Comedouro eletrônico de dietas restritivas para animais com sobrepeso

Danie Auler
Universidade de Brasília
Faculdade Gama
Brasília, Brasil
danielauler7@gmail.com

Paulo Franklin Gomes
Universidade de Brasília
Faculdade Gama
Brasília, Brasil
pgomespereiradeoliveira@gmail.com

I. Justificativa

Em se tratando do século XXI, muitas pessoas que moram em apartamentos na zona urbana, possuem animais de estimação como gatos e cachorros. Muitos destes animais sofrem por ter sobrepeso, pois eles acabam se alimentando de maneira muito rápida, o que corrobora para o estresse desses animais, além do peso exacerbado[1].

A ideia de construir um comedouro eletrônico de dieta restritivas para animais com sobrepeso surgiu a partir de um problema vivenciado por um cachorro que mora perto da minha casa. Este possui um problema intestinal e, devido a isso, ele deve usufruir de uma dieta balanceada e cadenciada em horários previamente determinados.

Figura 1. Bichinho de estimação se alimentando rapidamente

Contudo, uma solução encontrada para contornar esse problema é projetar um sistema que é capaz, em horários previamente determinados, enviar uma foto da tijela do animal, via o aplicativo do telegram, que terá como função verificar o quanto de ração este comeu, bem como ter a funcionalidade de cancelar o bastecimento de ração na tijela do bichinho, caso esta esteja cheia no momento. Caso contrário, o sistema liberará aos poucos a quantidade de comida necessária para alimentar o animalzinho.

II. Objetivos

Criação de um comedouro eletrônico que na hora específica irá enviar para o usuário via telegram uma foto com a tigela de comida para que o mesmo saiba o quando da ração o animal comeu. Caso o usuário não cancele o acionamento da tigela, a mesma irá liberar aos poucos ou de uma vez (escolha do usuário) a quantidade de ração necessária nos horários especificados.

III. Requisitos

O comedouro será equipado com um raspberry pi zero conectado a internet e conectado a uma câmera apontada para a tigela do animal. Além disso, o comedouro contará com um sensor de presença ultrassônico para saber se o animal comeu a porção parcial. O rasp fará requisições http para o bot para avisar o usuário sobre as ações que aconteceram e estará com um server aberto para ouvir as requisições do bot quando o usuário quiser inserir um comando de intervenção ou ajuste nos horários de alimentação.

IV. Benefícios

O projeto tem como principal beneficiário o dono de animais com transtorno de ansiedade ou alguma doença que necessite de restrição alimentar com horários especificados. Em consequência disso, temos como beneficiário também pessoas que deixam o animal de estimação sozinho em casa e não querem deixar um pote muito grande com muita ração parada para que o animal consiga se alimentar ao longo do dia.

V. Servo motor

O servo motor é muito utilizado em projetos de automação industrial, onde de fato é necessário um controle de precisão maior em relação ao sistema que está sendo projetado. Outrora, imaginava-se que este só serviria para aplicações que envolvessem apenas projetos relacionados ao controle preciso da direção do torque, além da velocidade e da posição. Entretanto, este dispositivo vem sendo uma alternativa muito boa para substituir motores convencionais tais como, indução e atuadores pneumáticos e hidráulicos.

Todavia de fato esta é uma máquina, eletromecânica, que apresenta movimento proporcional a um comando, como dispositivo de malha fechada, ou seja: recebe um sinal de controle que verifica a posição atual para controlar o seu movimento indo para a posição desejada com velocidade monitorada externamente sob feedback de um dispositivo denominado taco ou sensor de efeito Hall ou encoder ou resolver, ou tachsin, dependendo do tipo de servomotor e aplicação.

Para o ponto de controle 4 foi aperfeiçoado o desenvolvido do código em C, que pode ser observado no apêndice A, com

o objetivo de controlar um micro servo motor da Tower Pro 9g SG90 com auxílio da Raspberry Pi 3.

Nesse código foi criado três funções, uma que faz o servo motor girar 90 graus e outra que faz girar -90 graus, além da função alimentar, que tem como objetivo liberar a ração que irá cair na tijela do animal.

Tal função alimentar é chamada na função principal, ou seja, a função main. Após ser chamada, a mesma tem como funcionalidade chamar a função girar motor, na qual ela fará o servo motor girar 90 graus de modo a liberar a ração durante um determinado tempo. Com o passar desse tempo, a função alimentar executar a função girar motor, por mais uma vez, porém no sentido de -90 graus com intuito de fechar a escotilha na qual cai a ração.

Pois posteriormente este será utilizado para abrir a escotilha pra cair a ração. Tal código movimenta o servomotor 5 vezes, usando o pino GPIO4, que é referenciado pela biblioteca wiringPi utilizando o pino 7.

Logo, abaixo é mostrado como foi feito a conexão entre os pinos do servo motor e da Raspberry Pi 3.

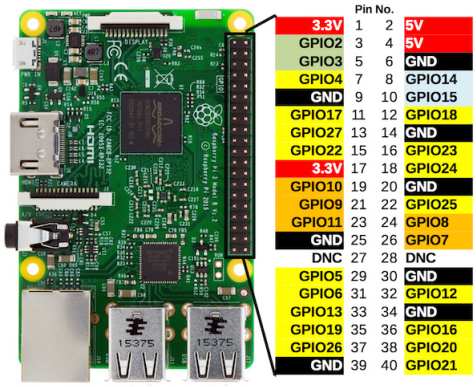

Figura 2. Pinagem da Raspberry Pi 3 com o servo motor

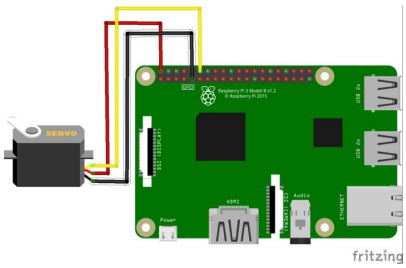

Figura 3. Conexão entre os pinos da Raspberry Pi 3 com o servo motor

VI. Bot do telegram

Com objetivo de conectar o usuário ao comedouro foi feito uma plataforma do telegram com sua API aberta e bem documentada, bem como um fluxograma demostrando como é feita essa conexão entre o usuário, o bot do telegram e o comedouro (figura 3). Para testarmos essa conexão foi realizado um teste com foco de enviar um comando via o bot do telegram, no qual este acende um led com uso da raspberry, além de verificar o recebimento desse comando. Tal código pode ser visualizado no apêndice B em anexo.

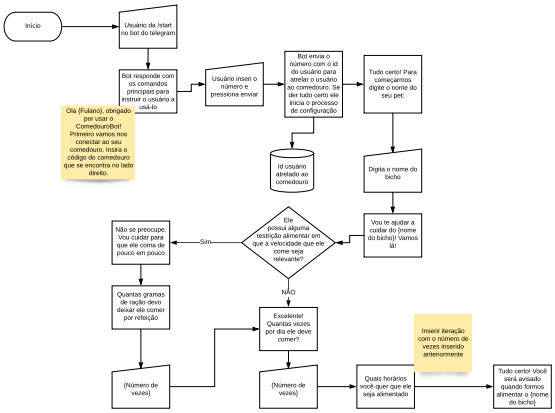

Figura 4. Conexão entre os pinos da Raspberry Pi 3 com o servo motor

VII. Servidor

Nesta etapa foi implementado um servidor em linguagem C. Porém neste código consta uma biblioteca denominada tgbot, que está implementada em C++. Esta biblioteca tem a incubencia de fazer a conexão com o telegram. Apesar dessa biblioteca ser na linguagem C++, é possível executá-la dentro do código em C. Por isso, não terá prolemas na hora da execução do projeto como um todo.

A função main foi composta pelo servidor do telegram que recebia e tratava todos os códigos recebidos. Assim que o usuário começa a usar, ele manda o comando "/start", que retorna uma mensagem informando o comando "/help"para mais informações. Com o help temos as três funções principais do sistema. Alimentar - função principal do sistema, consiste numa solicitação de alimentação, o sistema verifica se a tigela ainda está cheia, se sim ele retorna uma mensagem com uma foto da tigela. Se não, ele alimenta e retorna uma mensagem confirmando. Agendar - o sistema agenda para alimentar duas vezes por dia em horário pré definido. Alimentwr sem verificar - ele pula a verificação e alimenta direto.

VIII. Processamento de imagem

O processamento de imagem foi utilizado pensando em três etapas, sendo elas a segmentação da imagem, o limiar e o tratamento da imagem. O conceito de segmentação de imagem foi desenvolvido por meio do algoritmo K-Means, enquanto que o limiar foi implementado usando o algoritmo Hough Circle Transform e o tratamento da imagem foi realizada usando o conceito de histograma.

No que diz respeito ao algoritmo do K-Means é um processo iterativo que possui como objetivo agrupar os dados e compartilhá-los em grupos. Um dos recursos importante que é fornecida pela versão do OpenCV é a possibilidade que o usuário tem para personalizar o grupo para melhor atender a sua funcionalidade no projeto. Um programador

pode implementar o código especificando quais iterações irão percorrer, bem como escolher quais grupos serão separados para o tratamento de imagem. Com isso ele pode decidi que valores associados ao RGB irá servir de crítero para agrupar os dados.

No caso do algoritmo Hough Circle Transform foi utilizado para encontrar círculos em uma imagem. A partir disso, esse círculo é usado para criar uma máscara. Esta máscara foi implementada para separar o que é a cor da tigela em relação a cor da ração, de modo a viabilizar o tratamento da imagem.

Com este código é possível determinar se há ração na tigela medindo os valores de intensidade da tigela. Por sua vez, se houver ração, isso indicará que existe uma menor intensidade de cor em relação a uma tigela vazia.

Usando o tratamento de imagem possibilita uma redução significativa no quesitos de números de cores na imagem, além de fazer as bordas dos objetos em análise se destacarem mais.

IX. Protótipo da estrutura do comedouro etrônico

Figura 5. Protótipo da estrutura do comedouro etrônico

Pensando em como testar o projeto de fato com um menor investimento possível foi criado um protótipo da estrutura do comedouro eletrônico. Tal estratégia está relacionada com a a sigla MVP, que é a sigla Minimum Viable Product (Produto mínimo viável).

Esta estrutura representa uma versão mínima do produto, na qual pussui a finalidade de cumprir com os requisitos necessários para a validação do projeto.

A partir da confecção dessa estrutura foi possível testar realmente a teoria na prática. Nela foi observado que a mesma cumpria com os requisitos preestabelicos para o projeto. Uma delas foi a liberação de ração com a utilização do servo motor. Além disso, foi testado o servidor para verificar se o mesmo enviava e recebia as mensagens via o bot do telegram.

X. Resultados

A partir da integração do código do servidor com a conexão do bot do Telegram, foi possível aferir os seguintes resultados que foram proposto para o projeto. A seguir foram tirados prints da tela do celular mostrando quais são as funções disponíveis para o usuário interagir com o programa com auxílio da Raspberry Pi 3.

Figura 6. Tela inicial do bot do Telegram

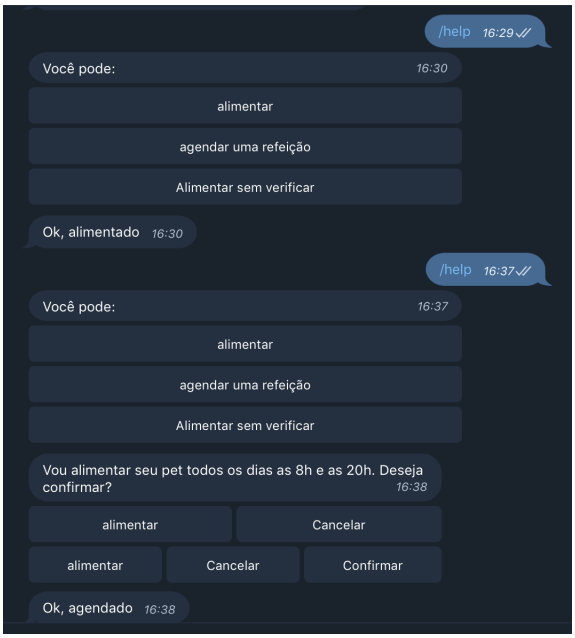

Figura 7. As opções que o usuário tem para interagir com o progrma

Em relação a (figura 6) e (figura 7), ambas mostram a tela inicial do programa. A partir disso, é mostrado uma mensagem para usuário falando que ele pode digitar o comando "/help"para ele ter acesso a mais opçoes. Após a execução desse comando é apresentado três opções, sendo elas: o comando de alimentar, agendar uma refeição e alimentar sem verificar.

No quesito do comando, agendar uma refeição, o usuário do sistema tem a possibilidade de agendar um horário, e aí quando der esse horário o sistema é acionado para liberar a ração na tigela do animal. Enquanto que no comando, alimentar sem verificar, o usuário pode decidir liberar a ração sem fazer o tratamento de imagem. Foi pensando essa função, pois o dono do animal de estimação pode está em casa no momento e acabar verificando que a tigela de ração já está vazia, e aí ele pode executar esse comando, fazendo com seja liberado a ração.

Importante ressaltar que na (figura 8) é mostrado uma mensagem avisando ao usuário que a tigela está cheia de ração. Isso só foi possível, pois o usuário acionou o comando, alimentar, onde é requisitado ao sistema que tire uma foto da tigela com ração, trate essa imagem, e aí retorne, se tem ração ou não na tigela.

Figura 8. Mostra o resultado após o tratamento de imagem

Para continuação do projeto recomenda-se que todo o processo vinculado ao processamento de imagem seja feito pelo um servidor mais potente, haja vista que o processamento de imagem sendo feita pela Raspberry Pi 3 é menos eficiente, pois seu processador é muito limitado para esse tipo de aplicação.

Por meio dos testes realizados foi possível constatar que o projeto funciona bem como um comedouro automático para alimentar cachorros e gatos em determinado período do dia. Porém é importante ressaltar, que não tivemos êxito no que diz respeito ao sobrepeso do animal, haja vista que esse último dependia da implementação de uma função, que conseguisse como objetivo controlar alimentação do bichinho de estimação. Porém avançamos muito no desenvolvimento do projeto, no que tange a função de agendamento, onde este possui a finalidade de possibilitar o usuário a agendar um horário para, que o cachorro ou gato, alimente-se adequadamente durante o dia.

XI. Bibliografia

Referências

[1] dicas para auxiliar o seu cachorro comer-mais devagar. , 2018. Disponível em: <https://www.portaldodog.com.br/cachorros/adultos-cachorros/alimentacao-adulto/dicas-para-auxiliar-o-seu-cachorro-comer-mais-devagar>. Acesso em: 29 set. 2019.

[2] SERVO Motor SG90 Datasheet. [S. l.], 2014. Disponível em: http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf. Acesso em: 6 out. 2019.

[3] OS BOTS do Telegram estão mais inteligentes. [S. l.], 2016. Disponível em: https://tecnoblog.net/194163/telegram-bots-2/. Acesso em: 6 out. 2019.

[4] CACHORRO comendo muito rápido? Como fazer ele comer mais devagar. [S. l.], 13 fev. 2017. Disponível em: https://tudosobrecachorros.com.br/cachorro-comendo-muito-rapido-como-fazer-ele-comer-mais-devagar/. Acesso em: 6 out. 2019.

[5] OS 5 Melhores Comedouros Para Cães Do Mercado: Qual é o melhor comedouro para cães?. [S. l.], 2018. Disponível em: https://amoraospets.com/melhor-comedouro-para-caes/. Acesso em: 6 out. 2019.

XII. Apêndice

Apêndice A: Implementação do código principal em C ++:

```
#include <stdio.h>
#include <unistd.h>
#include <thread>
#include <csignal>
#include <cstdio>
#include <cstdlib>
#include <tgbot/tgbot.h>
#include <unistd.h>
#include <wiringPi.h>
#include "verificaTigela.h"

#define SERVO 1

using namespace std;
using namespace TgBot;

void ScheduleFeed()
{
    cout << "Agendando" << endl;

    system("crontab ../activate.agent");
}

void takePic()
{
    cout << "Tirando foto" << endl;

    system("fswebcam 320x240 foto_img.jpg");
}

bool verifyBowl(string photoFilePath)
{
    cout << "Verificando tigela" << endl;

    bool existencia;
    // system("rm -rf foto_img.jpg")
    // thread takePhoto(takePic);
    // takePhoto.join();

    char *tab2 = new char
    [photoFilePath.length() + 1];
```

```
    strcpy(tab2, photoFilePath.c_str());

    existencia = existencia_racao(tab2);
    cout << "tem ração: "
    << existencia << endl;
    return existencia;
}

void sqwv(int pin, int degree, int N)
{
    int t1 =
    (100 * degree + 4) / 9 + 1500;
    int t2 = 20000 - t1;
    int i;
    for (i = 0; i < N; i++)
    {
        digitalWrite(pin, HIGH);
        usleep(t1);
        digitalWrite(pin, LOW);
        usleep(t2);
    }
}

void feederFunction(int delayTime, int N)
{
    cout << "Alimentando" << endl;
    sqwv(SERVO, -90, N);
    sleep(delayTime);
    sqwv(SERVO, 0, N);
};

int main()
{
    const string photoFilePath
    = "foto_img.jpg";
    const string photoMimeType
    = "image/jpeg";
    int N = 40;
    wiringPiSetup();
    pinMode(SERVO, OUTPUT);
    sqwv(SERVO, 0, N);
    Bot bot("931015860:
    AAHG6qZTMlopgG29lXC6-_
    rAPSmNrKiYXm4");

    InlineKeyboardMarkup::Ptr keyboard
    (new InlineKeyboardMarkup);
    InlineKeyboardMarkup::Ptr keyboard2
    (new InlineKeyboardMarkup);
    vector<InlineKeyboardButton::Ptr> row0;
    InlineKeyboardButton::Ptr checkButton
    (new InlineKeyboardButton);
    InlineKeyboardButton::Ptr checkButton2
    (new InlineKeyboardButton);
    InlineKeyboardButton::Ptr checkButton3
    (new InlineKeyboardButton);
    InlineKeyboardButton::Ptr checkButton4
    (new InlineKeyboardButton);
    InlineKeyboardButton::Ptr checkButton5
    (new InlineKeyboardButton);
    vector<InlineKeyboardButton::Ptr> row1;
    vector<InlineKeyboardButton::Ptr> row2;

    checkButton->text
    = "alimentar";
    checkButton->callbackData
    = "alimentar";
    row0.push_back
    (checkButton);
    keyboard->inlineKeyboard.
    push_back(row0);
    checkButton2->text
    = "agendar uma refeição";
    checkButton2->callbackData
    = "agendar";
    row1.push_back
    (checkButton2);
    keyboard->inlineKeyboard.
    push_back(row1);

    checkButton3->text
    = "Alimentar sem verificar";
    checkButton3->callbackData
    = "semVerificarAlimentar";
    row2.push_back(checkButton3);
    keyboard->inlineKeyboard.
    push_back(row2);

    checkButton4->text = "Cancelar";
    checkButton4->callbackData = "cancelar";
    row0.push_back(checkButton4);
    keyboard2->inlineKeyboard.push_back(row0);

    checkButton5->text = "Confirmar";
    checkButton5->callbackData = "confirmado";
    row0.push_back(checkButton5);
    keyboard2->inlineKeyboard.push_back(row0);

    bot.getEvents().onCommand
        ("start", [&bot](Message::Ptr message) {
        bot.getApi().sendMessage(message->chat->id
        "Olá, vou te ajudar a manter seu
        pet alimentado.
        Use o comando /help para mais informações"
    });

    bot.getEvents().onCommand
        ("semVerificarAlimentar", [&bot]
        (Message::Ptr message) {
        thread feeder(feederFunction, 2, 40);
        feeder.join();
        bot.getApi().sendMessage
```

```
    (message->chat->id, "Alimentado");
});

bot.getEvents().onCommand
("semVerificarAlimentar", [&bot]
(Message::Ptr message) {
    thread feeder(feederFunction, 2, 40);
    feeder.join();
    bot.getApi().sendMessage
    (message->chat->id, "Alimentado");
});

bot.getEvents().onCommand
("alimentar", [&bot, &photoFilePath,
&photoMimeType]
(Message::Ptr message) {
    cout << message << endl;
    bot.getApi().sendMessage
    (message->chat->id,
    "Aguarde por favor!");
    bool existencia
    = verifyBowl(photoFilePath);
    if (!existencia)
    {
        thread feeder(feederFunction, 2, 40);
        feeder.join();
        string response = "Ok, alimentado";
        bot.getApi().sendMessage
        (message->chat->id, response);
    }
    else
    {
        bot.getApi().sendPhoto
        (message->chat->id,
        InputFile::fromFile(photoFilePath,
        photoMimeType),
        "A tigela ainda está cheia!");
    }
});

bot.getEvents().onCommand
("agendar", [&bot, &keyboard2]
(Message::Ptr message) {
    string response
    = "Vou alimentar seu pet
    todos os dias as 8h e as 20h.
    Deseja confirmar?";
    bot.getApi().sendMessage
    (message->chat->id,
    response, false, 0,
    keyboard2, "Markdown");
});

bot.getEvents().onCallbackQuery([&bot]
(CallbackQuery::Ptr query) {
    if (StringTools::startsWith
    (query->data, "confirmado"))
    {
        thread schedule
        (ScheduleFeed);
        schedule.join();
        string response
        = "Ok, agendado";
        bot.getApi().
        sendMessage
        (query->message->chat->id,
        response);
    }
});

bot.getEvents().
onCallbackQuery([&bot]
(CallbackQuery::Ptr query) {
    if (StringTools::startsWith
    (query->data,
    "semVerificarAlimentar"))
    {
        thread feeder
        (feederFunction, 2, 40);
        feeder.join();
        string response
        = "Ok, alimentado";
        bot.getApi().
        sendMessage
        (query->message->
        chat->id, response);
    }
});

bot.getEvents().onCallbackQuery
([&bot, &keyboard2]
(CallbackQuery::Ptr query) {
    if (StringTools::startsWith
    (query->data, "agendar"))
    {
       string response =
       "Vou alimentar seu pet
       todos os dias as 8h e as 20h.
       Deseja confirmar?";
    bot.getApi().sendMessage
    (query->message->chat->id,
    response, false, 0, keyboard2,
    "Markdown");
    }
});

bot.getEvents().onCommand
("help", [&bot, &keyboard]
(Message::Ptr message) {
    bot.getApi().sendMessage
    (message->chat->id,
    "Você pode: ", false, 0,
```

```
    keyboard, "Markdown");
});

bot.getEvents().onCallbackQuery
([&bot](CallbackQuery::Ptr query) {
    if (StringTools::startsWith
    (query->data, "alimentar"))
    {
        const string photoFilePath
        = "foto_img.jpg";
        const string photoMimeType
        = "image/jpeg";

        bool existencia
        = verifyBowl(photoFilePath);
        if (!existencia)
        {
            thread feeder
            (feederFunction, 2, 40);
            feeder.join();
            string response
            = "Ok, alimentado";
            bot.getApi().sendMessage
            (query->message->chat->id,
            response);
        }
        else
        {
          bot.getApi().sendPhoto
          (query->message->chat->id,
          InputFile::fromFile
          (photoFilePath,
          photoMimeType),
            "A tigela ainda
            está cheia!");
        }
    }
});


bot.getEvents().onCallbackQuery
([&bot](CallbackQuery::Ptr query) {
    if (StringTools::startsWith
    (query->data, "cancel"))
    {
        string response = "ok";
        bot.getApi().sendMessage
        (query->message->chat->id,
        response);
    }
});

signal(SIGINT, [](int s) {
    printf("SIGINT got\n");
    exit(0);
});
    try
    {
        printf("Bot username:
        %s\n", bot.getApi().
        getMe()->
        username.c_str());
        bot.getApi().
        deleteWebhook();
        TgLongPoll longPoll(bot);
        while (true)
        {
            printf("Long poll started\n");
            longPoll.start();
        }
    }
    catch (TgException &e)
    {
        printf("error: %s\n", e.what());
    }
    return 0;
}
```

Apêndice B: Código de verificação da tigela para saber se tem ração ou não:

```
#include "verificaTigela.h"

using namespace cv;

bool existencia_racao(char *filename){

bool existe_racao;
int pixeis_racao;
int pixeis_tigela;

Mat fonte = imread(filename, 1);
Mat amostras(fonte.rows *
fonte.cols, 3, CV_32F);
for (int y = 0; y < fonte.rows; y++)
for (int x = 0; x < fonte.cols; x++)
for (int z = 0; z < 3; z++)
amostras.at<float>
(y + x*fonte.rows, z)
= fonte.at<Vec3b>(y, x)[z];

int clusterCount = 4;
Mat labels;
int tentativas = 5;
Mat centros;
kmeans(amostras, clusterCount,
labels, TermCriteria(CV_TERMCRIT_ITER |
CV_TERMCRIT_EPS, 10000, 0.0001),
tentativas, KMEANS_PP_CENTERS, centros);

    Mat new_image(fonte.size(), fonte.type());
```

```
for (int y = 0; y < fonte.rows; y++)
for (int x = 0; x < fonte.cols; x++){
int cluster_idx
= labels.at<int>(y + x*fonte.rows, 0);
new_image.at<Vec3b>(y, x)[0]
= centros.at<float>(cluster_idx, 0);
new_image.at<Vec3b>(y, x)[1]
= centros.at<float>(cluster_idx, 1);
new_image.at<Vec3b>(y, x)[2]
= centros.at<float>(cluster_idx, 2);
}

// imshow("clustered image", new_image);
imwrite("clustered_image.jpg", new_image);
Mat img =
imread("clustered_image.jpg", 0);
// Contabilizar quantos pixeis
estao associados ao label da racao,
// e quantos correspondem
ao label da tigela
pixeis_racao = 0;
pixeis_tigela = 0;
for (int y = 0; y < fonte.rows; y++){
for (int x = 0; x < fonte.cols; x++){
int cluster_idx
= labels.at<int>
(y + x*fonte.rows, 0);
if (cluster_idx == 1){
// 1 usar o label da racao
pixeis_racao++;
}
if (cluster_idx == 0){
// 2 usar o label da tigela
pixeis_tigela++;
}
}
}

if ((float)((1.0 * pixeis_racao)
/ (1.0 * pixeis_tigela) > 0.2)){
// estar fazendo a conversao
    // para float
existe_racao = true;
}
else
existe_racao = false;

Mat cimg;
Mat thresh = Mat::zeros
(img.size(), img.type());
medianBlur(img, img, 5);
cvtColor
(img, cimg, COLOR_GRAY2BGR);
std::vector<Vec3f> circles;

HoughCircles
(img, circles, HOUGH_GRADIENT, 1, 500,
100, 30, 200, 450); //
//change the last two parameters
//(min_radius & max_radius)
to detect larger circles

for (size_t i = 0; i < circles.
size(); i++){
Vec3i c = circles[i];
ellipse(cimg, Point
(c[0], c[1] * 3 / 4),
Size(c[2], c[2] * 3 / 4), 0, 0,
360, Scalar(0, 255, 0), 3, LINE_AA);
ellipse(thresh, Point
(c[0], c[1] * 3 / 4),
Size(c[2], c[2] * 3 / 4), 0, 0,
360, Scalar(255, 255, 255),
-1, LINE_AA);
circle(cimg, Point(c[0], c[1]), 2,
Scalar(0, 255, 0), 3, LINE_AA);
}

//imshow("detected circles", cimg);
//imshow("threshold img", thresh);
Mat hist;
int histSize = 256;
float range[] = { 0, 256 };
const float* histRange
= { range };
calcHist(&img, 1, 0, thresh, hist, 1,
&histSize, &histRange, true, false);

// Draw hist
int hist_w = 512;
int hist_h = 400;
int bin_w = cvRound
((double)hist_w / histSize);
Mat histImage(hist_h, hist_w,
CV_8UC3, Scalar(0, 0, 0));

/// Normalize the result to
[ 0, histImage.rows ]
normalize(hist, hist, 0,
histImage.rows, NORM_MINMAX,
-1, Mat());

/// Draw for each channel
for (int i = 1; i
< histSize; i++){
line(histImage,
Point(bin_w*(i - 1), hist_h -
cvRound(hist.at<float>(i -
1))),
Point(bin_w*(i), hist_h -
cvRound(hist.at<float>(i))),
Scalar(255, 0, 0), 2, 8, 0);
```

```
}

/// Display
//namedWindow("calcHist Demo",
CV_WINDOW_AUTOSIZE);
//imshow("calcHist Demo",
histImage);
std::cout << "Mean intensity is:
" << mean(hist) << std::endl;
waitKey();
return existe_racao;
}
```

Apêndice C: Implementação da função para controlar o motor:

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

#define SAIDA 1

void sqwv(int pin, int degree, int N)
{
    int t1 = (100 * degree + 4) / 9 + 1500;
    int t2 = 20000 - t1;
    int i;
    for (i = 0; i < N; i++)
    {
        digitalWrite(pin, HIGH);
        usleep(t1);
        digitalWrite(pin, LOW);
        usleep(t2);
    }
}

int main(void)
{
    int N = 40;
    wiringPiSetup();
    pinMode(SAIDA, OUTPUT);

    delay(1000);
    sqwv(SAIDA, -90, N);
    delay(2000);
    sqwv(SAIDA, 0, N);
    return 0;
}
```