Analisador de Imagens de Alvos de Competições para a Plataforma Android

Acadêmico – André Luís Hensel
Orientador – Mauro Marcelo Mattos

Roteiro

- Introdução
- Fundamentação teórica
- Desenvolvimento
- Resultados e discussão
- Conclusão / Extensão
- Demonstração

Introdução

- Objetivos do trabalho
  - Criar uma aplicação para dispositivos móveis para auxiliar os competidores na análise dos alvos e de seus disparos
  - Identificar os alvos e de seus disparos
  - Destacar os alvos e seus disparos

FURB
UNIVERSIDADE REGIONAL DE BLUMENAU

Fundamentação teórica

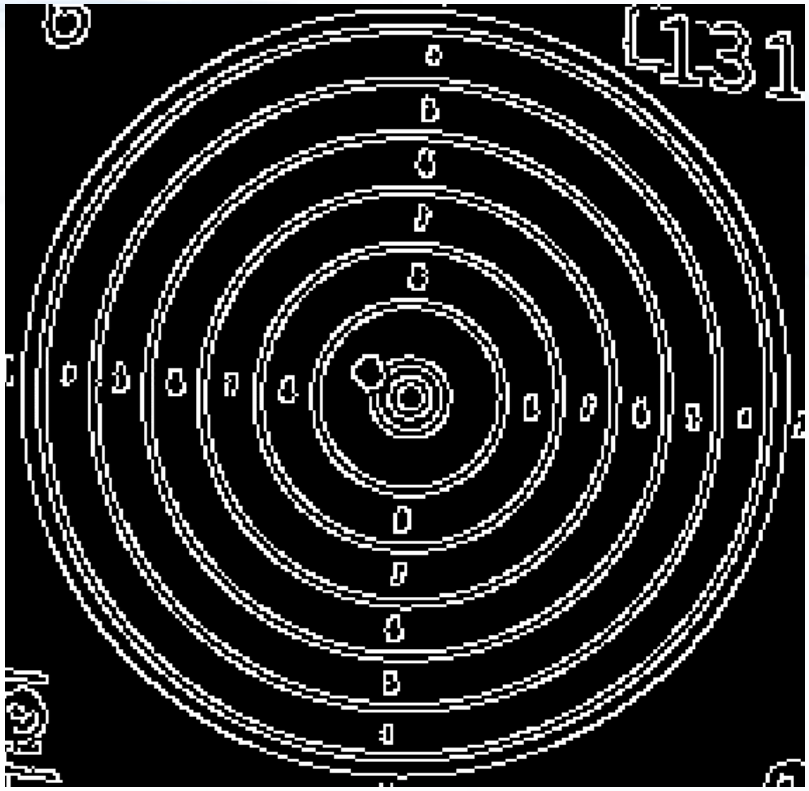
- Visão computacional
  - Conjunto métodos e técnicas através dos quais um sistema computacional pode ser capaz de interpretar imagens

- Etapas de um Sistema de Visão Computacional
  - Aquisição
  - Pré-processamento
  - Extração

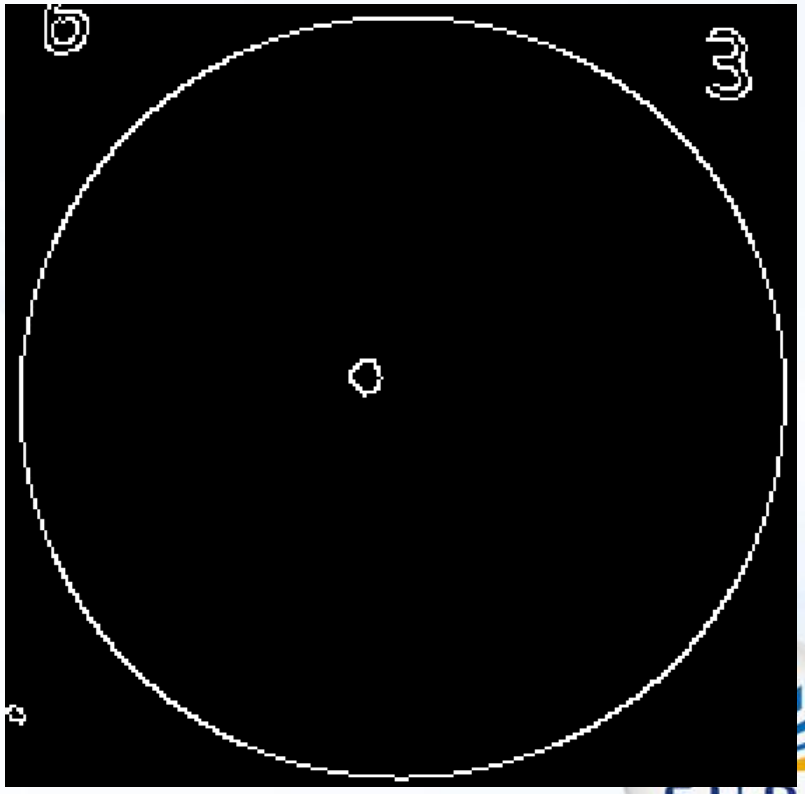
Fundamentação teórica

- Operador Canny

Limiar 100 / 200



Limiar 300 / 600



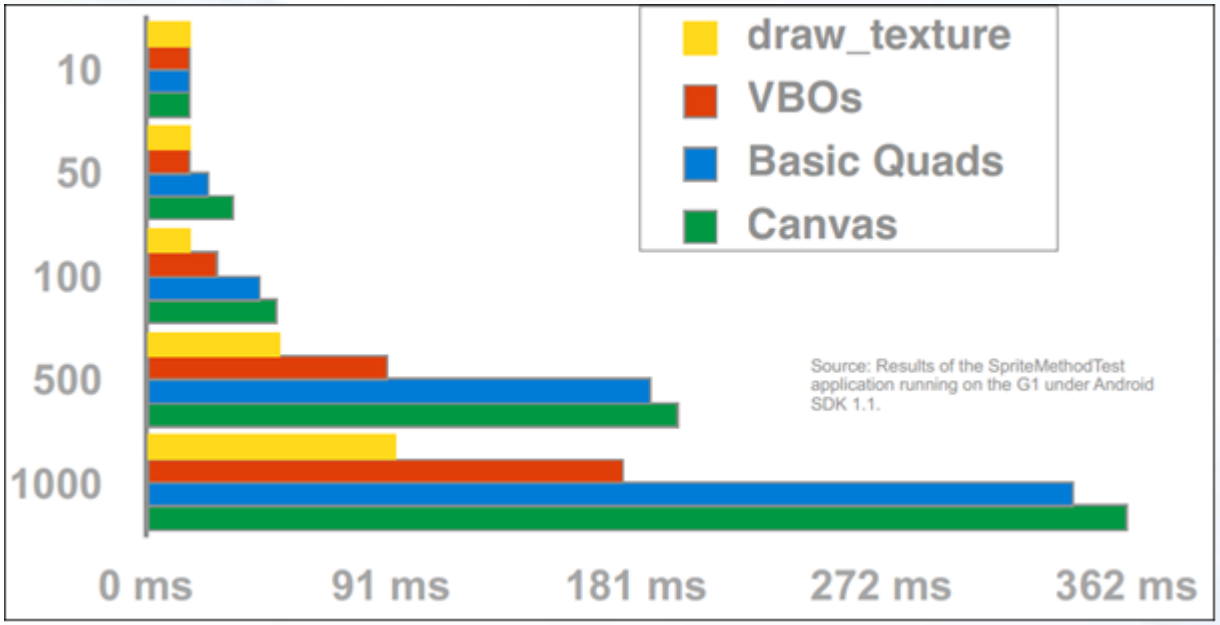
Fundamentação teórica

- Transformada de Hough

- OpenCV

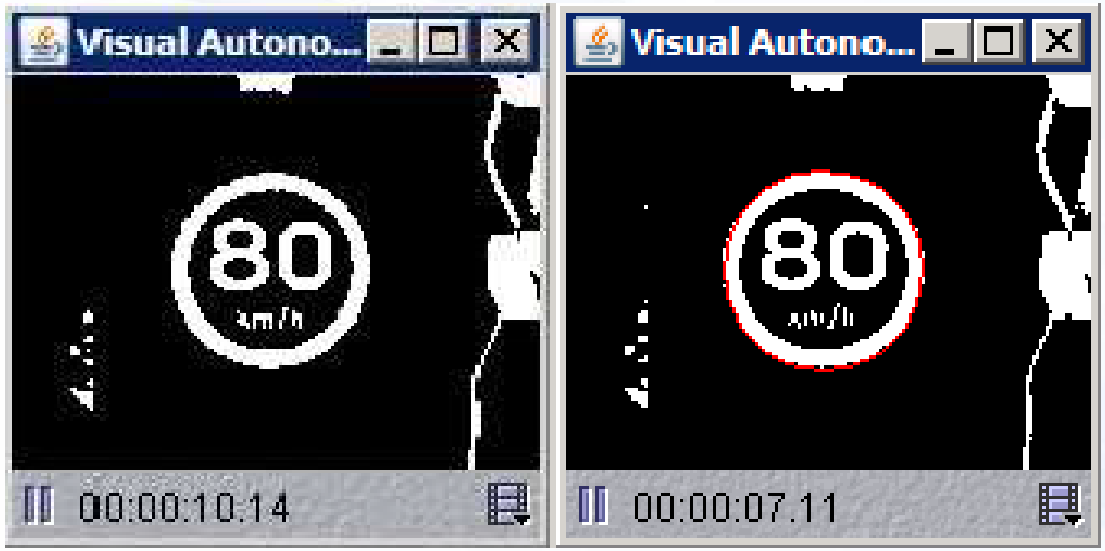
Fundamentação teórica

- OpenGL ES
  - Performance



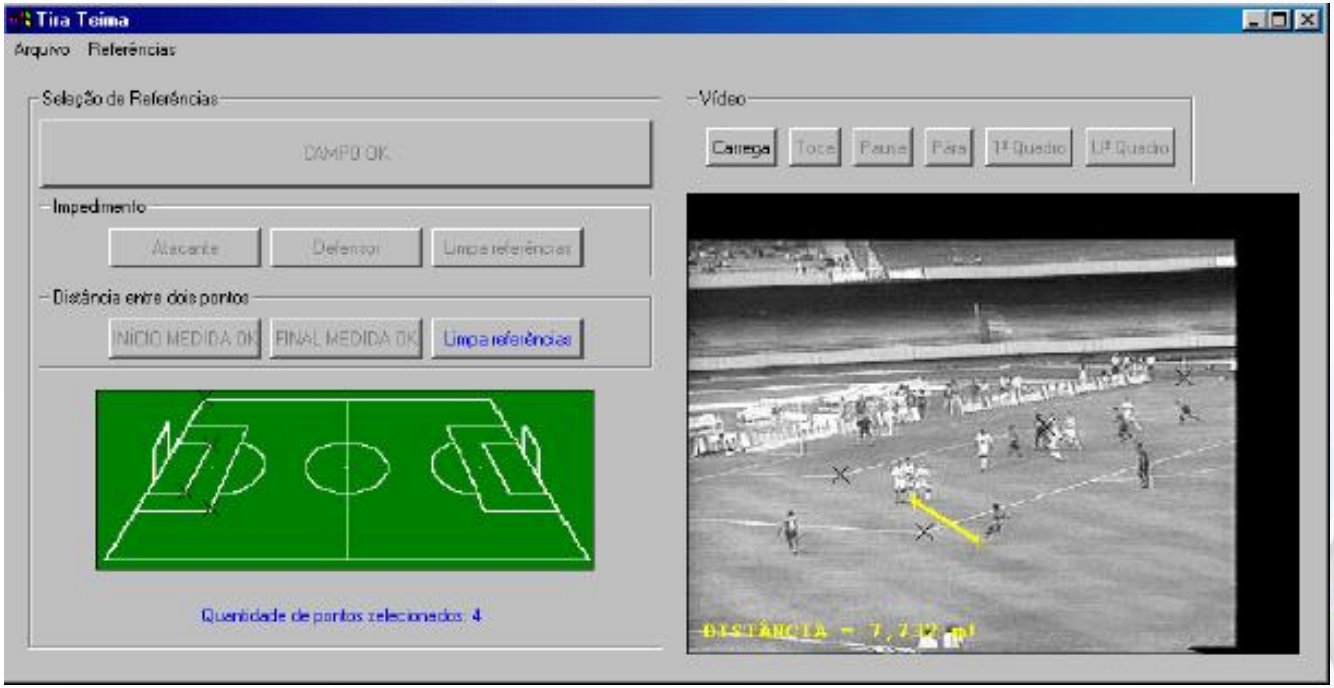
Fundamentação teórica

- Trabalhos correlatos
  - FURB - Visual Autonomy – Protótipo para reconhecimento de placas de trânsito



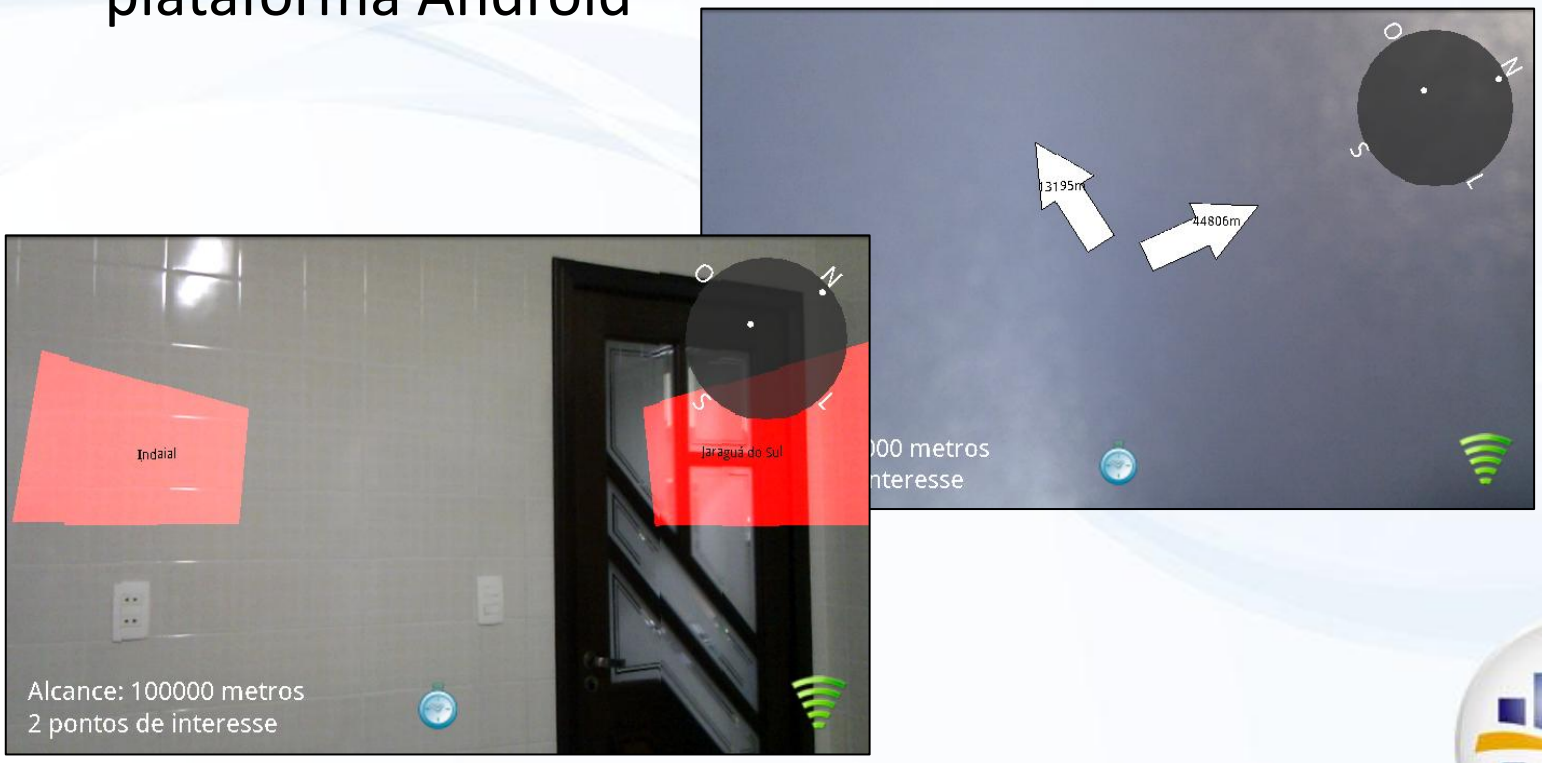
Fundamentação teórica

- Trabalhos correlatos
  - FURB - Calibração de câmeras para utilização no cálculo de impedimentos de jogadores de futebol a partir de imagens



Fundamentação teórica

- Trabalhos correlatos
  - FURB – Um estudo sobre realidade aumentada para a plataforma Android



Desenvolvimento

- Requisitos funcionais
  - Mostrar pontuação total, tempo entre disparos, horário de início e tempo para fim da prova
  - Projetar sobre a imagem do alvo pontuação do disparo e distância do mesmo para o centro do alvo
  - Permitir a visualização dos resultados

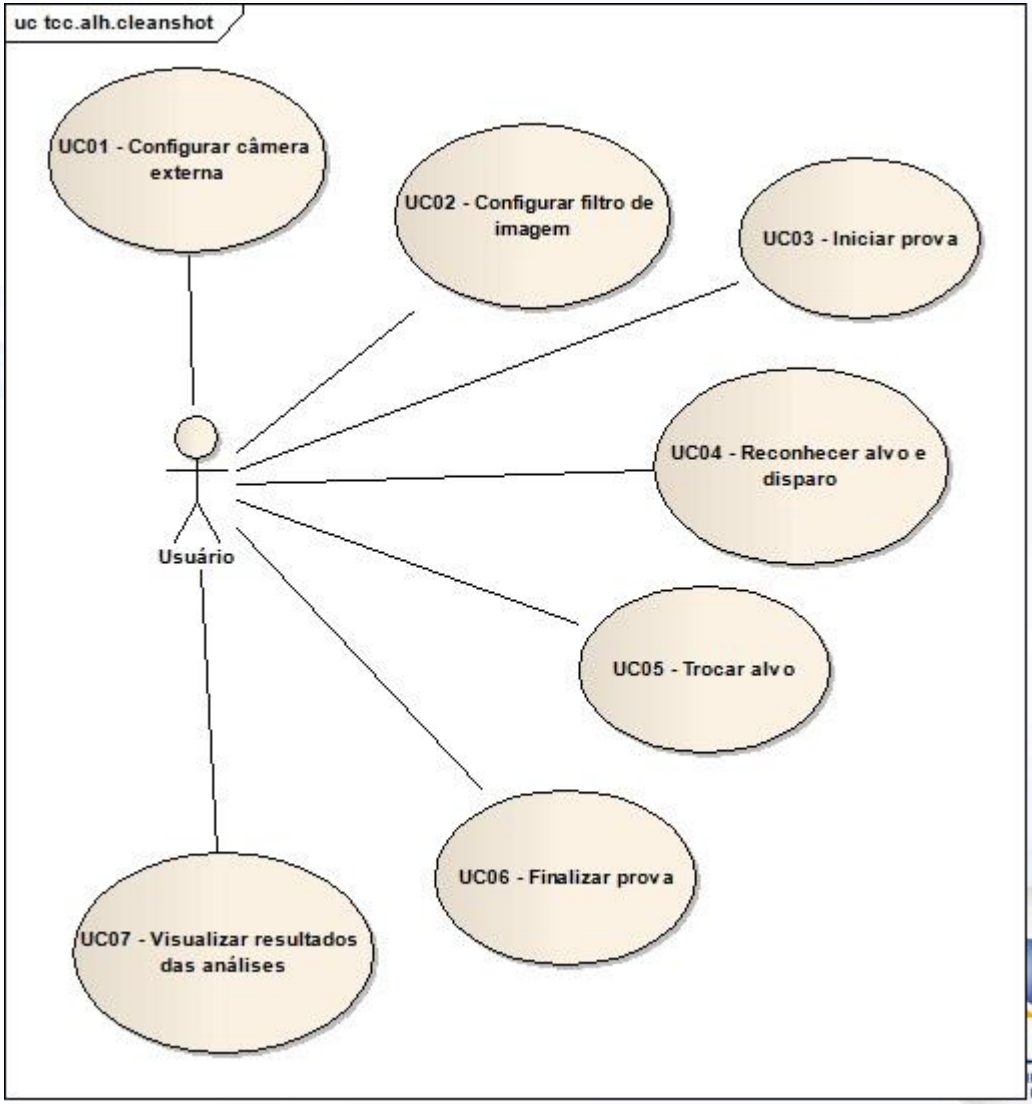
Desenvolvimento

- Requisitos não funcionais
  - Linguagem de programação Java
  - Ambiente de desenvolvimento Eclipse
  - Biblioteca OpenCV para análise das imagens
  - Banco de dados SQLite para armazenar as análises
  - Plataforma Android
  - Menos de 1 segundo para analisar uma imagem de um alvo

FURB
UNIVERSIDADE REGIONAL DE BLUMENAU

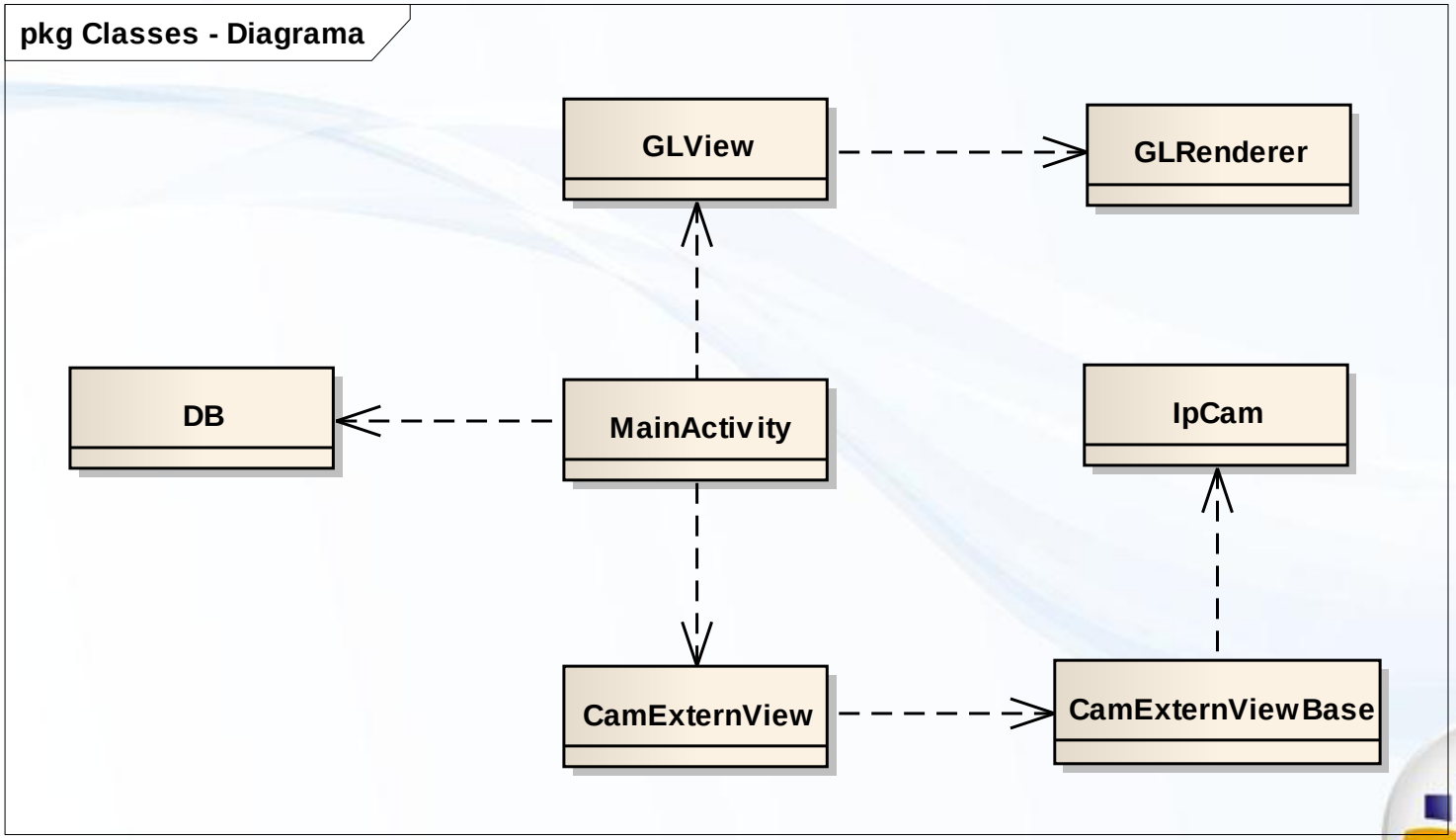
Desenvolvimento

- Especificação



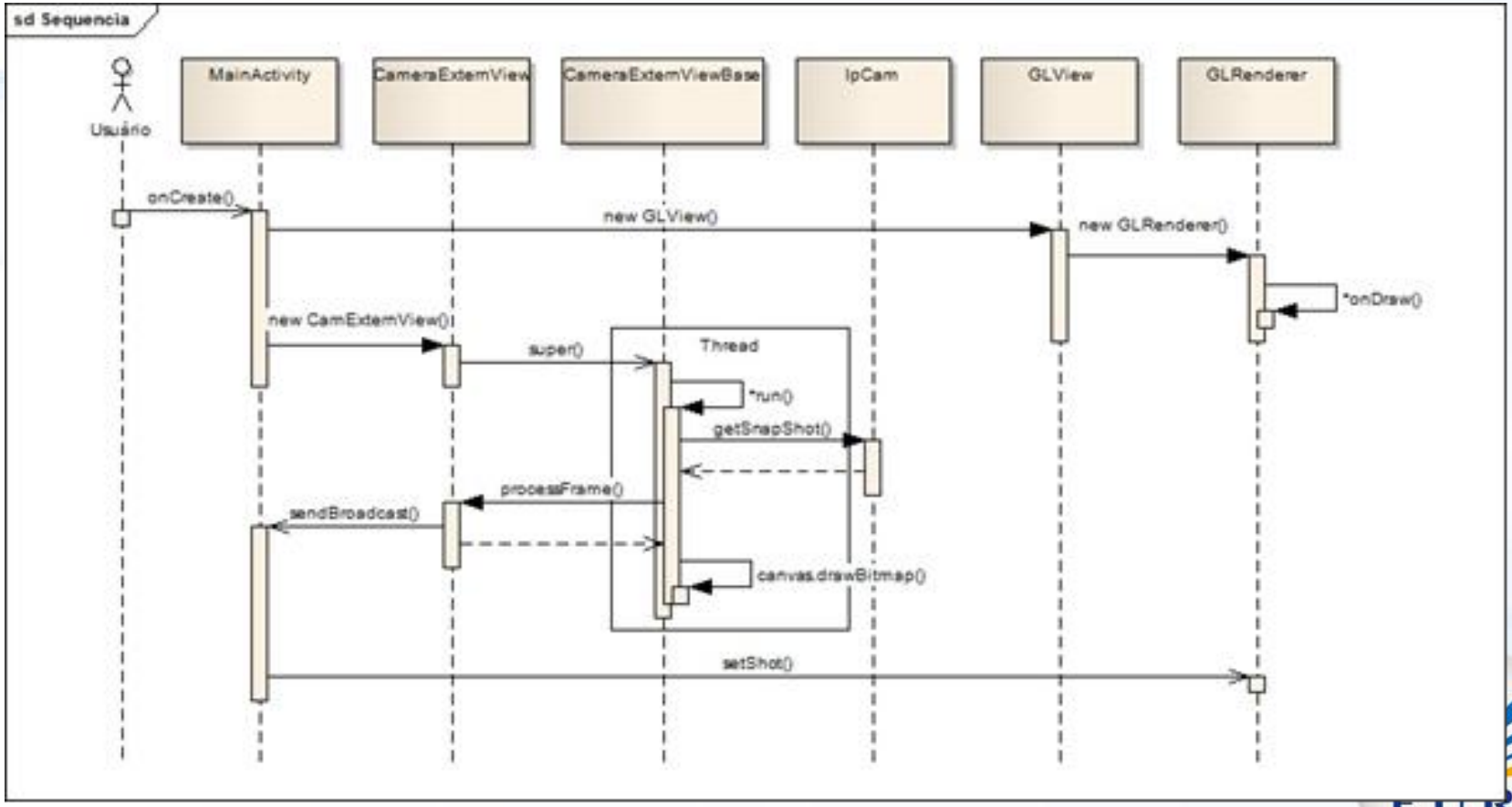
Desenvolvimento

- Principais classes



Desenvolvimento

- Sequência



Desenvolvimento

- Técnicas e ferramentas utilizadas
  - Linguagem Java
  - Eclipse Juno
  - Plugins e SDK da plataforma Android
  - Motorola XOOM
  - Câmera IP Foscam

FURB
UNIVERSIDADE REGIONAL DE BLUMENAU

Desenvolvimento

- Camadas

```
// Alocar a view do OpenglES
_glView = new GLView(_context);

// Alocar a view da câmera
_camView = new CamExternView(_context);

// Adicionar as camadas ao frame
_frame.addView(_glView);
_frame.addView(_camView);
```

Desenvolvimento

- CamExternViewBase
  - Inicializa a câmera
  - Thread
    - Obtém as imagens da câmera
    - Solicita o processamento da imagem
    - Desenha a imagem processada na tela

Desenvolvimento

- Thread da classe CamExternViewBase

```
while (_runThread) {
            Bitmap bmp = null;

            // Buscar a imagem na câmera
            bmp = _camera.getSnapShot();
            bmp = processFrame(bmp);

            Canvas canvas = _holder.lockCanvas();

            // Redimensionar a imagem
            float scaleWidth = (float) _width / bmp.getWidth();
            float scaleHeight = (float) _height / bmp.getHeight();
            Matrix matrix = new Matrix();
            matrix.postScale(scaleWidth, scaleHeight);

            // Desenhar a imagem
            canvas.drawBitmap(bmp, matrix, null);
            _holder.unlockCanvasAndPost(canvas);
            bmp.recycle();
}
```

FURB
UNIVERSIDADE REGIONAL DE BLUMENAU

Desenvolvimento

- CamExternView
  - Implementa a função abstrata *processFrame*

```
// Joga a imagem dentro de um array n-dimensional
Utils.bitmapToMat(bmp, _rgba);

// Transformar a imagem para cinza
Imgproc.cvtColor(_rgba, _gray, Imgproc.COLOR_BGRA2GRAY);
// Aplicar o filtro de blur
Imgproc.GaussianBlur(_gray, _gray, new Size(3, 3), 1f, 1f);

int iAccumulator = 60;
int rad = _defaultTargetRadius;
int interval = 2;

// Detectar os círculos na imagem
Imgproc.HoughCircles(_gray, _circles, Imgproc.CV_HOUGH_GRADIENT, 1, _gray.rows(),
_defaultCannyHigh, iAccumulator, rad, rad + interval);

// Encontrou algum círculo
if (_circles.cols() > 0) { // Desenhar informações do alvo  }

// Criar um Bitmap da imagem processada
_bmp = Bitmap.createBitmap(_rgba.cols(), _rgba.rows(), Bitmap.Config.ARGB_8888);
Utils.matToBitmap(_rgba, _bmp);
return _bmp;
```

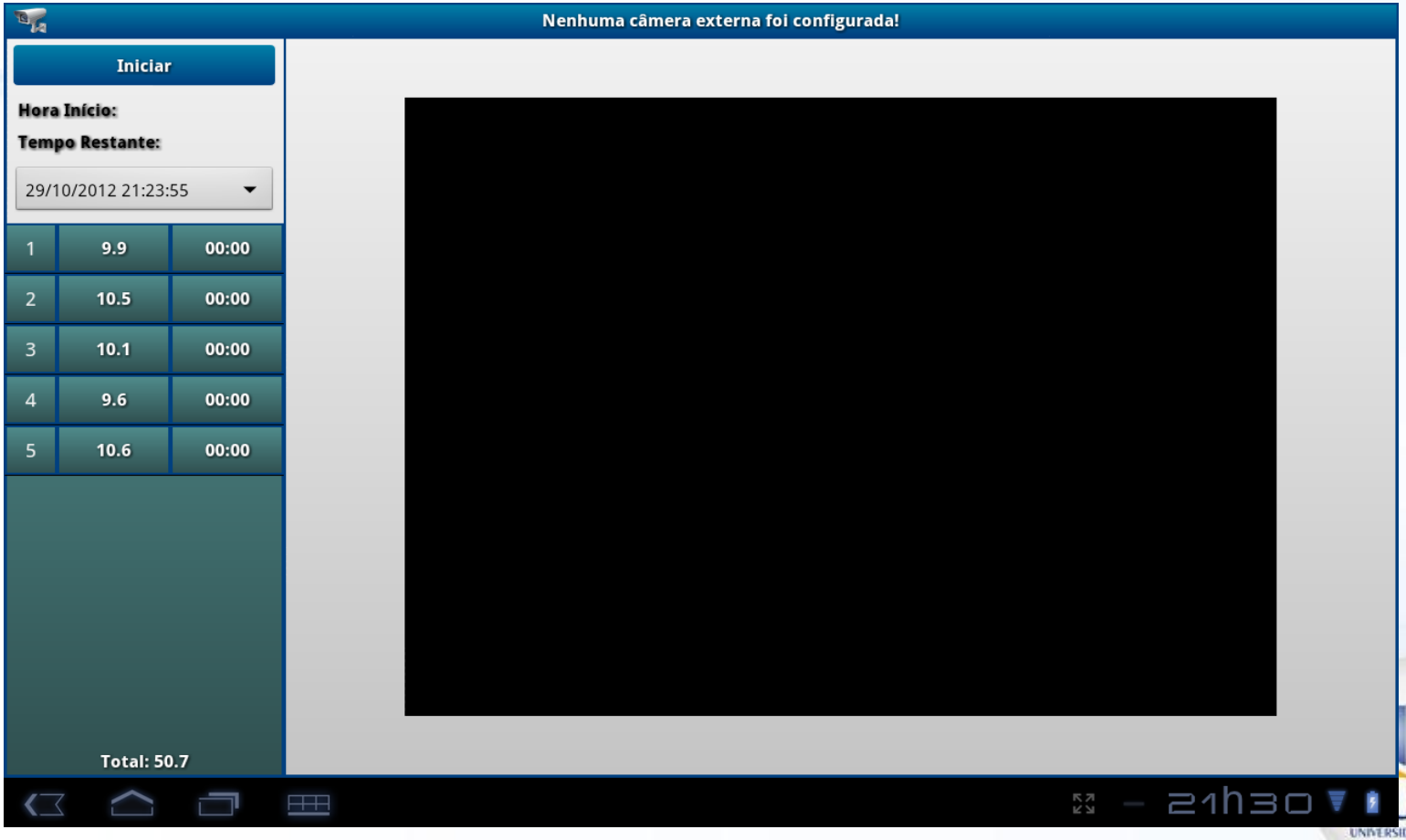
FURB
UNIVERSIDADE REGIONAL DE BLUMENAU

Desenvolvimento

- Operacionalidade da aplicação

Desenvolvimento

- Operacionalidade da aplicação

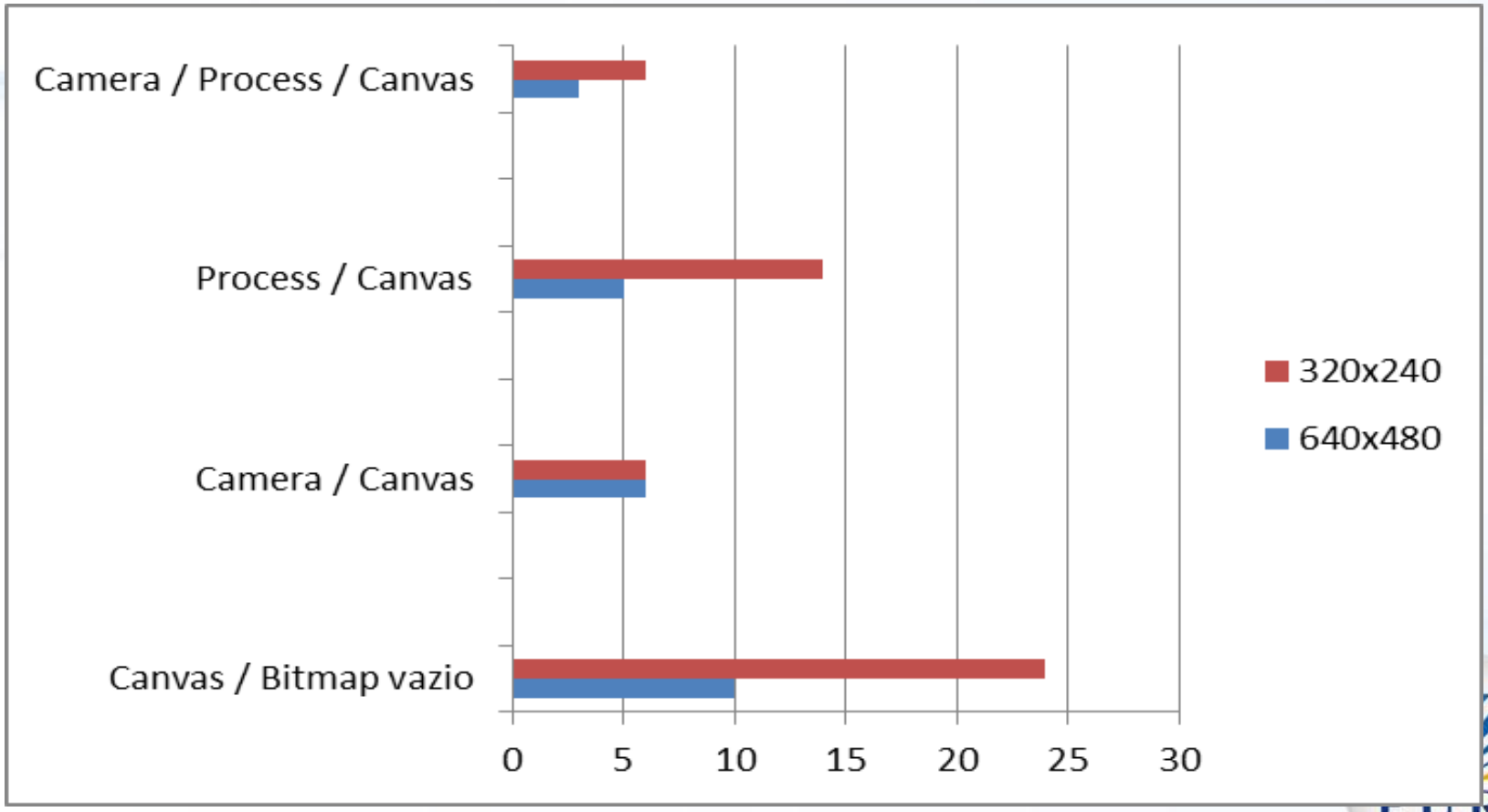


Resultados e discussão

- Resultados
- Resoluções
  - 320x240 e 640x480
- Problema na biblioteca OpenCV 2.31
- Limitação no OpenGL ES 1.0
- JavaCV

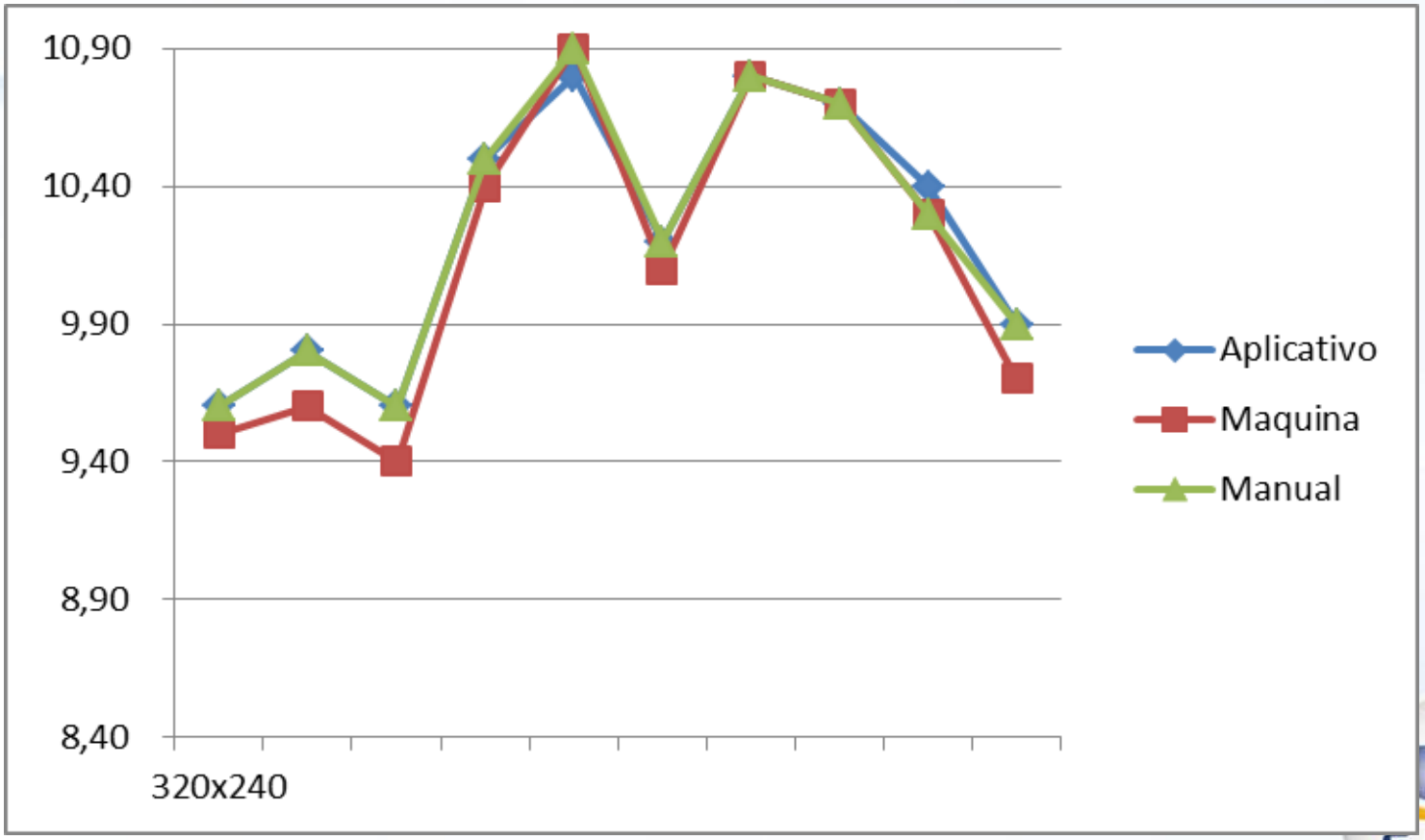
Resultados e discussão

- Gráfico de desempenho



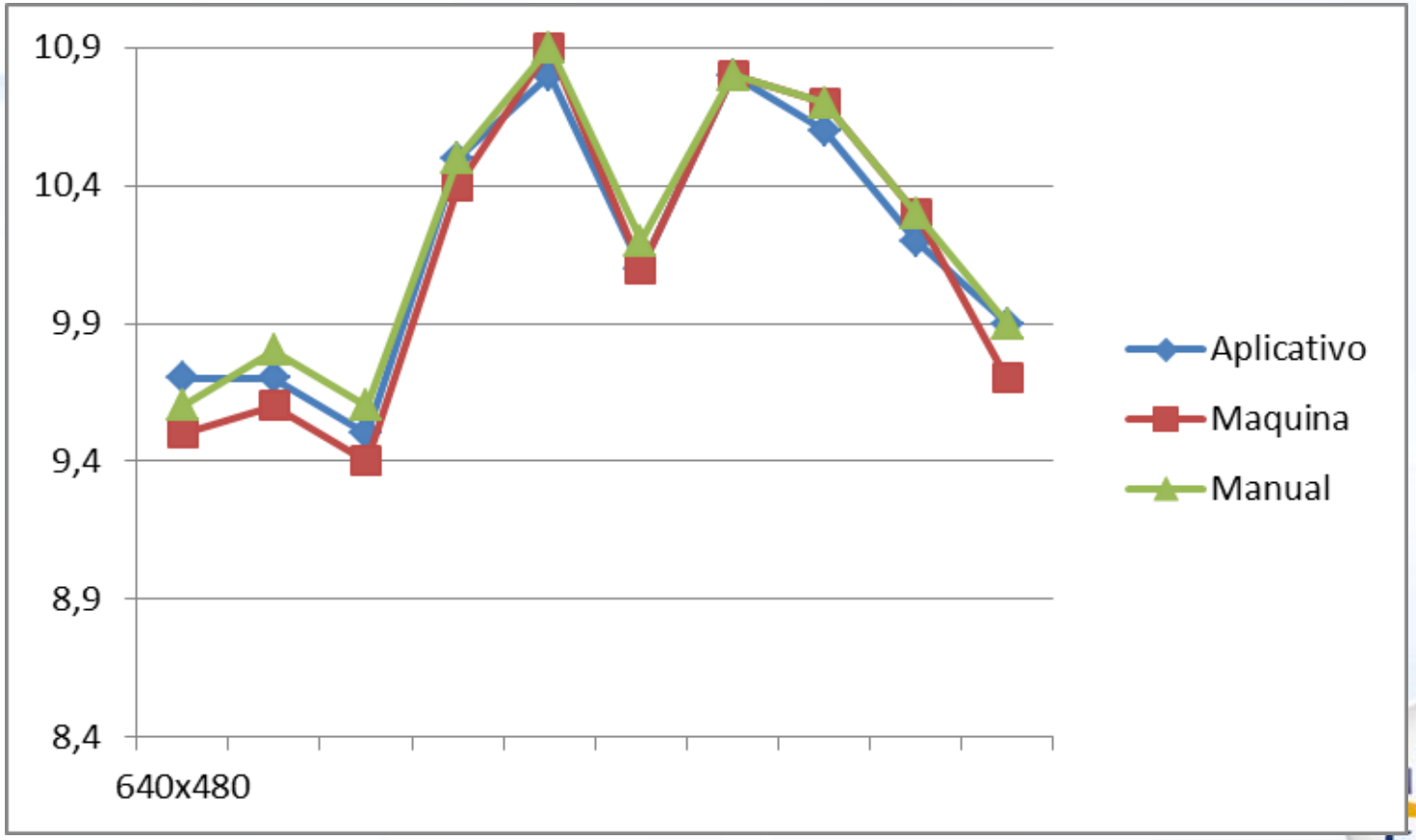
Resultados e discussão

- Gráfico de resultados – 320x240



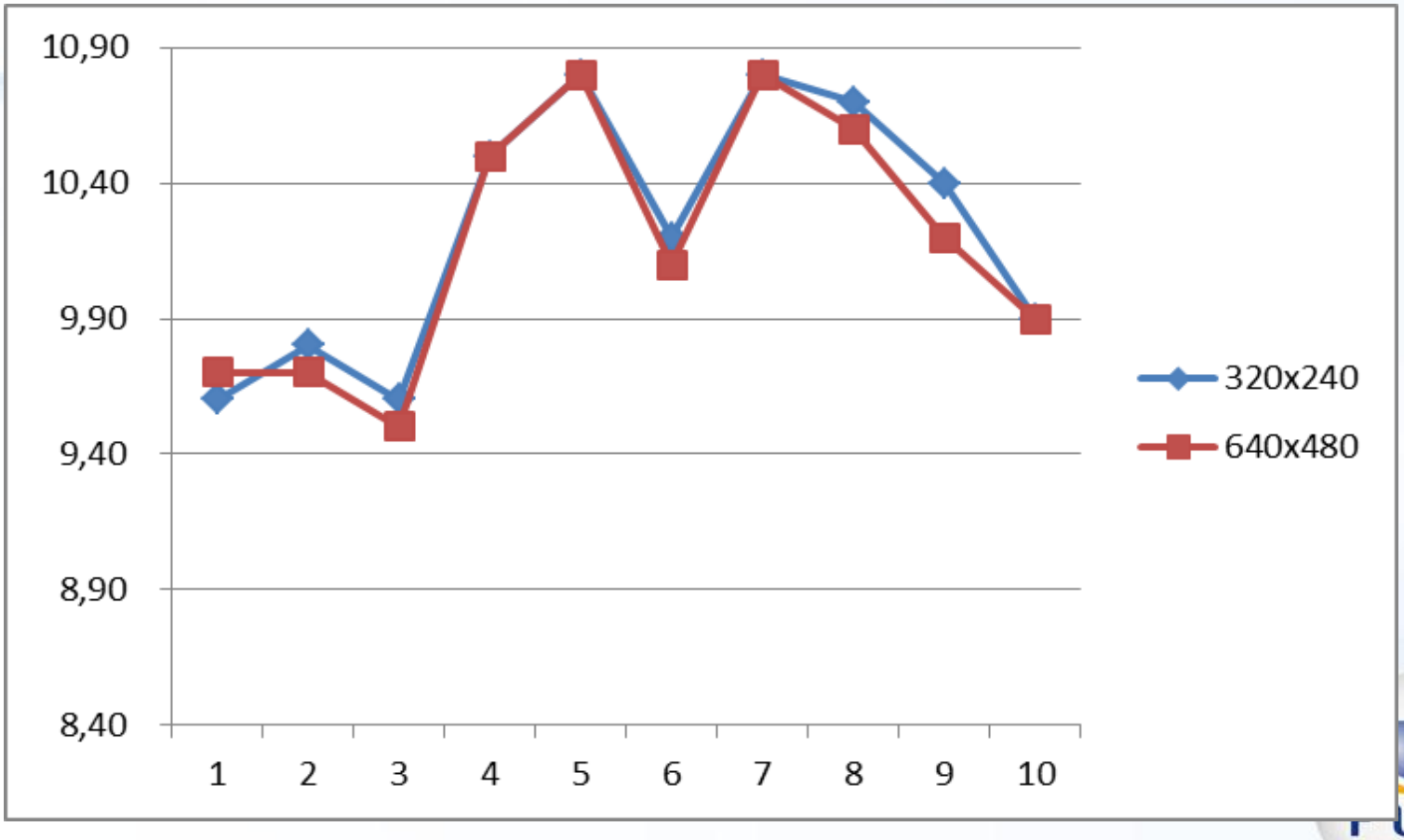
Resultados e discussão

- Gráfico de resultados – 640x480



Resultados e discussão

- Gráfico comparativo – 320x240 e 640x480



Conclusão

- Aplicação
- Resultados
- Desenvolvimento
  - Eclipse
  - Plugins do SDK

FURB
UNIVERSIDADE REGIONAL DE BLUMENAU

Extensões

- Câmera com zoom
- Calibragem da câmera
- Recortar área externa do alvo
- OpenCV para desenhar objetos gráficos
- Armazenar imagens dos alvos
- Aplicação servidor

Demonstração

Demonstração prática