Reconocimiento de imágenes utilizando GPUs, aplicacion a fútbol de robots

Ignacio Eguinoa
Facultad de Informática, UNLP

10 de febrero de 2015

Resumen

En este trabajo se desarrollan conceptos relacionados con la vision por computadoras utilizando unidades de procesamiento gráfico(GPUs). Se incluye una descripción de la libreria OpenCV, incluyendo el módulo que implementa aceleración mediante GPUs. Además, se realiza una desarrollo práctico que consiste en reimplementar una libreria para el procesamiento de imágenes provenientes de fúbol de robots. Se plantean variaciones en esta libreria que aceleran distintos pasos del procesamiento de imagenes utlizando una GPU. Las distintas variantes son evaluadas utilizando un sistema de pruebas y los resultados analizados en base a las caracterísicas de la arquitectura.

Índice general

Capítulo 1

Introduccion

1.1. Estructura del trabajo

En lo que resta de este primer capitulo se realiza una introducción a la arquitectura GPU. El objetivo es dar una idea general de las caracterísicas que posee y las posiblidades que ofrece tanto para procesamiento de gráficos como para cómputo de propósito general

En el capitulo 2 se describe de forma detallada la libreria OpenCV, principalmente los módulos y funciones que se utilizarán luego en el desarrollo.

El capitulo 3 contiene el desarrollo experimental del trabajo. En primer lugar se describe el contexto de la aplicación y la implementacion existente para el procesamiento de imágenes de fútbol robot(libreria bottracker). Luego se plantean modificaciones sobre esta librería, utilizando el módulo de GPU provisto por OpenCV. Se hacen evaluaciones de las distintas modificaciones y se analizan los resultados basandose en los conceptos explicados en los capitulos previos.

1.2. La arquitectura GPU

Capítulo 2

Libreria OpenCV

La libreria OpenCV [1] reune una gran cantidad de algoritmos asociados a la vision por computadoras. Es una libreria open-source que se distribuye bajo una licencia BSD. is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms.

La librería esta compuesta por distintos modulos que proveen funcionalidades independientes entre si. Algunos modulos comunes son:

core: Define funciones básicas utilizadas por los demás módulos y una estructura de datos que se utiliza para almacenamiento de imágenes(detallada mas adelante)

imgproc: contiene algoritmos para aplicar a imagenes (filtros, transformaciones, conversiones de colores, etc)

video: incluye algoritmos para estimar movimientos, seguimiento de objetos y para sustraer el fondo.

highgui: interface para captura de video.

El modulo core es el más importante por la utilidad de ... La principal función de la librería OpenCV es procesar imágenes, las cuales en cualquier sistema de cómputo se encuentran almacenadas como matrices numéricas. El módulo core contine una interface propia que es utilizada para el manejo de imagenes (entrada y salida) en todas las funciones de la libreria. Presenta una forma simple y segura de manejar este tipo de información y es, por lo tanto, una de las partes centrales de OpenCV.

Inicialmente, OpenCV fue implementado usando el lenguaje C y se usaban estructuras de memoria propias del lenguaje para manejar las imágenes. El problema de esto es que todo el proceso de alocar y desalocalr el espacio correspondiente se debe hacer de forma manual y depende de quien está utilizando la librería. A partir de la version 2.0 de OpenCV, el código se extendió usando el lenguaje C++ (aprovechando la compatibilidad entre ambos lenguajes) y en este proceso se introdujo una interface llamada Mat que apunta a automatizar todo el manejo de memoria. De esta forma, los programas que utilizaban la librería OpenCV se hacen mas simples de desarrollar y manejar, incluso para

[1] Open Source Computer Vision Library `http://opencv.org`

programas de gran tamaño. La mayoria de las funciones de OpenCV realizan la alocacion de memoria para el output automáticamente. Además, si el input consiste en un objeto Mat ya instanciado entonces el espacio de memoria de este es reutilizado

Mat es basicamente una clase con dos partes de datos: el encabezado de la matriz(contiene informacion tal como el tamaño de la matriz, el metodo usado para almacenarla, la direccion de memoria, etc) y un puntero a la matriz conteniendo los valores de los pixels (que puede tomar cualquier dimension, dependiendo del metodo usado para almacenar). El tamaño del header de la matriz es constante pero el tamaño de la matriz en si puede variar de imagen a imagen.

Dada la expansion en el uso de las arquitecturas GPU, se comenzó a implementar un modulo adicional que contiene optimizaciones realizadas sobre GPU. El modulo fue lanzado en el 2011, contiene algunos algoritmos que ya estan implementadas en diversos modulos de OpenCV y que fueron reimplementados con el fin de obtener una aceleracion extra mediante esta arquitectura.

El manejo de estructuras de datos es importante cuando interviene codigo sobre la gpu ya que la transferfencia es una parte relevante.

Por su parte, el modulo gpu define la siguiente clase:

class gpu::GpuMat Base storage class for GPU memory with reference counting. Its interface matches the Mat interface with the following limitations: no arbitrary dimensions support (only 2D) no functions that return references to their data (because references on GPU are not valid for CPU) no expression templates technique support

All GPU functions receive GpuMat as input and output arguments. This allows to invoke several GPU algorithms without downloading data. GPU module API interface is also kept similar with CPU interface where possible. So developers who are familiar with Opencv on CPU could start using GPU straightaway.

Como se dijo en el capitulo previo, la funcion inicial de la gpu era renderizar imagenes a partir de escenas. Con el tiempo la generalizacion en las aplicaciones llevo a que se implementen funciones totalmente distintas. Por ej. mediante el modulo de gpu de la libreria OpenCV se estan realizando la funcion opuesta, que es entender las escenas a partir de imagenes. ((ver filmina 11 de la presentacion))

Capítulo 3

Trabajo experimental

3.1. Futbol robot

El trabajo consiste en un sistema de visión por computador para el reconocimiento de objetos en un partido de Futbol de robots. El futbol de robots se puede definir como una competición de tecnología robótica de avanzada en un espacio contenido Este trabajo se acota a una categoría particular de la competencia, en la cual los equipos se componen de 5 robots controlados por un sistema centralizado. Los robots se identifican por el color de sus 'camisetas'. Cada robot debe llevar el color designado para su equipo y puede llevar otros colores para identificar los robots dentro de un equipo. No puede tener el color del equipo contrario en ningun lugar de su camiseta. Ademas, ningun robot puede tener el color caracterísico de la pelota(naranja) en su camiseta. Todo el procesamiento se realiza desde un sistema central y no se permite la intervencion de humanos a menos que el juego este detenido. El sistema central dispone de las imágenes tomadas por una unica camara central situada sobre el campo de juego. Una forma de definir el sistema de control es dividir el problema en las siguientes áreas: Reconocimiento del campo: usando la imagen de la cámara, y posiblemente información anterior, se determina la posición, velocidad y orientación de los robots de ambos equipos y de la pelota. Planificación de las acciones de los robots: Se determina las acciones a tomar con objeto de lograr el objetivo de trasladar la pelota al objetivo. Control de los robots: Se usa un sistema de comunicación inalámbrico para mover los robots de acuerdo a la estrategia definida.

3.2. Detección en tiempo real

Dentro de las etapas definidas en la seccion anterior hay diversos procesos que requieren de tiempo: latencia de la camara(desde que se toma la imagen hasta que se comienza a procesar), latencia de la deteccion, latencia de definicion de estrategia, latencia de comunicación. El objetivo es que la toma de decisiones y la ejecucion de estas se realice en un tiempo donde la imagen sobre la cual se estan tomando las decisiones sea representativa del estado actual del campo. La realidad es que el sistema está en continuo cambio, aún cuando los tiempos de latencia sean muy chicos, por lo tanto se realizan 2 aproximaciones:

-Modificar los algoritmos de toma de decisiones para tener en cuenta que el sistema ha cambiado desde que se tiene la informacion, posiblemente usando informacion anterior. -disminuir lo mas posible la latencia en todos los pasos del procesamiento, de forma tal que la imagen sea lo mas actual posible.

En un sistema de tiempo real duro, el procesamiento solo sería válido si se completan todas las etapas antes de que se capture la proxima imágen, la cual invalida el estado descrito por la imagen anterior.

3.3. Implementacion existente

El trabajo esta centrado en el primer paso del procesamiento, la etapa de reconocimiento del campo. En este paso, se recibe una imágen actual del estado del campo y a partir de esta se deben detectar las posiciones de los robots de cada equipo y de la pelota.

Para realizar esto disponemos de una libreria que permite procesar frames capturados a partir del video de un juego. La libreria permite detectar los elementos de cada cuadro (robots y pelota) devolviendo la información relevante.

El proceso de detección no se va a detallar nuevamente, se puede encontrar en detalle en Ref. [1, capitulo 5] En esta sección solo se verá en forma general los pasos y se detallan brevemente las funcionalidades que seran optimizadas usando gpu.

3.3.1. La libreria bottracker

La clase central de la libreria es bot_tracker [1]. Esta clase necesita ser instanciada y configurada con los parametros necesarios(imagen de background, colores, etc) para poder realizar todo el proceso de detección.

3.4. Implementaciones sobre GPU

Como se explicó previamente, una de las aproximaciones para que el procesamiento de imágenes se adapte a la realidad cambiante del juego de futbol es disminuir la latencia en todos los pasos que involucra este procesamiento. La aceleración de esta etapa mediante el uso de GPU tiene este objetivo. Si la optimizacion lograda mediante el uso de funciones implementadas sobre GPU no supera el tiempo extra para transferir los datos hacia/desde la memoria GPU, entonces se reducirá el tiempo total requerido para esta etapa y las decisiones que se tomaran a continuación estarán basadas en información mas actual.

La idea de esta sección es ir planteando modificaciones individuales en el algoritmo implementado usando funciones sobre CPU. El objetivo de esto es tener un conjunto de implementaciones distintas que implementen funciones independientes sobre GPU, ejecutando el resto sobre CPU. De esta forma se puede realizar una analisis en funcion de la operacion que se esta optimizando y no del contexto en el cual ocurre. Finalmente se plantea una version donde se implementan sobre GPU todas las funciones posibles (version mas optimizada) y es la que se usa para comparar la optimizacion lograda en el contexto de la deteccion de imagenes para futbol de robot.

3.4.1. Algoritmos

En las secciones anteriores se describio el codigo para la deteccion de imagenes en el cual se basa este trabajo. Hay 3 operaciones de este proceso que han sido implementadas en el modulo gpu de OpenCV. Estas son: conversion a escala de grises, diferencia absoluta con el fondo y conversion de la diferencia a valores binarios. Repasemos en primer lugar que hacen estas 3 operaciones(visto en el capitulo de implementacion existente) y cuales son sus posiblidades de paralelizacion.

Conversion a escala de grises:

A este item nada...

Las operaciones de matrices como la conversion a escala de grises, la diferencia absoluta y la umbralizacion son operaciones que trabajan proce- sando pixeles de la imagen en forma independiente y son casos interesantes para optimizar usando paralelismo. Son del tipo de operaciones Embarras- singly parallel [13], donde practicamente el tiempo de procesamiento podrıa acelerarse en el orden de la cantidad de procesadores usados.

3.4.2. Sistema de pruebas

Para realizar las pruebas se reutilizo gran parte del programa cliente descrito en [1, capitulo 4].

3.4.3. Resultados

3.4.4. Hardware utilizado

Capítulo 4

Conclusiones y trabajo futuro

Bibliografía

[1] Ignacio Jaureguiberry. Reconocimiento de imágenes en fútbol de robots - informe de trabajo final para tiempo real, 2011.