https://www.raspberrypi.org/>. Acesso em: 13 jul. 2017.

REALSENSE. Intel® RealSense™ Technology. Disponível em: <https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html>. Acesso em: 25 de jul. 2017

REZAEI, Mahdi. **Creating a Cascade of Haar-Like Classifiers: Step by Step.** Disponível em <https://www.cs.auckland.ac.nz/~m.rezaei/Tutorials/Creating_a_Cascade_of_Haar-Like_Classifiers_Step_by_Step.pdf>. Acesso em 13 de julho de 2017.

SANTOS, Túlio Ligneul. **Detecção de faces através do algoritmo de Viola-Jones.** Disponível em <http://www.lcg.ufrj.br/~marroquim/courses/cos756/trabalhos/2011/tulio-ligneul/tulio-ligneul-report.pdf>. Acesso em 13 de julho de 2017.

SHERRARD, Trevor. **Haar Cascade Classifier Training Tutorial.** Disponível em <http://www.trevorsherrard.com/Haar_training.html>. Acesso em 12 de julho de 2017.

SISLITE. **O que é a Domótica?**. Disponível em: <http://www.sislite.pt/domus.htm>. Acesso em: 11 jul. 2017.

YOUTUBE. **Exemplos de projetos: Seguindo Você!** Disponível em: <https://www.youtube.com/watch?v=uik4tiYI1vw>. Acesso em: 25 de jul. 2017.

XAVIER, Rafael. **Interação Natural na Domótica.** Disponível em <https://prezi.com/2kym0rjkmwlx/interacao-natural-na-domotica/>. Acesso em 11 de julho de 2017.