Tesis Doctoral

Estimación de movimiento en secuencias de imágenes RGB y RGB-D

Gómez Fernández, Francisco Roberto

2016-03-30

Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Estimación de movimiento en secuencias de imágenes RGB y RGB-D

Tesis presentada para optar por el título de Doctor de la Universidad de Buenos Aires en el área de Ciencias de la Computación

Francisco Roberto Gómez Fernández

Directores de tesis:	Marta E. Mejail
	Álvaro D. Pardo Piccone
Consejera de estudios:	Marta E. Mejail

Fecha de defensa: 30/03/2016

Buenos Aires, 2016

Estimación de movimiento en secuencias de imágenes RGB y RGB-D

Resumen. El movimiento es una característica fundamental para el procesamiento de video y sus posteriores aplicaciones. La estimación de movimiento en video es de gran utilidad para definir la correspondencia de puntos en una escena, calcular sus velocidades y así poder discriminar objetos, acciones, segmentar movimiento, etc.

El objetivo de este trabajo es realizar un seguimiento preciso y una estimación de movimiento de un gran conjunto de puntos. Esto se conoce como estimación densa de movimiento.

Para ello, se proponen dos líneas principales de estudio: modelos estadísticos de movimiento utilizando texturas dinámicas y el cálculo del flujo óptico minimizando la energía con *graph cuts*, en ambos casos considerando secuencias de imágenes RGB y RGB-D.

El modelo de texturas dinámicas está muy bien condicionado para la segmentación de movimiento, y dentro de este contexto desarrollamos una aplicación con características novedosas: (i) proceso de aprendizaje desacoplado y (ii) algoritmos optimizados para trabajar en placas gráficas *GPU (Graphic Process Unit)*. Además, el modelo ha sido extendido para contemplar secuencias de imágenes RGB-D, el cual no había sido estudiado hasta el momento, permitiéndonos identificar procesos visuales en 3D.

Experimentos sobre la base de datos *DynTex* muestran resultados exitosos de performance y de clasificación para la mayoría de las casos. Luego, nuestros análisis sobre secuencias RGB-D revelan la viabilidad de este modelo para aplicaciones 3D.

El problema de la estimación del flujo óptico (*optical flow*) fue abordado mediante la minimización de la energía del campo de vectores utilizando la técnica de *graph cuts* con una formulación novedosa de la energía. Ampliamos esta formulación para tener en cuenta la profundidad y así calcular el

flujo de la escena (*scene flow*). Hasta donde sabemos, en la literatura, nunca se había utilizado *graph cuts* para estimar el *scene flow*. Los resultados obtenidos sobre el dataset *Middlebury* muestran que nuestros algoritmos son competitivos comparados con los presentes en el estado del arte y los mejores con en términos de error angular.

Palabras clave: estimación de movimiento, texturas dinámicas, optical flow, scene flow, graph cuts

Motion estimation in RGB and RGB-D image sequences

Abstract. Motion is a fundamental cue for video processing and its further applications. Video motion estimation is very useful to find correspondences of points in a scene, computing their velocities, discriminate objects, actions, segment motion, etc.

The aim of this work is to accurately track the motion of a large set of points in videos. This is known as dense motion estimation. To this end, two main lines of study were proposed: statistical models of motion using dynamic textures and optical flow calculation using graph cuts for energy minimization, considering in both cases sequences of RGB and RGB-D images.

The dynamic textures model is well suited for motion segmentation, and in this context we develop an application with novel features: (i) a decoupled learning step (ii) GPU-translated algorithms optimized to work on GPU (Graphic Process Unit). Also, the model has been extended to process RGB-D sequences, which had not been studied so far, allowing us to identify visual processes in 3D. Experiments on the *Dyntex* dataset show successful results of performance and classification for most cases. Then our analysis of RGB-D sequences reveal the viability of this model for 3D applications.

The problem of optical flow estimation was addressed by minimizing the energy of the vector field using the graph cuts method with a novel formulation of energy. We extend this formulation to take depth into account and thus estimate the Scene Flow. To the best of our knowledge, scene flow estimation using graph cuts has never been used in the literature. The results obtained on the *Middlebury* dataset show that our algorithms are competitive with the state of the art and the best performing in terms of angular error.

Keywords: motion estimation, dynamic textures, optical flow, scene flow, graph cuts

Agradecimientos

En primer lugar quiero agradecer a Marta que sin ella no hubiera sido posible llegar hasta este lugar, y no solamente por haber sido mi directora sino por todo el apoyo incondicional que siempre me brindó en todo momento, ya sea personal, espiritual o académico. También, en estos años de doctorado tuve la suerte de tener a Álvaro como co-director: una gran persona e investigador que me abrió las puertas de su casa y me hizo querer aún más al país hermano y vecino, Uruguay.

Por supuesto que debo agradecer a la familia, a mi madre, padre y hermanos. Con ellos me siento más que agradecido y no solo por haberme acompañado durante el doctorado sino también porque me dieron la vida que tengo y la suerte de estar a su lado. A Mita por su amor incondicional y por estar siempre y darme hasta lo que no tiene. A Papo por sus consejos, su sabiduría, contención y palabra de aliento en el momento justo. A Pepo y Lula, mis hermanos del alma, las personas más importantes de mi vida, por compartir tanto y estar siempre, por su excepcionalidad como personas. A mi hermano Pepo, por su fuerza y empuje, su apoyo incondicional, su compañerismo y presencia, por ser mi ejemplo en muchos aspectos de mi vida y por su gran pasión y amor. A mi hermana Lula, por su entereza y sensibilidad, por ser mi guía académica y docente, por ser un almita sensible y dedicada a sus hermanos. A Labuela que me grita ”vamos picho”desde el cielo brindando siempre su empuje y fuerza para seguir adelante y pelear por lo que uno cree. A Matiz por su gran amor y generosidad que siempre la caracterizó. A Bingen que también me mira desde el cielo y al cual debo

mi apodo ”Patxi”que hoy, ya más que un apodo, es mi nombre e identidad.

También, no puedo dejar de mencionar:

A mi amigos, El Pocho, Fran, Guille y el Marino con los cuales compartí muchísimos gratos e inolvidables momentos, tanto en la facultad, en las pocho/pachi/fran-oficinas como en los asados, reuniones y salidas.

A mi querido amigo Noro por su calidez y alegría como persona, y por su guía y sus consejos durante el doctorado. A mis amigos de imágenes Juanete, Seba, Mota y Martín por compartir tantos momentos y estar siempre conmigo en estos años de doctorado.

A Mailu por su apoyo, acompañamiento y por haber compartido tanto tiempo y tantas cosas.

Al Departamento de Computación y a la Universidad de Buenos Aires por haberme brindado las herramientas para poder desempeñarme en la vida académica. Al Grupo de Procesamiento de Imágenes y Visión por Computadora, al Laboratorio de Robótica y principalmente a toda la gente que lo compone desde el año 2009 cuando incursioné en la investigación: Dani, Julio, M. Elena y Mariano Tepper.

A los pasante de imágenes Samy, Saúl e Isma, que en el poco tiempo que compartí con cada uno de ellos llegué a apreciarlos como grandes personas que son.

A mis tesistas de licenciatura, Nadia, Mariano y Esteban.

Al CONICET y al Ministerio de Ciencia y Técnica por haberme dado la oportunidad de realizar un doctorado y formarme durante todo este tiempo, apostando al futuro, a la ciencia, a la educación y al desarrollo del país, que hicieron posible los ex presidentes Nestor y Cristina Kirchner mediante sus políticas de ciencia y técnica.

Contents

Chapter 1

Introduction

Motion estimation in video is an essential task with regard to the extraction of relevant information and subsequent analysis of its contents. Object detection and segmentation are greatly benefited if a good motion estimation is available.

The main goal of motion estimation is to have a good point correspondence between consecutive images of a video in order to discriminate and understand scene contents and the movement of their objects.

1.1 Motivation

The *real* motion of a scene is not always observable, instead what we are seeing is the "apparent" motion, a projection of a smooth and well captured by the camera. For example, in Figure 1.1, all 3D displacements from $\mathbf{P_t}$ to $\mathbf{P'_t}$ falling between the dotted lines have the same displacement in the 2D plane image.

In order to formalize methods and algorithms for motion estimation, the following assumptions about the scenes and images acquired are typically considered:

- color/brightness constancy: intensity of corresponding pixels do not change across time
- coherence: pixels on the same surface share similar move.

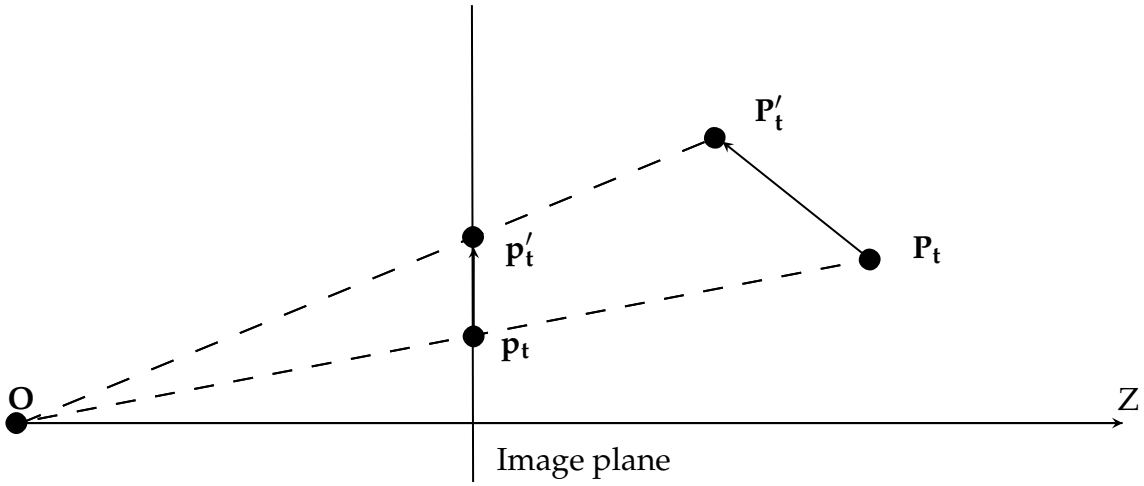


Figure 1.1: Example of apparent motion.

- "smooth" and continuous movements

The most common methods in the literature employ windowed analysis to deal with the complexity of the different motions in the scenes where both fast and slow motions compete. This brings two major problems (see Figure 1.2):

- Aperture (O): small windows capture small movements, but large windows do not meet the assumption of coherence.
- Adhesion (A): a window covers objects in different surfaces and depths.

Another recurring and inevitable problem in motion estimation is **occlusion**. Occlusion occurs when an object is moving in the scene and then no longer appears in the camera view. This can happen because another object is occluding it or the same object disappears from the field of view of the camera.

The most basic methods on motion estimation are based on difference between consecutive images. The most popular example is *Motion templates* [BD96, BD02]. While these methods heavily used the assumptions above, they do not apply block analysis and therefore only capture large and very distinct movements. However, they achieve high speed performance and are used in human action recognition.

Figure 1.2: Aperture (O) and adherence (A) problems.

Most of the works on motion estimation rest on the assumption of the optical flow [LK81, HS81] constraints, which postulate that the intensity of a point along their movement remains constant. The precise motion tracking for a large set of points is a task that has been addressed many times [BBPW04, BM11] and is still object of study of the scientific community.

The simplest way, and one of the oldest, is to identify the optical flow, i.e. the apparent field of instant velocity by the "local" solution of Lucas-Kanade [LK81] consisting to assume that the field is locally constant and estimate it at each point by minimizing a local mean optical flow equation. Later, solutions were proposed as "global", with the dual aim of relaxing the restriction of optical flow fields and extracting more complex motions. The best known is due to Horn and Schunk [HS81]. Other variants of this model have been proposed. We can mention the work of Weickert, Bruhn et Schnorr [BWS05] approach combining "local" Lucas-Kanade with the "comprehensive approach" of Horn-Shunck. The above-described methods have the difficulty of determining the motion in presence of occlusions. When using motion information for applications, such as segmentation and tracking, ignoring occlusions can downgrade the overall application performance.

In video object tracking the position of an object of interest in a sequence, we try to find it on the following frames. In some applications it is enough to mark a point within the reference object or define rectangular regions where you can find the object in motion. This is an area that widely benefits from the motion estimation methods and hence its importance.

Istepanian et al. [Ist03] presented an algorithm that uses edge detection and mathematical morphology tools to follow a contour previously marked in a sequence of images. Detected edges are projected to the next frame and along the edges of the box are used to detect the new position of contour. In those presented sequences of medium complexity, the results are acceptable. Another work, presented by Kim et al. [KH02], use tracking of edges as the main feature for the automatic detection of moving objects. Also, in [MYN07]a method of object tracking is based on graph cuts segmentation [BJ01, JB06, KT05] is proposed. The ability to perform real-time monitoring provides an invaluable tool for developing augmented reality applications, in order to score the environment with virtual content adding markers and labels for objects detected over consecutive frames.

As it was mentioned before, traditional methods of optical flow estimation fail when the movement of objects in the scene is large, either by a moving camera or the objects themself. On the other hand, there are methods that seek to establish specific mappings using local descriptors such as SIFT (Scale Invariant Feature Transform) [Low99]. These methods have two major problems, the first is that the number of corresponding pixels is usually low and the second is that since no type of regularization is imposed outliers occur. In short, it is not easy on the basis of this information to calculate an estimate of dense optical flow. Following this line, Brox et al. [BBM09] propose a method that combines start correspondence regions as an initial condition and obtains a dense optical flow using variational calculation techniques.

Dense motion tracking allows us to draw a much finer level of granularity compared with the methods that try to search scatter correspondences, which allows for applications like *structure-from-motion*, motion segmentation, action recognition in sports, or automatic video summary and more.

Another applications could be: interpolation between frames, restoration of corrupted frames, format conversion, slow motion, tracking, etc.

Video processing requires more computational performance than still images, and a greater amount of information to process. These facts generally involve high quality methods of motion estimation with high computational costs. A solution to these problems is the implementation of algorithms in graphic cards using techniques of parallel processing on the GPU (*Graphic Process Unit*). This enables video processing in real time, with the additional advantage that it can be done in low-cost PCs. This is also an area of growing interest in the scientific community, both in regard to image processing and scientific computation in general.

The *visual processes* in videos can be characterized using statistical models that describe the behavior of each pixel over time. In general, the statistical models are computationally expensive and therefore efficient implementations that make use of modern hardware and GPGPU are necessary.

The *dynamics textures* [DCWS03] model provides a different approach than the optical flow problem. This approach postulates a mathematical model of what a video and motion contained in it. In particular we are interested in the Mixture of Dynamic Textures model [CV08].

In this thesis we approach the study of video motion analysis with two main lines of research: statistical models of motion using dynamic textures and optical flow and scene flow calculation minimizing the energy with *graph cuts*.

If the object of analysis are RGB-D sequences of video, where in addition to the 2D information the camera also captures the 3D information (depth), motion estimation becomes even more challenging. The optical flow in 3D becomes *Scene Flow* [WBV+11]. It is what is needed to get the information to calculate three-dimensional movement in this type of videos.

Feature detection in an image is a very important process and in many cases, can determine the success or failure of a whole system of image processing or video. A feature can be described as an interesting part of an object within an image, which gives distinctiveness, i.e., which serves to

distinguish it from its peers. The study of above methods and descriptors are a challenging task and still do not have a close solution in the state of start knowledge [TH98, BTVG06, CLO+12].

The main objective is to achieve a dense motion estimation and still be able to deal with missing information and large displacements, problems that often occur in real-world sequences.

1.2 RGB-D sensors and applications

In this section, we show the importance of using RGB-D sensors for applications in computer vision.

RGB-D sensors are composed of an RGB camera that captures color information of the scene, and a depth sensor which provides information about the distance from the camera to the objects in the scene.

Although the use of depth sensors and the fusion of color and depth information have been around for many years, the recent introduction of simple and affordable RGB-D sensors, such as the Microsoft Kinect[1], the Asus Xtion Live[2], etc., opened new research opportunities in the computer vision community. Commonly RGB-D cameras use two sensors, a RGB camera and a pair of emitter and structured light and infrared receiver which are used to estimate depth information (see Figure 1.3). To obtain an object's depth, a known infrared light pattern is first projected onto the scene. Then, the infrared camera captures this projected pattern, which will be deformed by the scene geometry. Finally, using the difference between these patterns, distances are estimated using structured light algorithms. Depth information used in conjunction with color information (RGB) allows 3D scene reconstruction. The calibration parameters for the RGB-D sensor can be used to estimate each pixel's position in the 3D scene, forming what is called a point cloud, where each point contains depth and color information. Figure 1.4 shows examples of RGB-D images and its reconstructed point cloud.

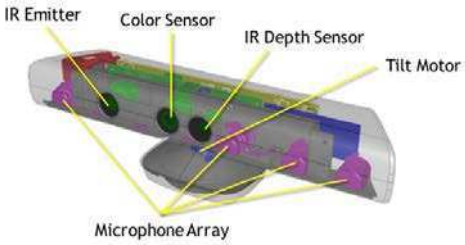
The RGB-D sensors output synchronized color and depth information

[1] http://www.xbox.com/en-us/kinect

[2] http://www.asus.com/Multimedia/Xtion

where for each pixel we can access color and depth information. For instance, the Kinect sensor acquires RGB-D images with a default resolution of 640x480 pixels at rate of 30 fps with 8 bits of depth for RGB and 11 for depth, giving 2048 different levels of sensitivity. The distance range varies between 1.2 and 3.5 m, with the possibility of being extended to the 0.7 to 6 m range. It can give images up to 1280x1024 pixels but at a smaller frame rate.

(a) RGB-D cameras. Asus Xtion Live (above), Microsoft Kinect (below).



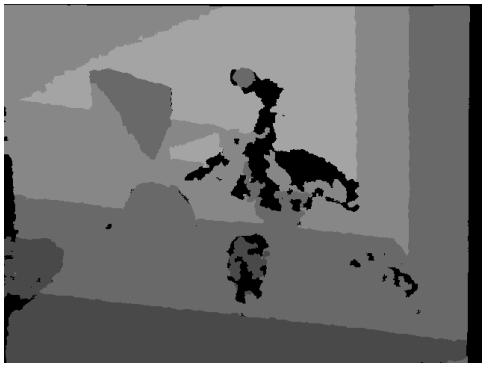
(b) Component diagram of a Microsoft Kinect.

Figure 1.3: Commercial RGB-D sensors using structured lighting.

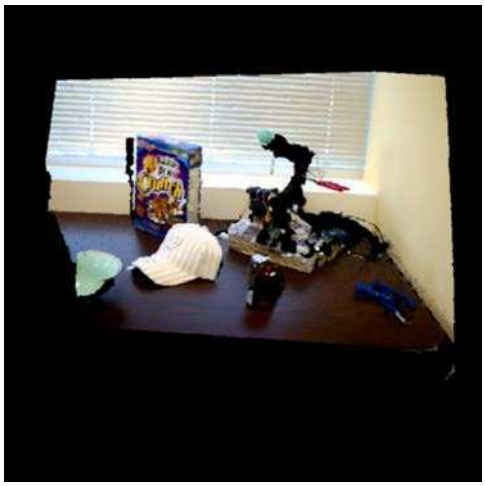
Object recognition and tracking

In [BHF+15] an evaluation of RGB-D and 3D descriptors in the context of Object Recognition and Object Tracking is presented. 3D point clouds are used instead of RGB and depth images independently. Working directly with 3D point clouds is a recent trend in computer vision; traditionally applications build polygonal meshes and then discard point cloud data. Also, local descriptors (CSHOT [TSDS10], Spin-Images [JH99] and ECV context [BKK+13]) are evaluated for object recognition because they are more robust to noise, clutter and occlusion. Once an object is recognized in a target frame, an Iterative Closest Point (ICP) [Zha94, BM92] based tracking is performed to track the object frame by frame. Results show that shape is the most important cue for accurately match RGB-D descriptors for object recognition. However, texture helps discrimination when objects are large or have little structure. Analogously, when tracking objects depth information complements RGB and improves the overall system accuracy. These

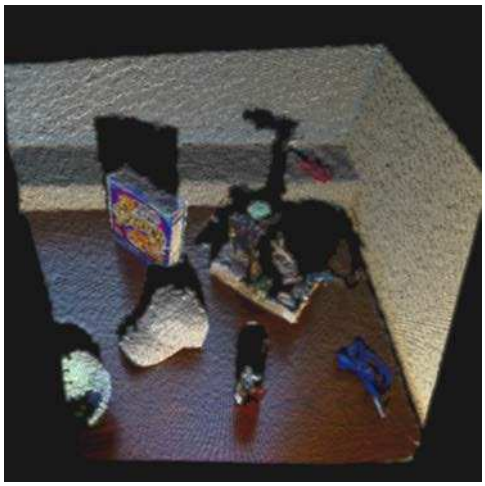
(a) Color RGB image.



(b) Depth image.



(c) Reconstructed 3D point cloud from (a) and (b).



(d) An upper view of the point cloud in (c).

Figure 1.4: (a) and (b) Examples of a RGB-D image captured using an Asus Xtion Live sensor. In (b) Lighter colors are far from the sensor and black is due to an unknown measure. (c) and (d) Views of the reconstructed 3D point cloud from the RGB-D image in (a) and (b) using the intrinsics parameters of the camera.

facts show the potential of using RGB-D descriptors in computer vision applications. Figure 1.5 shows the recognition of a soda can in a target frame and tracked subsequently in the following frames.

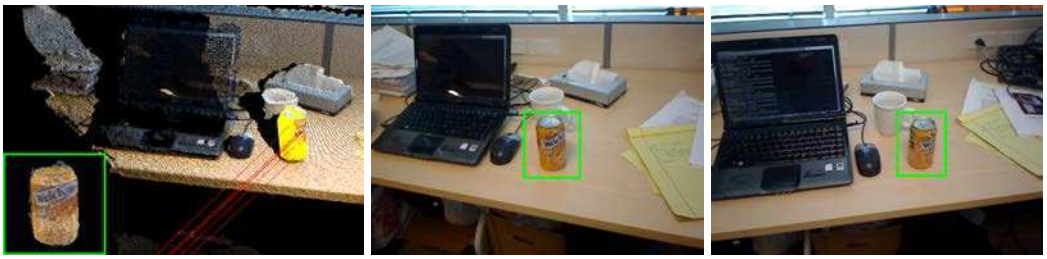


Figure 1.5: Two apps

Automatic Camera-Screen Localization

Knowing the location of the TV screen with respect to a camera it is important for many applications. For instance, in Human-computer interaction [CNM83], it is critical to know whether a user is paying attention to the screen or not. This requires the knowledge of the screen position with respect to the camera. This work addresses this problem in a configuration where there are people looking at the TV and a RGB-D camera facing them, located near the TV screen [FLPM14]. Therefore, it is assumed that a person is looking in the direction of their nose and use head pose as a coarse indication of gaze [MCT09]. The proposed method [FLPM14] automatically estimates the screen location and camera rotation using only people's head pose obtained from a Face Tracking analysis [CGZZ10] on the RGB-D video. It is noted that with more temporal information, i.e. longer sequences, better estimations are obtained. Figure 1.6 shows three subjects looking at the TV with their associated view directions. The screen location is inferred using the intersection of people's view directions during time.

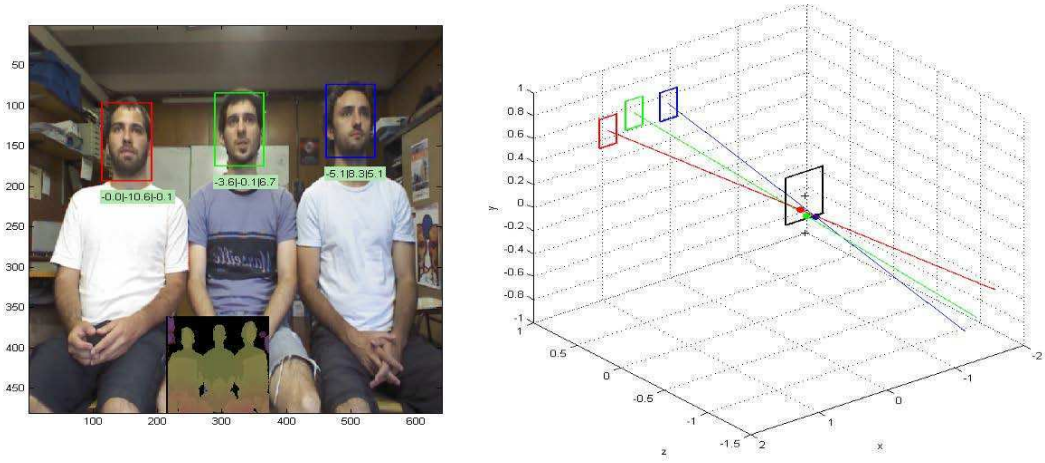


Figure 1.6: Left: A color image with its associated depth and people's head pose detected. Right: people's view-direction in a 3D coordinate system centered at the camera. The black square near the origin represents the screen. The units are meters for distances and degrees for rotation angles. Figure best viewed in colors.

1.3 Contributions

The main contributions of this thesis are:

- The method studied in Chapter 3, the Mixture of Dynamic Textures (MDT), introduces a decoupled learning process that allows a motion segmentation algorithm that can be trained off-line, leaving segmentation as an on-line process, thus achieving substantial improvements in performance.

- MDT algorithms are carried to the GPU and a specific implementation is employed for the case inversion of symmetric positive definite matrices, a repeated operation considered as a "bottleneck" in this type of statistical models, as regards of the computational performance.

- The MDT model is extended to contemplate RGB-D sequences which has not been studied so far, allowing identification of visual processes in 3D.

- A new formulation for the energy minimization in optical flow and scene flow is proposed which uses graph cuts optimization in order

to obtain a smooth flow field while preserving discontinuities in flow and account for occlusions.

- The scene flow problem is resolved in the 3D space using geometry consistency and brightness constancy over point clouds.
- Also, a detailed analysis of the impact of this energy minimization via graph cuts is also presented and how parameters affect in the resulting vector field.
- Finally, the analysis of these methods on RGB-D sequences is a contribution that result very useful for the scientific community because of the current rise in the use of low-cost RGB-D sensors.

1.4 Publications

The contributions mentioned in the previous section, add innovative contributions to the state of the art and have been included in publications related to the thesis.

Published:

- Gómez-Fernández, Francisco, Rodríguez, Juan Manuel, Buemi, María Elena, and Jacobo-Berlles, Julio. "Performance of Dynamic Textures Segmentation using GPU". Journal of Real-Time Image Processing. Springer-Verlag, 2013. ISSN: 1861-8219.
- Rodríguez, Juan Manuel, Gómez-Fernández, Francisco, Buemi, María Elena and Jacobo-Berlles, Julio. "Dynamic textures segmentation with GPU". In Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, pp. 607-614. Springer International Publishing, 2012.

Related:

- Bianchi, Mariano, Heredia, Nadia, Gómez-Fernández, Francisco, Pardo, Alvaro and Mejail, Marta. "Two Applications of RGB-D Descriptors in Computer Vision." In Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, pp. 236-244. Springer International Publishing, 2015.

- Gómez-Fernández, Francisco, Liu, Zicheng, Pardo, Alvaro and Mejail, Marta. "Automatic Camera-Screen Localization." In Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, pp. 588-595. Springer International Publishing, 2014.

Submitted (in evaluation):

- Gómez-Fernández, Francisco, Pardo, Álvaro and Mejail, Marta. "Scene Flow estimation using graph cuts". Pattern Recognition (ICPR), 2016 23rd International Conference on. IEEE, 2016.

1.5 Organization of the thesis

The rest of this thesis is organized as follows:

- In Chapter 2 we present all the necessary background on motion estimation in order to introduce the main principles treated in the other chapters. We begin with a review of the camera model and the motion estimation in 2D and 3D. Then, related works and salient state of the art papers are discussed that are mentioned later in the following chapters.

- Chapter 3 tackles the study of Dynamic Textures to motion estimation and segmentation with emphasis in applications and computer performance. Also, an extension to 3D using RGB-D images is approached.

- Motion estimation techniques that capture rigid and non-rigid body motions, such as optical flows and its 3D counterpart scene flow are developed in Chapter 4. Graph cuts optimization is used for both methods in order to obtain an accurate motion field imposing global

smoothness through a discontinuity preserving function in order to avoid oversmoothing at object's boundaries.

- Finally, in Chapter 5 we summarizes our main conclusions and future work.

Capítulo 1

Introducción

La estimación de movimiento en video es una tarea fundamental en lo que respecta a la extracción de información y pistas relevantes para el posterior análisis de su contenido. La detección y segmentación de objetos se ven ampliamente beneficiadas si se cuenta con una buena estimación de movimiento previa.

En la estimación de movimiento en video se tiene por objetivo obtener una buena correspondencia de puntos entre imágenes consecutivas de un video y así lograr la discriminación de los objetos según su movimiento.

1.1 Motivación

El movimiento real no es siempre observable, en cambio lo que vemos es el movimiento "aparente", una proyección de un movimiento que se supone rígido, suave y bien capturado por la cámara. Por ejemplo, en la Figura 1.1, todos los desplazamientos 3D de $\mathbf{P_t}$ a $\mathbf{P'_t}$ que caen entre las lineas punteadas tienen el mismo desplazamiento 2D en el plano imagen.

En general, para poder emplear métodos matemáticos y algoritmos de estimación de movimiento entre pares de imágenes, se establecen los siguientes supuesto sobre las escenas y las imágenes adquiridas:

- Color/brillo del píxel constante
- Los píxeles en una misma superficie se mueven de forma similar, coherentemente.

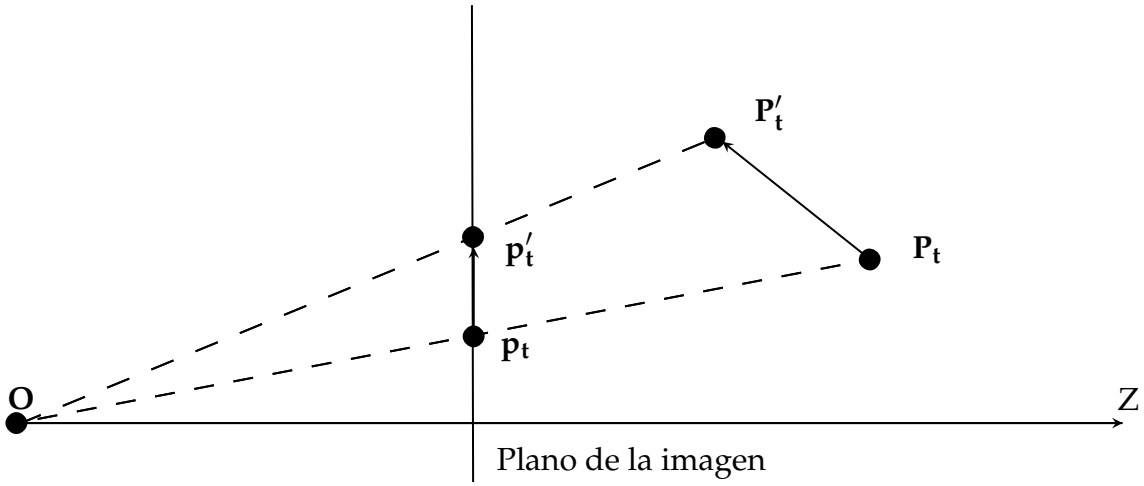


Figura 1.1: Ejemplo de movimiento aparente.

- Movimientos "suaves" y continuos

Los métodos más comunes emplean análisis por bloques o ventanas para lidiar con la complejidad de los distintos movimientos en las escenas, donde conviven tanto movimientos rápidos y lentos. Esto trae aparejado dos grandes problemas (ver Figura 1.2):

- Apertura (O): ventanas pequeñas capturan movimientos pequeños, pero ventanas grandes no cumplen el supuesto de coherencia
- Adherencia (A): una ventana abarca objetos en distintas superficies y profundidades

Otro problema recurrente e inevitable en la estimación de movimiento es la **oclusión**. La oclusión se produce cuando un objeto que se está moviendo en la escena deja de aparecer de la visión de la cámara. Esto puede ocurrir debido a que otro objeto se interpuso o el mismo objeto desapareció del rango de la cámara.

Los métodos de estimación de movimiento más básicos se basan, se basan en diferencia entre imágenes consecutivas; el ejemplo más popular es el de *Motion templates* [BD96, BD02]. Si bien estos métodos utilizan fuertemente los supuestos mencionados anteriormente, no aplican análisis por bloques

Figura 1.2: Problemas locales: apertura (O) y adherencia (A).

y por ende solo capturan movimientos grandes y muy marcadas. Sin embargo, se destacan por su gran velocidad de ejecución y suelen emplearse en tareas de reconocimiento de acciones humanas.

La casi totalidad de los trabajos sobre estimación de movimiento reposa sobre la hipótesis de restricción del flujo óptico u *optical flow* [LK81, HS81], que postula que la intensidad de un punto al que seguimos a lo largo de su movimiento permanece constante. El seguimiento preciso del movimiento para un gran conjunto de puntos es una tarea que ha sido abordada en muchas ocasiones [BBPW04, BM11] y que aún es objetivo de estudio de la comunidad científica.

La manera más sencilla, y una de las más antiguas, de poder identificar el flujo óptico, es decir, el campo de velocidades aparente instantáneo, es mediante la solución "local" de Lucas-Kanade [LK81] que consiste en suponer que el campo es localmente constante y estimarlo en cada punto mediante la minimización de una media local de la ecuación de flujo óptico. Más tarde se propusieron soluciones "globales", con el doble objetivo de relajar la restricción de flujo óptico y de extraer campos de velocidades más complejos. La más conocida es debida a B.K. Horn y B.G. Schunk [HS81]. Diferentes variantes de este modelo han sido propuestas. Podemos mencionar

los trabajos de Weickert, Bruhn et Schnorr [BWS05] que combinan el enfoque "local" de Lucas-Kanade con el enfoque "global" de Horn-Shunck. Los métodos antes descriptos tienen dificultades para determinar el movimiento si hay oclusiones. En caso de utilizar información de movimiento para segmentación y seguimiento este tipo de problemas se puede propagar al resultado si no se tiene en cuenta y deteriora los resultados.

El seguimiento de objetos de video implica dada la posición del objeto de interés en un cuadro de la secuencia encontrarlo en cuadros siguientes u anteriores de la misma. En algunas aplicaciones alcanza con marcar un punto dentro del objeto como referencia o con definir regiones rectangulares donde se puede encontrar el objeto en movimiento. Ésta es una área que se beneficia ampliamente de los métodos de estimación de movimiento y de aquí la importancia del mismo.

En [Ist03] se presentó un algoritmo que utiliza detección de bordes y herramientas de matemática morfológica para seguir un contorno previamente marcado en una secuencia de imágenes. Los bordes detectados son proyectados al cuadro siguiente y junto con los bordes de este cuadro son usados para detectar la nueva posición del contorno. Para las secuencias presentadas, de complejidad media, los resultados son buenos. En [KH02] se presenta otro trabajo que usa el seguimiento de bordes como característica principal para la detección automática de objetos en movimiento. También, en [MYN07] se presenta un método de seguimiento de objetos basado en el concepto de segmentación utilizando Graph Cut [BJ01, JB06, KT05]. La posibilidad de realizar seguimiento en tiempo real provee una invaluable herramienta para desarrollar aplicaciones de realidad aumentada, con el objetivo de anotar el entorno con contenido virtual agregando marcadores y etiquetas para los objetos detectados a lo largo de frames consecutivos.

Como ya vimos los métodos tradicionales de estimación de Flujo Óptico fallan cuando los desplazamientos de los objetos en la escena son grandes, ya sea por movimiento de los propios objetos o la cámara. Las estrategias de cómputo coarse-to-fine si bien logran capturar mayores movimientos nunca sobrepasan unos pocos píxeles [BBM09]. Por otra parte existen métodos que buscan establecer correspondencias puntuales mediante el uso de

descriptores locales, como por ejemplo SIFT (Scale Invariant Feature Transform) [Low99]. Estos métodos tienen dos problemas principales, el primero es que la cantidad de píxeles correspondientes es normalmente bajo y el segundo es que dado que no se impone ningún tipo de regularización se producen outliers. En resumen, no es sencillo en base a esta información poder calcular una estimación de flujo óptico denso. En [BBM09] los autores siguiendo esta línea proponen un método que combina la puesta en correspondencia de regiones como condición inicial y la obtención de un flujo óptico denso mediante técnicas de cálculo variacional.

El seguimiento de movimiento denso nos permite extraer información a un nivel de granularidad mucho más fino comparado con los métodos de búsquedas de correspondencias esparzas, lo que permite obtener aplicaciones como *structure-from-motion*, segmentación de movimiento, reconocimiento de actividades en deportes, o resumen automático de video, entre otras. Las aplicaciones podrían ser interpolación entre frames, reparación de partes perdidas en los frames, conversión de formatos, cámara lenta, tracking, etc.

El procesamiento de videos requiere más recursos computacionales que el análisis de imágenes estáticas, y una mayor cantidad de información para procesar. Estos hechos implican, generalmente, que los métodos de estimación de movimiento de alta calidad sean bastante lentos y que requieran de la adquisición de costosos clusters de computadoras para acelerarlos. Una solución a estos problemas es la implementación de los algoritmos en placas gráficas utilizando técnicas de procesamiento paralelo en la GPU (*Graphic Process Unit*). Esto permite realizar el procesamiento de video en tiempo real, con la ventaja adicional de que puede realizarse en computadoras personales de bajo costo. Ésta también es un área de creciente interés en la comunidad científica, tanto en lo que refiere a procesamiento de imágenes como en el cómputo científico en general.

Los *procesos visuales* en videos pueden ser caracterizados utilizando modelo estadísticos que describen el comportamiento de cada píxel a lo largo del tiempo. En general los modelos estadístico son caros computacionalmente y por lo tanto, implementaciones eficientes que hagan uso de hard-

ware moderno como las GPGPU son necesarias.

Las Texturas Dinámicas [DCWS03] proveen un enfoque distinto al del flujo óptico. Este enfoque postula un modelo matemático de lo que es un video y el movimiento que contiene en el mismo. En particular nos interesa el modelo de Mixturas de Texturas Dinámicas [CV08].

En este trabajo enfocamos el estudio del análisis de movimiento en video con dos líneas principales de investigación: los modelos estadísticos de movimiento usando Texturas Dinámicas y el cálculo de flujo óptico utilizando descriptores y minimizando la energía con *graph cuts*.

Por otro lado, si el objeto de análisis son secuencias de video RGB-D, donde además de la información 2D de una cámara RGB también se posee la información tridimensional capturada por un sensor de profundidad, la estimación de movimiento se vuelve aún más desafiante. En este sentido, el flujo óptico de la escena o *scene flow* [WBV+11], es lo que se necesita calcular para obtener la información de movimiento tridimensional en este tipo de videos.

La detección de características en una imagen es un proceso muy importante que en muchos casos, puede determinar el éxito o fracaso de todo de un sistema que utilice procesamiento de imágenes o video. Una característica puede describirse como una parte interesante de un objeto dentro de una imagen, que le da cualidad distintiva, es decir, que sirve para distinguirlo de sus semejantes. El cómputo y la descripción de estos puntos característicos antes mencionados es una tarea desafiante y que todavía no posee una solución cerrada en el estado del arte del conocimiento [TH98, BTVG06, CLO+12].

El objetivo principal es lograr una estimación densa del movimiento y a la vez ser capaces de lidiar con información faltante y grandes desplazamientos, problemas que ocurren con frecuencia en secuencias capturadas del mundo real.

1.2 Sensores RGB-D y aplicaciones

En esta sección, se muestra la importancia de utilizar sensores RGB-D para aplicaciones de visión por computadora.

Los sensores RGB-D se componen de una cámara RGB que captura la información de color de la escena, y un sensor de profundidad, que proporciona información de la distancia desde la cámara a los objetos en la escena.

Aunque el uso de sensores de profundidad y la fusión de información de color y profundidad han existido desde hace muchos años, la reciente introducción de sensores RGB-D sencillos y asequibles, como el Microsoft Kinect[1], el Asus Xtion Live[2], etc., han abierto nuevas oportunidades de investigación en la comunidad de visión por computadora. Los dispositivos RGB-D más comunes utilizan dos sensores, por un lado una cámara RGB, y por el otro un un emisor de luz estructurada y una cámara infrarroja que se utilizan para estimar la información de profundidad (ver Figura 1.3). Para obtener la profundidad de un objeto, primero se proyecta sobre la escena un patrón conocido de luz infrarroja. Luego, la cámara de infrarrojos captura este patrón proyectado, que se deforma por la geometría de la escena. Por último, utilizando las diferencia entre estos patrones, las distancias se calculan utilizando algoritmos de luz estructurada. La información de profundidad utilizada junto con la información de color (RGB) permite la reconstrucción de la escena en 3D. Los parámetros de calibración del sensor RGB-D se pueden utilizar para estimar la posición de cada píxel en la escena 3D, formando lo que se llama una nube de puntos, donde cada punto contiene información de profundidad y de color. En la Figura 1.4 se muestran ejemplos de imágenes RGB-D y su nube de puntos reconstruida.

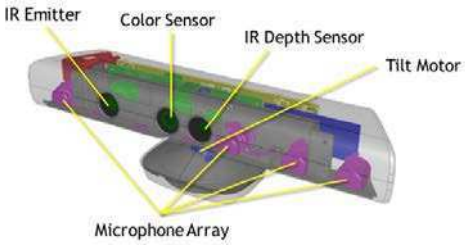
Los sensores RGB-D producen información de profundidad y de color de forma sincronizada. Por ejemplo, el sensor Kinect adquiere imágenes RGB-D con una resolución de 640x480 píxeles por defecto a una velocidad de 30 cuadros por segundo con 8 bits de profundidad para RGB y 11 bits de profundidad, dando 256 valore de color por canal RGB y 2048 diferentes

[1] http://www.xbox.com/en-us/kinect

[2] http://www.asus.com/Multimedia/Xtion

niveles de sensibilidad de profundidad. El rango de alcance varía entre 1,2 y 3,5 m, con la posibilidad de ser extendido al rango de 0,7 a 6 m. También se pueden obtener imágenes de hasta 1280x1024 píxeles, pero a una velocidad menor.

(a) Cámaras RGB-D. Asus Xtion Live (arriba), Microsoft Kinect (abajo).



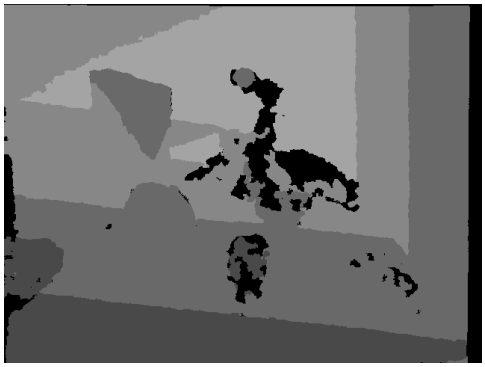
(b) Diagrama de componentes del sensor Microsoft Kinect.

Figura 1.3: Sensores RGB-D comerciales que usan iluminación estructurada.

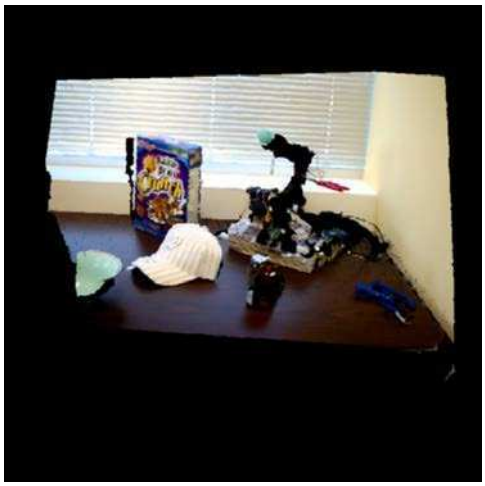
Reconocimiento y seguimiento de objetos

En [BHF+15] se presenta una evaluación de descriptores RGB-D y 3D en el contexto de reconocimiento de objetos y de seguimiento de objetos. Se utilizan nubes de puntos 3D en lugar de las imágenes RGB y de profundidad de forma independiente. Trabajar directamente con las nubes de puntos 3D es una tendencia reciente en visión por computadora; tradicionalmente, las aplicaciones de construyen mallas poligonales y luego descartan la información de nubes de puntos. Además, los descriptores locales (CSHOT [TSDS10], Spin-Images [JH99] y ECV context [BKK+13]) se evalúan para el reconocimiento de objetos, ya que son más robustos al ruido, el desorden y la oclusión. Una vez que un objeto se reconoce en un cuadro de video RGB-D, se aplica un seguimiento basado en *Iterative Closest Point* (ICP) [Zha94, BM92] cuadro por cuadro. Los resultados muestran que la forma del objecto es la información más importante para establecer correspondencias precisas entre descriptores RGB-D en el reconocimiento de objetos. Sin embargo, la textura ayuda a la discriminación cuando los objetos son grandes o tienen poca estructura. Análogamente, cuando se hace el seguimiento de objetos, la información de profundidad complementa la de RGB y

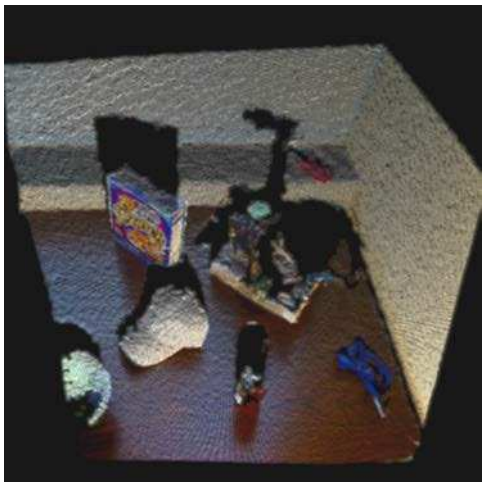
(a) Imagen de color RGB.



(b) Imagen profundidad.



(c) Nube de puntos 3D reconstruida a partir de (a) y (b)



(d) Una vista superior de la nube de puntos en (c).

Figura 1.4: (a) y (b) Ejemplos de una imagen RGB-D capturada utilizando un sensor Asus Xtion Live. En (b) Los colores más claros representan distancias que están más lejos del sensor y el color negro se debe a una medida de profundidad desconocida. (c) y (d) Vistas de la nube de puntos 3D reconstruida con la imagen RGB-D en (a) y (b) usando parámetros intrínsecos de la cámara.

aumenta así la fiabilidad del sistema. Estos hechos demuestran el potencial del uso de descriptores RGB-D en aplicaciones de visión por computadora. La Figura 1.5 muestra el reconocimiento de una lata de gaseosa en un cuadro de video RGB-D y seguido posteriormente en los cuadros subsiguientes.

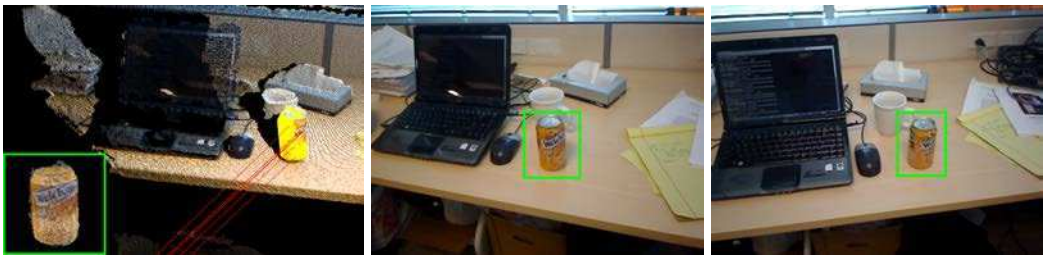


Figura 1.5: Izquierda: reconocimiento de la lata de gaseosa. Centro y derecha: dos imágenes de la secuencia en donde se hace el seguimiento de la lata de gaseosa.

Localización automática cámara-pantalla

Conocer la ubicación de la pantalla del televisor con respecto a una cámara, es importante en muchas aplicaciones de visión por computadora. Por ejemplo, en el campo de la interacción hombre-máquina (*human-computer interaction* [CNM83], es muy importante saber si un usuario está prestando atención a la pantalla o no. Esto requiere el conocimiento de la posición de la pantalla con respecto a la cámara. En este trabajo se aborda este problema en una configuración en la que hay personas que miran la televisión y hay una cámara RGB-D frente a ellos, situada cerca de la pantalla del televisor [FLPM14]. Por lo tanto, se supone que una persona está mirando en la dirección de su nariz y se usamos la pose de la cabeza como una indicación gruesa de su mirada. El método propuesto [FLPM14] calcula automáticamente la ubicación de la pantalla y la rotación de la cámara usando sólo la información de la pose de la cabeza de las personas obtenida a partir de un análisis de seguimiento de la cara [CGZZ10] en el vídeo RGB-D. Se observa que más información temporal, es decir, con secuencias más largas, se obtienen mejores estimaciones. La Figura 1.6 muestra tres personas que miran la televisión con sus direcciones de mirada asociadas. La ubicación de la pan-

talla se infiere utilizando la intersección de la las direcciones de las miradas de las personas durante el tiempo.

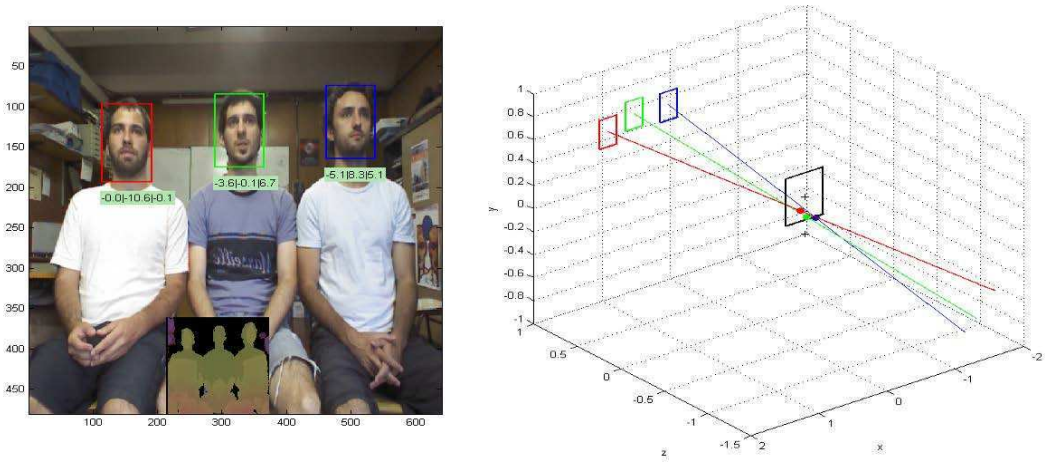


Figura 1.6: Izquierda: Una imagen en color con su profundidad asociada y la pose de la cabeza detectada en las personas. Derecha: estimación de la vista de las personas en 3D en marco de referencia centrado en la cámara. El cuadrado negro cerca del origen representa la pantalla. Las unidades son en metros para las distancias y en grados para los ángulos de rotación. Figura mejor apreciada en colores.

1.3 Aportes

Las principales contribuciones de esta tesis son:

- El método estudiado en el Capítulo 3, mixturas de texturas dinámicas (MDT), introduce un proceso de aprendizaje disociado que permite que un algoritmo de segmentación de movimiento que puede ser entrenado fuera de línea, dejando la segmentación como un proceso en línea, logrando así mejoras sustanciales de desempeño.
- Los algoritmos de MDT se portaron a la GPU y se realizó una implementación específica para el caso de la inversa de matrices simétricas definidas positivas, una operación considerada como un "cuello de botella" en este tipo de modelos estadísticos, en lo que respecta al rendimiento computacional.
- El modelo MDT se extendió para contemplar secuencias RGB-D, lo que permite la identificación de procesos visuales en 3D.
- Se propone una nueva formulación para la minimización de la energía en el flujo óptico y el flujo de la escena utilizando *graph cuts* con el fin de obtener un campo de vectores de movimiento suave, a la vez que se preservan discontinuidades en el flujo y se tienen en cuenta las oclusiones.
- El problema de flujo de la escena se resuelve en el espacio 3D usando consistencia geométrica y de brillo sobre las nubes de puntos.
- Por último, el análisis de estos métodos en secuencias RGB-D es una contribución que resulta de gran utilidad para la comunidad científica debido al creciente uso de sensores RGB-D de bajo costo.

1.4 Publicaciones

Las contribuciones mencionadas en el apartado anterior, poseen contribuciones innovadoras al estado del arte y han sido incluidos en publicaciones relacionadas con la tesis.

Publicados:

- Gómez-Fernández, Francisco, Rodríguez, Juan Manuel, Buemi, María Elena, and Jacobo-Berlles, Julio. "Performance of Dynamic Textures Segmentation using GPU". Journal of Real-Time Image Processing. Springer-Verlag, 2013. ISSN: 1861-8219.

- Rodríguez, Juan Manuel, Gómez-Fernández, Francisco, Buemi, María Elena and Jacobo-Berlles, Julio. "Dynamic textures segmentation with GPU". In Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, pp. 607-614. Springer International Publishing, 2012.

Relacionados:

- Bianchi, Mariano, Heredia, Nadia, Gómez-Fernández, Francisco, Pardo, Alvaro and Mejail, Marta. "Two Applications of RGB-D Descriptors in Computer Vision.Ïn Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, pp. 236-244. Springer International Publishing, 2015.

- Gómez-Fernández, Francisco, Liu, Zicheng, Pardo, Alvaro and Mejail, Marta. .Automatic Camera-Screen Localization.Ïn Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, pp. 588-595. Springer International Publishing, 2014.

Presentado (en evaluación):

- Gómez-Fernández, Francisco, Pardo, Álvaro and Mejail, Marta. "Scene Flow estimation using graph cuts". Pattern Recognition (ICPR), 2016 23rd International Conference on. IEEE, 2016.

1.5 Organización de la tesis

El resto de esta tesis se organiza de la siguiente manera:

- En el Capítulo 2 presentamos los antecedentes en estimación de movimiento con el fin de introducir los principios fundamentales tratados en los siguientes capítulos. Comenzamos con una revisión del modelo de cámara y de la estimación de movimiento en 2D y 3D. Luego, los trabajos relacionados y salientes del estado del arte que se mencionan más adelante en los capítulos siguientes, son discutidos y revisados.

- El Capítulo 3 aborda el estudio de las texturas dinámicas para la estimación de movimiento y su segmentación con énfasis en aplicaciones y su rendimiento computacional. Además, una extensión novedosa a 3D usando imágenes RGB-D se propone.

- Las técnicas de estimación de movimiento que captura movimientos de cuerpo rígido y no rígidos, tales como el flujo óptico y el flujo de la escena, su contraparte en 3D, se desarrollan en el Capítulo 4. Ambos métodos son desarrollados en un marco de optimización con *graph cuts* con el fin de obtener un campo de movimiento preciso e imponer suavidad global a través de funciones que preservan discontinuidades con el fin de evitar sobresuavizado en los límites de los objeto.

- Por último, en el Capítulo 5 se resumen nuestras principales conclusiones y el trabajo futuro.

Chapter 2

Background on motion estimation

In this chapter we establish the basic notions and assumptions that we use throughout this work. Also, we review the related work on motion estimation following our lines of study.

First, we make a quick review of the rigid body motion model that we will use as reference later in this thesis. This review is based on Yi Ma et al. [MSKS12].

Second, we analyse the past and ongoing work on dynamic texture. And then, we focus on optical flow and scene flow literature.

2.1 Rigid-body motion

In order to describe the motion of an object in the world, which is captured by a camera, one should specify the trajectory of all of its points as a function of time. However, assuming the object has a rigid composition, only a single point on its surface is needed, because the distances between any two given points on it, remain constant during time [MSKS12]. This is know as *rigid body motion* model (see Figure 2.1).

The mapping function $g : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is defined as a rotation followed by a translation, for a given point $\mathbf{X}$ at time t. It can be shown that g, the rigid body transformation, preserves the norm, angle, inner and cross products of vectors in the space (and also volumes). Thus, g can describe the complete trajectory of an object during time providing a coordinate frame attached to

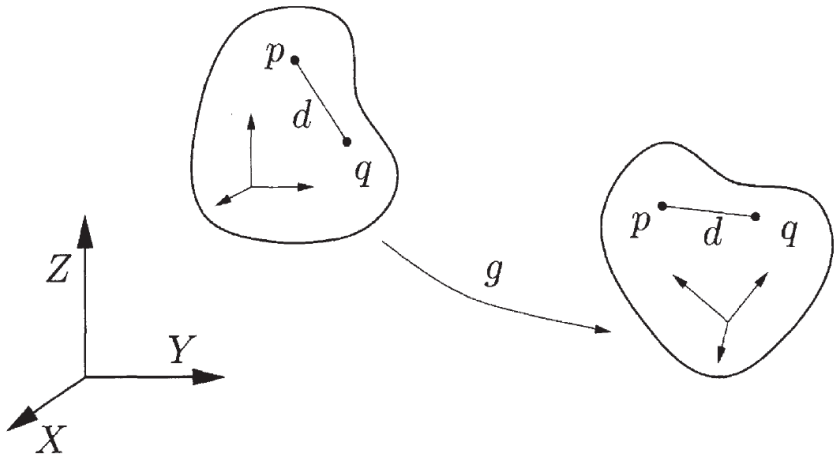


Figure 2.1: In the rigid body motion model, any distance d between a pair of given points p and q on the object's surface, remains constant. The mapping function g describes the motion.

it and with respect to a world reference frame.

Non-rigid motions. Real world objects present deformations and very complex movements that are not well represented with the rigid body motion model. In order to represent a non-rigid motion with deformable characteristics of an object, it is required a complete representation of it. In general, this representation consist of a 3D model (previously known), which is not a common case in real-world applications.

Our goal is to model the entire flow of a scene and therefore non-rigid motions model will not be studied in this thesis.

However, non-rigid motions of a scene can be described as a set of motions that are locally rigid to the surface of a deformable rigid object. This fact, is generally addressed using a dense estimation for the motion of a scene, i.e. each point captured by a sensor, has its own rigid motion. Therefore, a set of motions corresponding to a set of pixels of an object can represent deformations.

2.2 Camera model

The pinhole camera model is the most common and most used in the computer vision community. This model mimics the way a human eye sees the world. In this model, the optical center **O** is the center of projection where the camera frame is placed, Z points in the direction of sight and X and Y axes are defined by the right handed rule (see Figure 2.2).

A point in the world $\mathbf{X} = (X, Y, Z)^T$ is projected in the image plane as $\mathbf{x}_p = (x_p, y_p)^T$. This projection can be calculated by similarity of triangles as:

$$\begin{pmatrix} x_p \\ y_p \end{pmatrix} = \begin{pmatrix} f_x \frac{X}{Z} \\ f_y \frac{Y}{Z} \end{pmatrix} \tag{2.1}$$

where f_x and f_y are the camera focal lengths and give the relation between pixels an metric unit (usually meters).

The intersection of the Z axis and the image plane defines the *principal point* that is represented in image coordinates as c_x and c_y. Since the image reference frame is located in the upper left of the image, the actual image coordinates for **x** are:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{pmatrix} \tag{2.2}$$

Therefore, f_x, f_y, c_x and c_y are known as the intrinsics parameters of the camera. This model can be modified to handle non-square pixels adding a skew factor but we assume it equal to zero for our practical applications. Generally, the camera parameters are obtained performing camera calibration [BP98, Zha00] which also estimate the radial distortion produced by the camera lenses.

In this thesis, we assume that camera parameters are given and images were previously undistorted.

From 2D to 3D. In general, we have an image pixel **x** and we want to obtain the corresponding point **X** in the real world. Without any other information than a single image, the world point corresponding to that image pixel can not be retrieved, since a pixel is a 2D projection and thus infinite real world 3D points exist, corresponding to that pixel. However, given

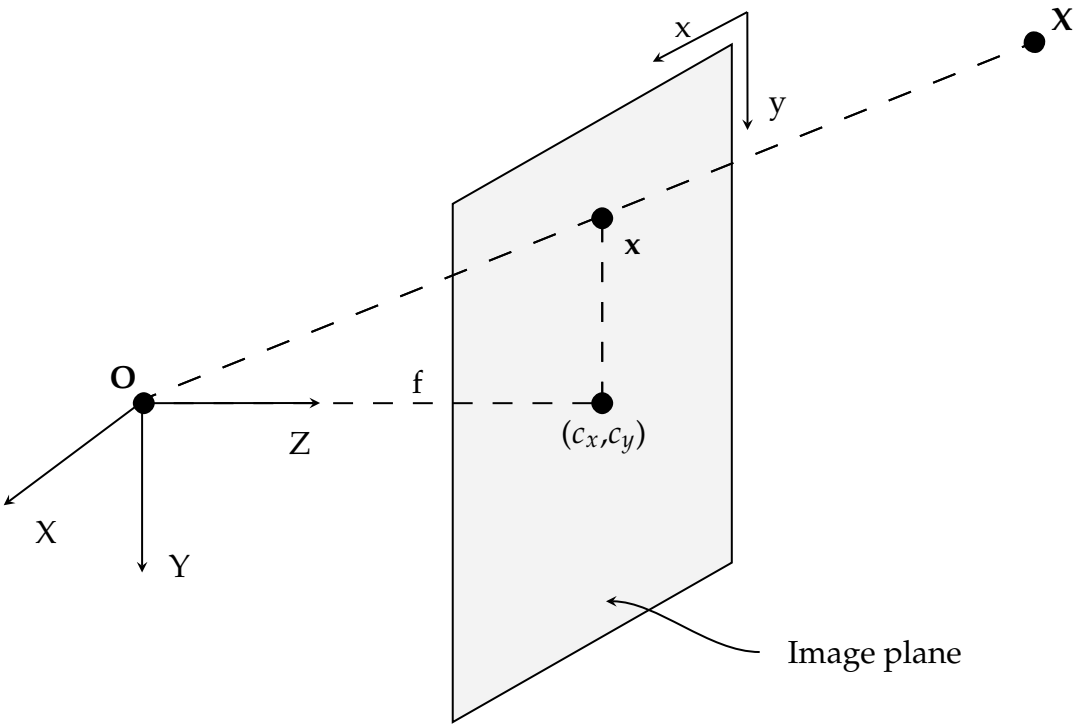


Figure 2.2: The pinhole camera model. We assume $f_x = f_y = f$ for simplicity.

a depth measurement d for $\mathbf{x} = (x, y)^T$ captured by a RGB-D sensor (see Section 1.2) or computed via stereo triangulation from another view of the scene, the real world point $\mathbf{X} = (X, Y, Z)^T$ can be easily obtained as:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} d\,\frac{x-c_x}{f_x} \\ d\,\frac{y-c_y}{f_y} \\ d \end{pmatrix} \tag{2.3}$$

where f_x, f_y, c_x and c_y are the intrinsics parameters of the camera.

Above calculation are best express using matrix formulation. Let $\mathbf{M}$ the matrix of the intrinsics parameters of the camera, defined as:

$$\mathbf{M} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.4}$$

where f_x, f_y and c_x, c_y correspond to the focal length and center of the camera. Let $\mathbf{x} = (x, y)$ the projection on the image plane of $\mathbf{X}$, then $\mathbf{x} = \hat{\mathbf{M}}(\mathbf{X})$ where $\hat{\mathbf{M}} : \mathbb{R}^3 \to \mathbb{R}^2$ is the projective function derived from $\mathbf{M}$ such that $x = \frac{f_x X}{Z} + c_x$ and $y = \frac{f_y Y}{Z} + c_y$, equivalently $\mathbf{x}' = (x', y') = \hat{\mathbf{M}}(\mathbf{X})$.

Note that bold variables represent vectors.

Analogously, we define the inverse of the projective function $\hat{\mathbf{M}}^{-1} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$ which takes the coordinates of a pixel $\mathbf{x} = (x, y)$ and its corresponding depth d and computes the 3D point $\mathbf{X}$ as in Equation 2.3.

Static or moving camera. There is no trivial choice for the coordinate frame of reference for the world / scene. What matters to us is the relative motion between the scene and the object. The most common choice is to define the world coordinate frame as the camera coordinate frame. As a drawback, camera motions infer motions over the scene; this is not a serious problem, since we want to estimate the flow of the whole scene, the camera motion added a global rigid motion to the whole scene. Thus, with a dense motion estimate one could simply compute a global transformation of the scene (that will correspond to the camera motion) consistent with the motion field.

Throughout this thesis, we will assume that we have a fixed camera observing a scene with multiple motions.

2.3 Related work on dynamic textures

Dynamic textures model textures that vary in time and are also known as *visual processes*. They can be viewed as 2D time-varying stochastic processes. Understanding and analyzing motion in videos has been a challenging task in computer vision for decades [NP92, SDW01, MCLC13].

Dynamic Textures provide a different motion estimation approach than the optical flow problem (see Section 2.4 and 4.3). This approach postulates a statistical model of the motion contained in a sequence of images and can characterize complex non-rigid motions such as water, fire, smoke and crowds, among others. There are works that combine both approaches using optical flow as features for DT recognition in videos [FC07, CZS+13].

Dynamic textures are characterized by their statistical parameters for these models. These parameters can be learned form video samples, and their estimation can be used to perform video segmentation [DCWS03, Dor05, CZS+13]. However, when multiple dynamic textures (possibly superimposed) occur in a same scene, this model is not capable of discriminating them well. To face this problem, a *mixture of dynamic textures* [CV08] model has been proposed, to handle this issue as a constituent part of the model. Also, having the statistical parameters that represent the visual procesess allow us to synthetize new videos [YWLS04, DS03, CSS06] with practicable applications in movie editing.

The main drawback using mixture of dynamic textures [CV08] for video segmentation procedure is that the visual process is reduced, as a result of the patch-based partition, similar to the aperture problem in optical flow [BWS05]. To address this problem a global generative model called *layered dynamic texture* [CV09a, CV09b] has been introduced, which introduces co-occurring dynamic textures as part of the model.

Generative models for the visually dynamical processes can be used to carry out recognition and video classification [MCLC13, RCV13] that can be used in tasks such as video surveillance and environmental monitoring [Dor05].

2.4 Related work on optical flow

The optical flow estimation has been studied for decades. Up today, new articles have been published, presenting new approaches and methodologies for the optical flow suggesting that is still an open problem.

Besides of the excellent work of Beauchemin and Barron [BB95] there exists several surveys and reviews about optical flow [FBK15,CGN14] analysing different aspects of the optical flow problem.

Practically at the same time, two fundamental works appear using these constraints, on one side Lucas and Kanade [LK81] using a *local* approach, and Horn and Schunk [HS81] on the other, with a *global* solution. Lucas-Kanade [LK81] assume that the motion field is locally constant and estimate it at each point by minimizing a local mean optical flow equation. Horn and Schunk [HS81] enforces global regularization of the motion field, with the dual aim of relaxing the restriction of optical flow fields and extract more complex motions. Global approaches can compute a dense motion field for the optical flow while local methods can be applied sparsely or densely at each pixel, depending on the application.

Later on, variants of these model were combined [BWS05] with the aim to obtain the advantages of both methods.

Coarse-to-fine or multi-scale strategy have been proved to be very helpful to accelerate convergence and dealing with large displacement of optical flow algorithms [BBM09, LCKW94, WM95, XJM12]. In particular, for global methods based on variational optimization, real-time implementations have been proposed [BWF+05, BWKS05].

Occlusion handling is another matter of study in optical flow methods. Occlusions occur very often in motions captured by a sensor and they can be due to: an object deformation, when an object disappears from the camera field of view, or when another object is interposed. There are works that explicit deal with occlusions [IK08, ZB12, SFVG04, SBM+11]. However, it is very common to ignore them because, thanks to the smoothness constraint, an occluded point in the image will have a motion vector inferred from its neighbors.

Brox et al. [BBPW04] proposed an optical flow estimation based on an innovative variational framework that avoids the usual linerization of the data term in order to obtain an more accurate solution. This work was later accelerated using GPU (graphics processing unit) [SBK10] and extended to handle large displacements [BM11]. Many works were built upon Brox's method and is often referred as the one of the top performer state of the art algorithm in optical flow. It has inspired as well several methods on 3D motion estimation [WRV+08, RMWF10, BMK13, HD07].

The work of Xu et al. [XJM12] is another salient optical flow method, which combines several approaches to deal with most common failures in the optical flow estimation. They employ sparse descriptor matching using SIFT [Low04] to produce an initialization of the flow field at each level of the scale space pyramid and solving at the same time for small-scale structures. A selective combination of color and gradient constraint is also used which allows to retrieve fine detail motions in the flow field.

Also, there are methods based on discrete optimization, such as graph cuts [BVZ01b] and belief propagation [FH06], to compute the optical flow. Shekhovtsov et al. [SKH08] proposed a method based on dual-layer belief propagation to match images across non-rigid deformations and estimate the optical flow field as a consequence. The, SIFT Flow [LYT11] is proposed an extension of the previous method aimed to perform dense matching between images using SIFT descriptors sampled at each image pixel. They employ a dual-layer loopy belief propagation to minimize an energy function that preserves spatial discontinuities and also impose smoothness of the vector field.

As a drawback of discrete optimization methods for optical flow estimation is that the computed motion vectors are discretized. However, achieving subpixel resolution with discrete optimization is possible. Cooke [Coo08] presented a method that estimate a quantized vector field, up to a quarter pixel accuracy. They employ graph cuts optimization over a label space of quantized motion vectors and a 2D smoothing of the motion field to refine the flow estimation.

Tepper et al. [TS12] presented a method to estimate large displacements

in the optical flow using patch match [BSFG09]. This method is bases on SIFT Flow [LYT11] and also compute a discrete motion field, but runs much faster. Later, Hornáček et al. [HBK+14] proposed to the estimate the optical flow using patch match and belief propagation imposing constraints over 9 DoF (degrees of freedom) to improve accuracy.

2.5 Related work on scene flow

In this section, we review the most salient works on scene flow estimation, putting our special attention in those that influenced our work.

The scene flow can be defined as the three dimensional (3D) motion of points in the world. The name scene flow is due to the work of Vedula et al. [VBR+99] where they pose the problem as an optical flow extension to 3D. Since then, the scene flow has been formulated in a variety of ways but always referring to the problem of estimating the motion in a real world scene.

There are many models that can describe the 3D motion, depending on the assumptions we take form the scene and the data we have [BAHH92].

The most common approach is to assume a rigid body motion model where each point in the scene can perform a rotation followed by a translation (see Section 2.1). If the objects in the scene move slowly with respect to the analysed time interval (e.g. between consecutive frames in a ≈30 fps video sequence) and assuming enough image resolution, we can omit the rotational part of the rigid motion and focus only on translation, which is the case of most scene flow estimation methods.

There are multiple approaches to scene flow estimation. The most basic case is to compute the observed 2D motion (optical flow) and then use structural information from the scene (depth or 3D reconstruction) and infer the 3D motion (scene flow).

If we have information from stereo or multiple cameras, an option is to estimate the scene flow using multiple optical flow estimations [VBR+99].

Also, methods built on stereo o multi-view geometry [HD07, BMK10, WRV+08] can retrieve a dense 3D flow field estimating at the same time the

structure (depth or disparity) of the scene, enforcing brightness and depth consistency across different views. In order to avoid ambiguities in the flow due to occlusion, noise and depth discontinuities, a smoothness constraint is imposed. Genarally these methods are based on the variational framework proposed by Brox et al. [BBPW04] for 2D optical flow, which also includes a coarse to fine computation.

Huguet et al. [HD07] propose a method to estimate a dense scene flow field from a sequence of stereo images. They *couple* optical flow and disparity (stereo) estimation problems in a variational framework to jointly estimate structure and 2D motion of the scene between two time steps t and $t+1$. In this way, they can derive the scene flow from the estimated data. They handle occlusions by cross-checking pixels using disparity maps at and between times t and $t+1$. In spite of having a coarse-to-fine implementation, their algorithm requires a good initialization of disparity and optical flow in order to get an accurate scene flow estimation.

The works of Wedel et al. [WRV+08] for one side, and Gong et al. [GY06] for the other, decouple the estimation of structure and velocity. Wedel et al. [WRV+08] for performance issues and Gong et al. [GY06] estimate the motion in the disparity space also known as disparity flow.

A related problem to scene flow is *structure from motion* [MSKS12] where the main objective is to retrieve the 3D structure of a scene using partial views of the same camera or multiple cameras. The work of Basha et al. [BMK10, BMK13] propose to estimate the structure and the motion of the scene in a multi-view geometry approach [HZ03]. Instead of using a rectified stereo camera par, they constraint the scene flow problem by means of data acquired from multiple cameras. Using the images of N calibrated and synchronized cameras they impose a multi-view geometry consistency in the 3D space. Through 3D triangulation, at time t, they obtain the position in the real world of a point projected in the N cameras, and at the next time step $t+1$, the displacement in 3D. This allows them to obtain jointly the structure and flow of the 3D scene. Figure 2.3 shows a diagram of the method for two cameras, similar to the configuration of [HD07] Occlusions are handled using a modification of the Z-buffering algorithm over dispar-

ity maps. As in Huguet et al. [HD07] this method is affected by the cameras' baseline since the method performs well only over points that are visible in all the cameras. Also, 3D estimations tend to fail at depth discontinuities or not well textured image regions.

As many variational methods they impose regularization in order to obtain a motion estimation for regions with occlusions or discontinuities in the flow. This causes over-smoothing around object boundaries and low textured regions.

Another source of oversmoothing is the coarse to fine approach present in the variational framework. They try to avoid it using an initialization of the disparity map. So, the authors propose to set an initial depth estimation using a standard stereo algorithm [FH06], like in Huguet et al. [HD07], at a particular level of the pyramid (not the coarsest) and then continue propagating the results to the finer levels.

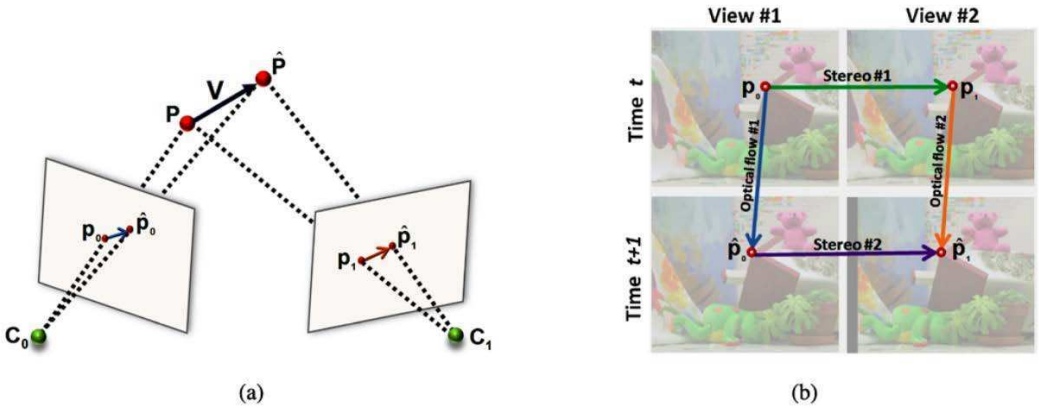


Figure 2.3: Basic formulation of the work form Basha et al. [BMK13] for two cameras. (a) A 3D point $\mathbf{P}$ has a motion $\mathbf{V}$ such that $\hat{\mathbf{P}} = \mathbf{P} + \mathbf{V}$. The points $\mathbf{P}$ and $\hat{\mathbf{P}}$ and the motion $\mathbf{V}$ can be recovered by means of 3D triangulation of their projections in both cameras $\mathbf{p_0}, \mathbf{p_1}, \hat{\mathbf{p}}_\mathbf{0}$ and $\hat{\mathbf{p}_1}$. (b) Relation between stereo disparity (depth) and 2D motion (optical flow) as in Huguet et al. [HD07]. Original Figure from [BMK13].

With the incursion of RGB-D sensors (see Section 1.2) we can avoid computing the structure of the scene and using the depth data with the corresponding appearance (RGB color or intensity). Spies et al. [SJB00] was one

of the first authors in using depth data combined with intensity data, but they treated both separately.

The work of Hadfield et al. [HB11, HB14] presents the scene flow estimation as a set of moving 3D points (point clouds) where each point moves independently. They model the 3D motion of each point in a probabilistic framework given a set of observations (depth and brightness data from RGB-D sensors). Most common assumptions are taken such as brightness constancy and translational 3D motion. However, they do not employ regularization to resolve ambiguities at object borders or occluded points since the accumulation of observations can produce a motion estimate in such cases. It is inherently sparse, because of the discrete and finite number of particles modelled per scene and the fact that probabilities can not be estimated well if there are not enough observations. However, it can be carried to dense using a ray-resampling post processing.

Herbst et al. [HRF13] present a variational solution based on Brox et al [BBPW04] in order to detect small and large motions but regularization is done in 2D and consequently incurs in the same problems as optical flow.

The work of Quiroga et al. [QDC13, QBDC14] combine a local scene flow estimation (in a Lucas-Kanade framewok [LK81, BM04]) with a global regularization using an adaptive total variational optimization. They use RGB-D sequences instead of stereo images and pose the scene flow problem over the image plane, using brightness and depth velocity constraints. The method proceeds alternating between a minimization for each pixel, using a locally rigid motion model, and a global optimization for all the image domain. This work was latter extended to handle camera motion when estimating the scene flow [QBDC14].

Another work that estimate the scene flow using RGB-D sequences is Hornáček et al. [HFR14]. They employ a patch match [BSFG09] based method over 3D point clouds (obtained from RGB-D images), in order to estimate a dense set of correspondences pairs which are then used to infer the scene flow. All point comparisons are performed using a 3D sphere support with radius adapted to the depth and with a match cost which weights gradient and depth similarities. They handle occlusion doing a consistency check

based on forward an backward flows in 2D and 3D. Additionally, they impose regularization through the minimization [BVZ01b] of a global energy which assumes locally rigid motions in the flow field.

A note on time performance The majority of the works reviewed in this section, do not perform complexity analysis of their algorithms. Some of them report execution times and the hardware utilized. Others do the same, and perform comparisons between those times. Without a sound analysis of time and complexity of the algorithms such comparison are not so theoretically valid and are rough comparisons that serve only as a mere information.

2.6 Resumen

En esta sección presentamos un resumen del Capítulo 2 donde se establecen las nociones y supuestos básicos que utilizamos a lo largo de este trabajo. Además, se revisa el trabajo relacionado de estimación de movimiento siguiendo las líneas de estudio de esta tesis.

En primer lugar, se hace una revisión rápida del modelo de movimiento de cuerpo rígido que vamos a utilizar de referencia más adelante en esta tesis. Esta revisión se basa en el trabajo de Yi Ma et al. [MSKS12]

En segundo lugar, se analiza el trabajo previo y en curso sobre el estado del arte de texturas dinámicas. Y después, nos centramos en la literatura de *optical flow* y *scene flow*.

Movimiento de cuerpo rígido

Con el fin de describir el movimiento de un objeto en el mundo, que es captado por una cámara, se debe especificar la trayectoria de todos sus puntos como una función del tiempo. Sin embargo, suponiendo que el objeto tiene una composición rígida, sólo se necesita un punto de su superficie, dado que las distancias entre cualquier par de puntos en el objeto, se mantienen constantes a lo largo del tiempo. Esto se conoce el como modelo de movimiento de cuerpo rígido (véase la Figura 2.4).

Modelo de cámara

El modelo de cámara estenopeica es el más común y el más utilizado en la comunidad de visión por computadora. Este modelo imita la forma en que un ojo humano ve el mundo. En este modelo, el centro óptico **O** es el centro de proyección donde se coloca el marco de referencia de la cámara, el eje Z apunta en dirección a la vista y los ejes X e Y son definidos por la regla mano derecha (ver Figura 2.5).

Un punto en el mundo $\mathbf{X} = (X, Y, Z)$ se proyecta en el plano de la imagen como $\mathbf{x}_p = (x_p, y_p)^T$. Esta proyección se puede calcular la similitud de

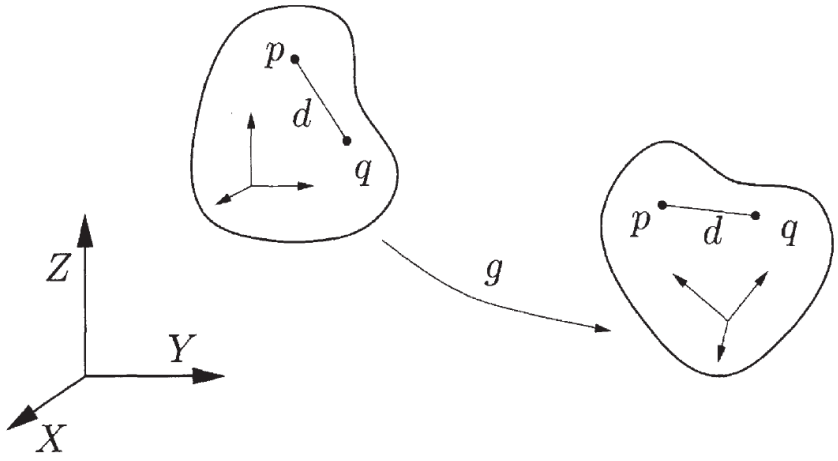


Figura 2.4: En el modelo de movimiento del cuerpo rígido, cualquier distancia d entre un par de puntos dados p y q en la superficie del objeto, permanece constante. La función de mapeo g describe el movimiento.

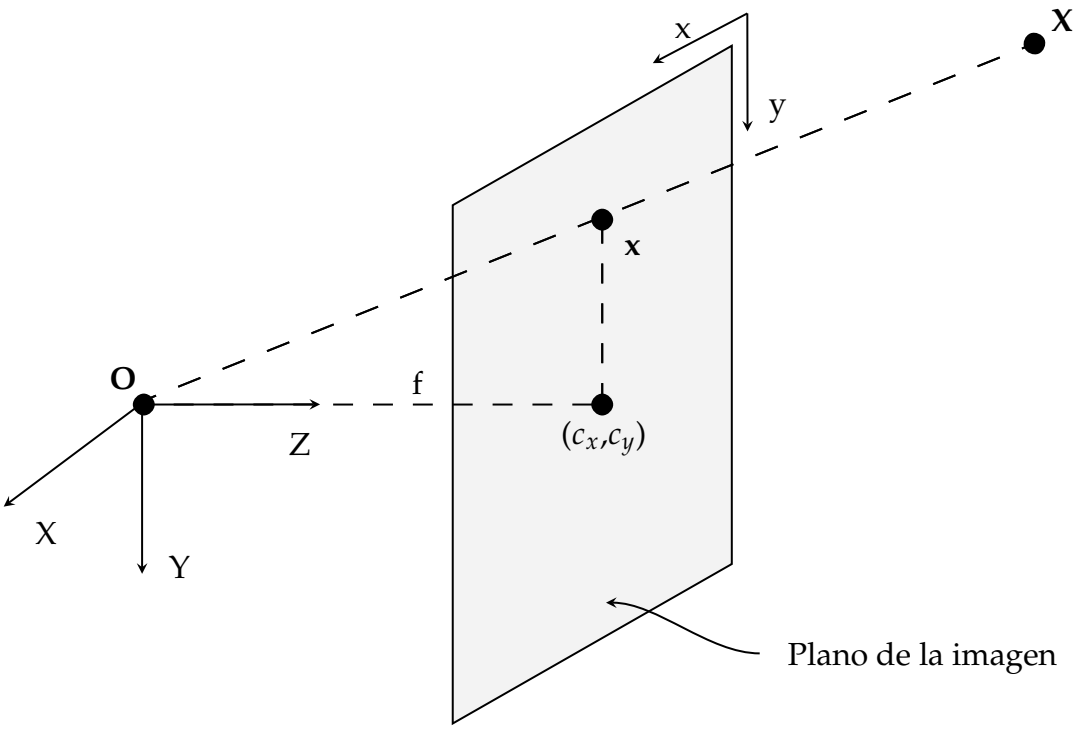


Figura 2.5: El modelo de cámara estenopeica. Se asume $f_x = f_y = f$ por simplicidad.

triángulos como:

$$\begin{pmatrix} x_p \\ y_p \end{pmatrix} = \begin{pmatrix} f_x \frac{X}{Z} \\ f_y \frac{Y}{Z} \end{pmatrix}$$

dónde f_x y f_y son las longitudes focales de la cámara y dan la relación entre píxeles y una unidad métrica (generalmente metros).

La intersección del eje Z y el plano de la imagen define el punto principal que se representa en coordenadas de la imagen como c_x y c_y. Dado que el marco de referencia de la imagen se encuentra en la parte superior izquierda, de hecho, las coordenadas de **x** en la imagen son:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{pmatrix}$$

Por lo tanto f_x, f_y, c_x y c_y son conocidos como los parámetros intrínsecos de la cámara.

De esta forma, dados los parámetros de la cámara podemos conocer las coordenadas en la imagen de un punto del mundo real. Además, de la forma inversa, es posible conocer las coordenadas en el mundo del mundo real para un píxel en la imagen, si con tamos con los parámetros intrínsecos de la cámara y una medición de profundidad para ese píxel.

Nótese que las variables en negrita representan vectores.

En esta tesis, se supone que los parámetros de la cámara son conocidos y además que las imágenes fueron previamente des-distorsionadas. También, vamos a suponer que tenemos una cámara fija que está observando una escena con varios movimientos.

Trabajo relacionado

En este capítulo también se presenta el trabajo previo y relacionado de los principales temas tratados luego en los Capítulos 3 y 4.

En la Sección 2.3 se presenta el trabajo relacionado a texturas dinámicas, remarcando lo principales trabajos que dieron origen al tema y los más novedosos del estado del arte. También, en esta sección se ven los antecedentes en mixturas de texturas dinámica (MDT por sus siglas en inglés), tema que se desarrolla en el Capítulo 3.

Finalmente, en las Secciones 2.4 y 2.5 se analizan los trabajos previos y relacionados sobre *optical flow* y *scene flow*. Dada la extensa literatura en ambos temas, se seleccionan los más importantes y relacionados con el trabajo de esta tesis, desarrollados en el Capítulo 4.

Chapter 3

Dynamic Textures

In video motion analysis, we can characterize the different types of visual motion as follows [PN97]:

- Activities: motions that are repetitive and localized in space. Examples of activities are jogging, walking, boxing, etc.
- Motion events: not repetitive or periodic nor localized; they occur "eventually" such as opening a door.
- Temporal textures: are a special characterization of 2D textures to the time domain. They show certain statistical regularity in space and time, like fire, water or vegetation.

Most of the tasks of video motion analysis can be classified in one of three categories and each of them present very different approaches to solve the main problems regarding motion in image processing and computer vision.

In this chapter we focus our study on the third category: temporal textures, and address the problem of motion estimation and segmentation in image sequences using mixtures of dynamic textures.

3.1 Introduction

Motion can be globally modelled as a statistical visual process known as dynamic texture. This is a generative video model [BL+07], where the ob-

served texture is a time-varying process commanded by a hidden state process.

Dynamic visual processes present in videos are closely related to motion phenomena. Videos containing complex non-rigid motions, such as water, fire, smoke and crowds, among others, are well characterized as visual processes and can be modelled with dynamic textures (DTs).

DTs have been used for segmentation of visual processes in video [DCFS03, CZS+13]. Nevertheless, dynamic textures do not perform very well when multiple dynamic textures occur in the same scene. The use of mixtures of dynamic textures (MDT) allows simultaneously handling different, possibly superimposed, visual processes [CV08].

For a review of these method and a deep related work the reader is referred to Section 2.3.

In the MDT model, the possibility of a video sequence to belong to one of several DTs is explicitly modeled. This allows us to use an expectation maximization algorithm [DLR77] to classify a given set of video sequences into K classes and learn their respective parameters simultaneously. These parameters are subsequently used to classify each pixel using maximum likelihood.

Implementing these statistical models is computationally demanding even for the case of MDT. This is why we study the impact of GPU computing on MDT-based video segmentation. Thus, taking advantage of cutting-edge technology is a necessity. Nowadays, the use of Graphics Processing Units (GPU) in computer vision applications is becoming increasingly popular because of their cost-benefit ratio and their suitability to general purpose computing.

We made two implementations, one in CPU and the other in GPU, of a known segmentation algorithm based on MDT. In the MDT algorithm, there is a matrix inversion process that is highly demanding in terms of computing power. We make a comparison between the gain in performance obtained by porting to GPU this matrix inversion process and the gain obtained by porting to GPU the whole MDT segmentation process. We also study motion segmentation performance by separating the learning part of

the algorithm from the segmentation part, leaving the learning stage as an off-line process and keeping the segmentation as an online process.

When for motion analysis a RGB-D sensor is available, new interesting challenges arise. Ti this end, we extend the MDT algorithm for motion segmentation to take advantage of depth information. We employ an approach that treats motion estimation independently for RGB and depth, and then combine their estimations to provide a segmentation that can discriminate visual processes that vary in texture and depth during time.

3.2 Dynamic texture model

A dynamic texture is a generative model for both the appearance and the dynamics of a video sequence [Dor05]. Dynamic textures can be modelled by means of a linear dynamic system that consists of two processes: an observable video process y in which the video frames are y_t, with t the time variable, and a hidden state process x, which possesses its own dynamics, and drives the evolution of the observable process y. The equations that define the system are:

$$\begin{cases} x_{t+1} = Ax_t + v_t \\ y_t = Cx_t + w_t \end{cases} \tag{3.1}$$

with $x_t \in \mathbb{R}^n$ and $y_t \in \mathbb{R}^m$.

Here, the next value of the state variable x_{t+1} depends on the present value x_t and the present value of the observable process y depends also on x_t. In both equations, there is a noise term, v_t and w_t respectively.

The system is defined by the dimensions of the state process n and the observable process m (usually $n \ll m$), the state transition matrix $A \in \mathbb{R}^{n \times n}$ and the observation matrix $C \in \mathbb{R}^{m \times n}$, and the distributions of the respective noise processes $v_t \sim \mathcal{N}(0, Q)$ and $w_t \sim \mathcal{N}(0, R)$, where Q and R are covariance matrices.

It is also necessary to define the initial conditions of the system in a random manner, which consist of the initial state x_1, considering it to have a normal distribution of mean μ and covariance S, so the probability $p(x_1) =$

$G(x_1, \mu, S)$, with G being the multidimensional Gaussian density. This allows learning a dynamic texture from multiple video samples with different initial frames.

Therefore, a dynamic texture is defined by the parameter vector $\Theta = \{A, Q, C, R, \mu, S\}$. It can be shown [RG99] that the conditional distributions for the state process $p(x_t|x_{t-1})$ and for the observation process $p(y_t|x_t)$ are also normal, so

$$p(x_t|x_{t-1}) = G(x_t, Ax_{t-1}, Q), \tag{3.2}$$

$$p(y_t|x_t) = G(y_t, Cx_t, R), \tag{3.3}$$

where Ax_{t-1} and Cx_t are the respective mean values, and Q and R are the respective covariance matrices, of the multidimensional Gaussian density G. To make explicit the number of the initial and final frames, we will denote $x_1^\tau = (x_1, \cdots, x_\tau)$ and $y_1^\tau = (y_1, \cdots, y_\tau)$ the sequences of states and observations with initial frame 1 and final frame τ.

The joint distribution of these sequences is

$$p(x_1^\tau, y_1^\tau) = p(x_1) \prod_{t=2}^{\tau} p(x_t|x_{t-1}) \prod_{t=1}^{\tau} p(y_t|x_t) \tag{3.4}$$

3.2.1 Mixtures of dynamic textures

Inhomogeneous videos are composed of several dynamic textures. In this work we consider the case in which each dynamic texture corresponds to a static (but unknown) region of the image frame. For this problem, a mixture of dynamic textures (MDT) is an appropriate model.

Under the MDT model, an observed video sequence is sampled from one of K dynamic textures, each having some non-zero probability of occurrence. This is a useful extension for two classes of applications. The first class involves a video that is homogeneous and the second class involves an inhomogeneous video, that is, a video composed of multiple processes that can be individually modelled as a dynamic texture with different parameters.

In the MDT model, we have a set of *a priori* K probabilities $\alpha_1, ..., \alpha_K$ for K dynamic textures, so $\sum_{j=1}^{K} \alpha_j = 1$.

In this model, the random variable $z \sim multinomial(\alpha_1, ..., \alpha_K)$ indicates which of the K dynamic textures is used to represent the observed sequence.

The model parameters are given by $\{\Theta_1, \ldots \Theta_i, \ldots \Theta_K\}$, where $\Theta_i = \{A_i, Q_i, C_i, R_i, \mu_i, S_i\}$ are the parameters for each of the K textures. The probability of y_1^τ is given by:

$$p(y_1^\tau) = \sum_{j=1}^{K} \alpha_j p(y_1^\tau | z = j), \tag{3.5}$$

where $p(y_1^\tau | z = j)$ is the conditional probability given that z is the j-th dynamic texture, i.e., the texture component parametrized by $\Theta_j = \{A_j, Q_j, C_j, R_j, \mu_j, S_j\}$. The equations that define each dynamic texture are:

$$\begin{cases} x_{t+1} = A_z x_t + v_t \\ y_t = C_z x_t + w_t \end{cases} \tag{3.6}$$

where, given z, $v_t \sim \mathcal{N}(0, Q_z)$ and $w_t \sim \mathcal{N}(0, R_z)$ are the gaussian noise terms, Q_z and R_z their covariance matrices, and A_z and C_z, the state transition matrix and the observation matrix.

Figure 3.1 shows the MDT as a graphical model [Jor98]. In this graphical model, an incoming arrow shows dependency, e.g., the observable variables y_t in the probabilistic model (Equations 3.5 and 3.6), depend on the hidden variables z and x_t.

3.2.2 Parameter estimation

To estimate the parameters $\Theta_i = \{A_i, Q_i, C_i, R_i, \mu_i, S_i\}$ with, $i = 1, \ldots, K$ from a set $\{y^{(i)}\}_{i=1}^N$ of N video sequences (the observed data), we have to deal with the fact that there is some missing information: (1) the assignments $z^{(i)}$ of each sequence to a mixture component, (2) the hidden state sequence $x^{(i)}$ that produces each $y^{(i)}$.

This is solved using the expectation maximization (EM) algorithm [DLR77], an iterative procedure for estimating the parameters of a probability distribution, given that the distribution depends on hidden variables.

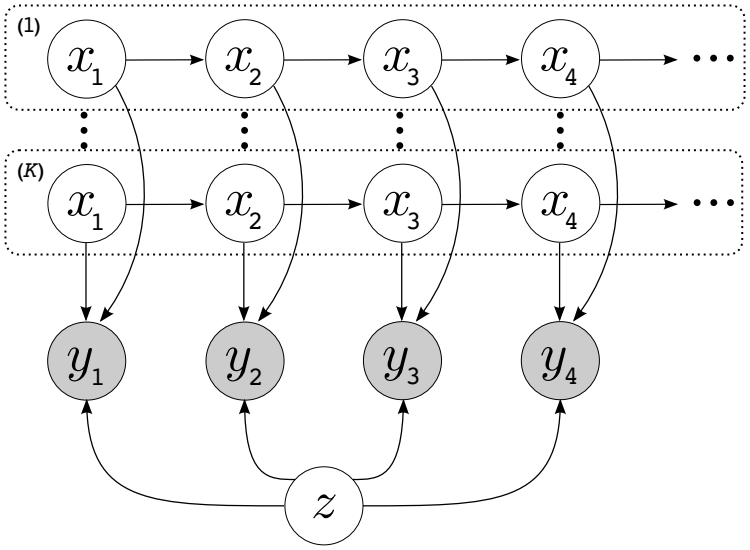


Figure 3.1: Graphical model for mixtures of dynamic textures.

The EM algorithm consists of iteratively performing two steps: the *estimation* step, in which the log likelihood of the complete data problem $p(X, Y, Z; \Theta)$, where X and Z are the set of hidden variables, Y the observable dataset and Θ the parameter set, is replaced by its conditional expectation given the observed data, and the *maximization* step, in which the function obtained at the estimation step is used to find the value of Θ that maximizes it (see [Bis06] as a reference).

- **E-step**: $\mathcal{Q}(\Theta; \hat{\Theta}) = \mathbb{E}_{X,Z|Y;\hat{\Theta}}(\log p(X, Y, Z; \Theta))$
- **M-step**: $\hat{\Theta}^* = \arg\max_{\Theta} \mathcal{Q}(\Theta; \hat{\Theta})$

The E-step estimates hidden states, and hidden assignment variables with the current parameters, and the M-step computes new parameters given the previous estimates.

In [CV08] the authors present EM for the MDT algorithm. In this method, the E-step relies on the Kalman smoothing filter to compute: (1) the expectations of the hidden state variables x_t, given the observed sequence $y^{(i)}$ and the component assignment $z^{(i)}$, and (2) the likelihood of the observation $y^{(i)}$ given the assignment $z^{(i)}$. Then, the M-step computes the maximum-likelihood parameter values for each dynamic texture component j by aver-

aging over all sequences $\{y^{(i)}\}_{i=1}^{N}$, weighted by the posterior probability of assigning $z^{(i)} = j$.

The initialization of the EM algorithm is done by setting each Θ_j using the method in [Dor05] on a random video sequence from the training set.

3.3 Video classification and motion segmentation

Mixtures of dynamic textures (MDT) are well suited to motion segmentation where a moving object or a group of them can be characterized by a dynamic texture. If the model of a dynamic texture is known, then one can estimate the probability of an observed sequence y generated by the model.

Classification In the context of MDT, given a set of video sequences $\{y^{(i)}\}_{i=1}^{N}$, once the MDT model is learned for each $\{y^{(i)}\}$, i.e., all parameters Θ of K dynamic textures are estimated, each sequence $y^{(i)}$ can be classified as the j-th mixture component with the largest posterior probability of being generated by j-th (see Equation 3.7).

$$\ell_i = \arg\max_j \left(\log p(y^{(i)}; \Theta_j) + \log \alpha_j \right) \tag{3.7}$$

This procedure automatically performs a classification of a video dataset into K categories, useful for video retrieval or video semantics.

Segmentation The aim of motion segmentation is to create a static image describing the regions from a video with homogeneous appearance and motion, i.e., annotate each video location with the number of components to which it belongs, such as fire, water, crowd, traffic jam, etc. This can be achieved generating a set of spatio-temporal patches from a single video and then classifying them using Equation 3.7 as mentioned before.

Algorithm 1 summarizes the motion segmentation process. Since the algorithm takes as input a single video y, the first step is to generate a set of videos from it. This can be achieved generating a set of sub-videos or spatio-temporal patches $\{y^{(i)}\}_{i=1}^{N}$ sampled from y in a non-overlapping manner

and with the same number of frames. Next, we can classify them using MDT as before. Finally, at each pixel location we extract a spatio-temporal patch and assign it to a mixture component, according to Equation (3.7). The output segmentation matrix M will have the same size as y and in each cell will have the number of mixture component that was assigned to the corresponding patch.

Algorithm 1 Motion segmentation with MDT

Input: video y, number of components K

Output: segmentation image M

1. Extract N non-overlapping spatio-temporal patches $\{y^{(i)}\}_{i=1}^{N}$ from input video y
2. Call EM algorithm with $\{y^{(i)}\}_{i=1}^{N}$ and K
3. For each p pixel location in y
   3.1. Let $y^{(i)}$ a spatio-temporal patch centered at p
   3.2. Set $\ell_i = \arg\max_j \left(\log p(y^{(i)}; \Theta_j) + \log \alpha_j \right)$
   3.3. Set $M_p \leftarrow \ell_i$
4. Return matrix M segmented into K components

3.3.1 Decoupling the learning step

In the motion segmentation process described in the previous section, there are two main stages: (1) a learning part, where the parameters Θ are extracted using the EM algorithm; (2) a classification part, using the maximum a posteriori probability ℓ. Therefore, we can apply (1) to a certain video, store parameters Θ and then use (2) to classify another video.

Decoupling these two stages has many advantages in applications where video capture has almost no variation and the visual process has a similar nature. This is the case in prediction/diagnostic of traffic conditions (see Section 3.4.3), crowd detection, etc, where we can train our system off-line

by learning the required parameters and then perform an online classification.

Also, decoupling the learning step has a clear reduction in computational load, because we can perform motion segmentation in videos skipping heavy numeric calculations of the EM algorithm, gaining in this way a meaningful improvement in computing time performance.

3.3.2 Extension to RGB-D sequences

We propose a straightforward extension for RGB-D images, assuming independence of appearance (intensity) and depth:

$$p(y;\Theta) = p(a;\Theta^a)\ p(d;\Theta^d) \tag{3.8}$$

where $p(a;\Theta^a)$ and $p(d;\Theta^d)$ are probabilities for the observed appearance a and depth d data, respectively, given the unknowns parameters Θ^a and Θ^d.

In this context, we model texture for one side and structure for the other, allowing us to compute the joint probability as the sum of the RGB and depth log likelihoods.

Thus, with a slightly modification of algorithm 1 we can estimate this joint probability. The main idea is to skip the classification part (line 3.2) of Algorithm 1) using it only for the likelihood estimation at each pixel. Obtaining in this way, the RGB and Depth likelihood independently. Finally, an RGB-D likelihood is estimated as the sum of both likelihoods. With this RGB-D likelihood, we can classify each pixel as belonging to one of the K component.

3.4 Experimental results

This section presents computer time performance results of the MDT algorithm presented in the previous section. Both CPU and GPU implementations were developed in order to compare each other. Also, an optimized version in GPU of covariance matrix inversion computation was done in order to assess its impact in time performance. Finally, an implementation

of the decoupling of the learning step of the MDT algorithm presented in Section 3.3 was evaluated.

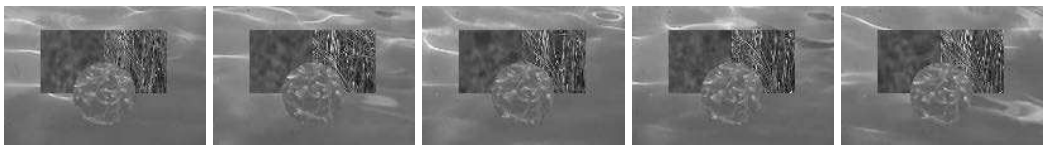
3.4.1 Evaluation of GPU impact on MDT algorithm performance

The most important parameters and video frame size were tried to test their impact on the execution time.

To carry out time performance tests, the synthetic dynamic texture segmentation database (SynthDB) [Cha09] was used. This database is composed of 299 synthetic videos especially generated to assess video segmentation of visual processes. Each video is composed of K components ($K \in \{2,3,4\}$), of different temporal textures such as water, fire, moving plants, etc (see Figure 3.2). Also, they contain a ground-truth template and an initial segmentation contour. The video dimensions are 160×110 with 60 frames at 30 fps with 60 frames total.

(a) K = 2

(b) K = 3



(c) K = 4

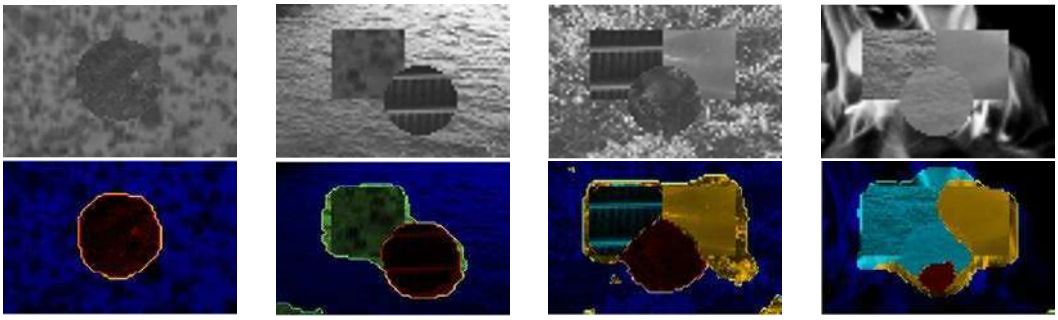
Figure 3.2: Example image sequences from videos of the SynthDB database with K different visual components

Algorithm implementations were tested on a CPU computer with an AMD Phenom processor and an NVIDIA Tesla C2070 1.15 GHz GPU. The

NVIDIA Tesla C2070 has 448 streaming processor cores and a total amount of global memory of 5376 MBytes. The CPU implementation was developed using Matlab and translated into GPU using functions built in the Parallel Computing Toolbox.

Figure 3.3 shows segmentation examples over the SynthDB. The segmentation quality is computed with the Rand index metric [HA85].

The Rand index is a measure of similarity between two data clusters. Rand index values range from 0 to 1, where 0 indicates complete disagreement and 1 indicates that the groups of clusters are the same.



(a) r=0.99 (b) r= 0.90 (c) r= 0.85 (d) r=0.82

Figure 3.3: Video frame examples from SynthDB with $K \in \{2, 3, 4, 4\}$, respectively (in rows) and their respective segmentations (best viewed in color). Rand indexes r are below each of them.

3.4.2 Optimizing the inverse of the covariance matrix

Following [HWHC11], in order to avoid the bottleneck of computing the inverse of a covariance matrix in the Kalman Filter at the expectation step, a C++ implementation [Bou12] of the Cholesky factorization using GPU [LTN+11] and CUBLAS [NVI12] was used. This implementation is based on the fact that, since covariance matrices are positive-definite, it is possible to perform the Cholesky factorization and then solve the (upper and lower) triangular systems using TRSM (Triangle Solve Multiple) function of CUBLAS [NVI12].

To perform the tests, we vary $K \in \{2, 3, 4\}$, the number of mixture components, and take four different video sizes with a scaling factor $SF \in$

$\{0.5, 1, 1.5, 2\}$ (grouped by segmentation component K in Figures 3.4 and 3.5). Obtaining in this way, for each K, four videos of 80×55, 160×110, 240×165, and 320×220, respectively, all with 60 frames.

We implemented three versions of the motion segmentation algorithm with MDT:

(1) Whole MDT algorithm on CPU

(2) Whole MDT algorithm on CPU and only the inverse on GPU

(3) Whole MDT algorithm on GPU

Then, two performance comparisons were carried out:

- (1) versus (2) shown in Figure 3.4
- (1) versus (3) shown in Figure 3.5

Figure 3.4 shows computing times obtained with and without the GPU's inverse implementation, while the rest of the computation was performed by the CPU. Darker bars show GPU times.

A reduction in computing times for large videos can be observed. This is because the ratio of data transfer time (to and from the GPU) with respect to GPU processing time is lower for larger videos. For smaller videos, computing times are better for CPU processing than for GPU processing.

Figure 3.5 shows the GPU performance against the CPU performance for an experience in which the comparison was made between performing the whole computation process in the GPU versus the same in the CPU. Only $SF = 2$ was considered for $K = 2$ due to GPU memory limitations.

Using the same argument as before, the overhead in time of switching data back and forth to GPU has to be lower than the processing time. Therefore, as expected, the GPU implementation outperforms the CPU implementation only when the video is bigger than the original size (1.5 and 2 times bigger).

Finally, the most important parameters and video frame sizes were tried in order to assess their impact in the execution time. As expected, frame size, sub-video size m and number of components to find K affect directly

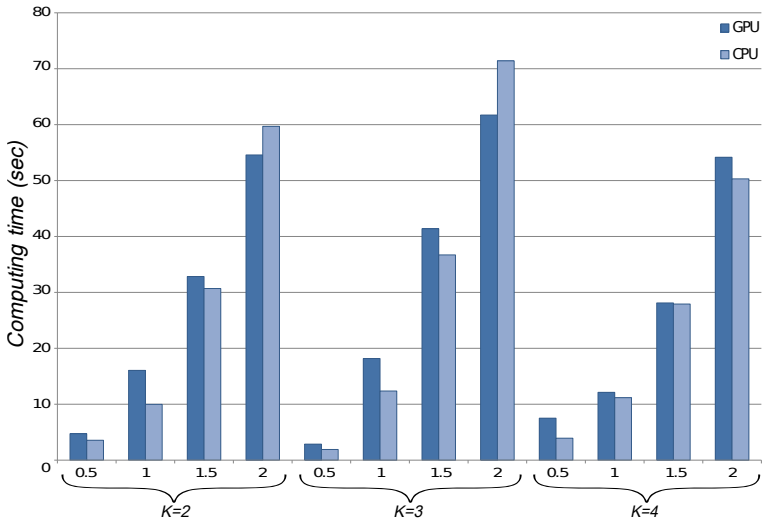


Figure 3.4: Time comparisons of two implementations of matrix inversion using CPU and GPU. The horizontal axis shows different scaling factors grouped by segmentation component K.

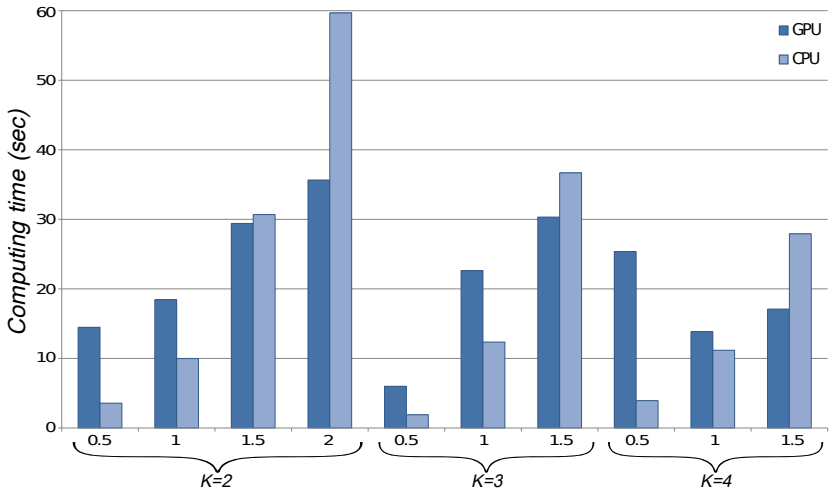


Figure 3.5: Computing time performances of the segmentation process on GPU and CPU. The horizontal axis shows different scaling factors grouped by the number of segmentation components K.

the computer execution time. However, by varying the size n of the hidden state x and fixing the remaining parameters, computer times remain approximately constant. Also, it was found that the best parameters were $m = 100$, $n = 10$ with respect to Rand index and execution time.

Figure 3.6: Example frames from the traffic database. Light, medium and heavy traffic conditions in the first, second and third rows, respectively.

3.4.3 Evaluation of performance gain obtained by decoupling the learning step

In this study we used the database introduced in [CV05b, CV05a] which consists of 254 videos of highway traffic in Seattle collected from a single stationary traffic camera over 2 days. This database includes various traffic and weather conditions. Traffic conditions were tagged as light, median and heavy by a human. The video frame size was 320×240 and videos were captured at 10 fps resulting in 42–52 frames per video. Figure 3.6 shows example frames taken from the traffic database. The parameters used for this test were set to $m = 100,\ n = 10$ as in the previous section.

Experimenting on this traffic database is important because of its possible real-world application of motion segmentation with MDT. Thus, we can assess the performance of our algorithm in a more realistic way than using only a synthetic database. Sadly, we lack ground-truth information of different visual components on this database; hence we are limited to present segmentation results qualitatively. Although the main focus of our work is in performance, motion segmentation results presented in this section seem to be promising to continue with a large variety of applications such as automatic traffic prediction and diagnosing.

In order to assess the benefit of decoupling the learning step in the segmentation algorithm we compute speedups (the ratio of improvement in

time performance).

To measure speedups, we took 50 videos from the traffic database (with different traffic conditions). From those 50 videos, we selected three different videos to be used in the learning step, corresponding to traffic conditions: light, medium and heavy (named V_L, V_M and V_H, respectively).

The speedups for each $K \in \{2, 3, 4\}$ in Table 3.1 were obtained as follows:

First, we ran the whole motion segmentation algorithm for all the 50 videos and then averaged all the resulting running times; let us call these average times T^K.

Second, for each video of V_L, V_M and V_H, we learned the parameters of the model and then classified all the 50 videos with these learned parameters; let us call T_L^K, T_M^K and T_H^K the averaged running times of classification (third column of Table 3.1).

Finally, the speedups for each K were computed as the ratio: $\frac{T^K}{1/3(T_L^K+T_M^K+T_H^K)}$ (fourth column of Table 3.1).

The best accelerations were obtained for $K = 2$ and $K = 3$. As mentioned before, the parameter K impacts directly on time performance and, in particular, on classification time. The speedup for $K = 4$ is low and is a consequence of higher classification times.

It was also observed from Table 3.1 that for each $K \in \{2, 3, 4\}$, the selected video used for learning does not impact on the classification time.

Figures 3.7 and 3.8 show examples of the motion segmentation algorithm applied to the traffic database, decoupling the learning step. These examples are presented as a quality assessment for motion segmentation in traffic. The relevant results were obtained for $K = 2$ and $K = 3$; $K = 4$ does not produce a good quality segmentation and was used only to test performance. The impossibility for the algorithm to find four different visual components is not a major drawback. In fact, segmenting the videos in four visual components is also a hard task for a human carefully watching the screen.

The best segmentations were obtained when the traffic condition was *medium* (cars flow and movement in the highway are nearly constant). This occurs because this is an ideal situation to model the scene with mixtures

K	Video	Avg. classif. time	Speedup
2	V_L	8.09s	1.50x
	V_M	8.76s	
	V_H	8.72s	
3	V_L	10.71s	1.44x
	V_M	11.16s	
	V_H	10.80s	
4	V_L	13.31s	1.15x
	V_M	14.97s	
	V_H	17.84s	

Table 3.1: Averaged classification times (in seconds) and speedups (in times) obtained decoupling the learning step.

of dynamic textures, where there are well-defined regions in the video that can be associated with the visual processes.

3.4.4 Evaluation on RGB-D sequences

In this section, we show the applicability of MDT for motion segmentation on RGB-D sequences. For the best of our knowledge there is no prior work using RGB-D or depth information when modelling dynamic textures.

SKIG dataset

To evaluate our segmentation methods in RGB-D motion sequences, we use the Sheffield KInect Gesture (SKIG) Dataset [LS13].

This is a database designed to make gesture recognition with Kinect, however, we select sequences that will allow us to segment the motion of a hand. It should be noted that this approach can also be used for video recognition and classification and therefore our method, could be employed to recognize gestures.

The SKIG dataset comprises over than 1000 hand gesture sequences acquired by a Kinect RGB-D sensor. It was recorded from 6 subjects perform-

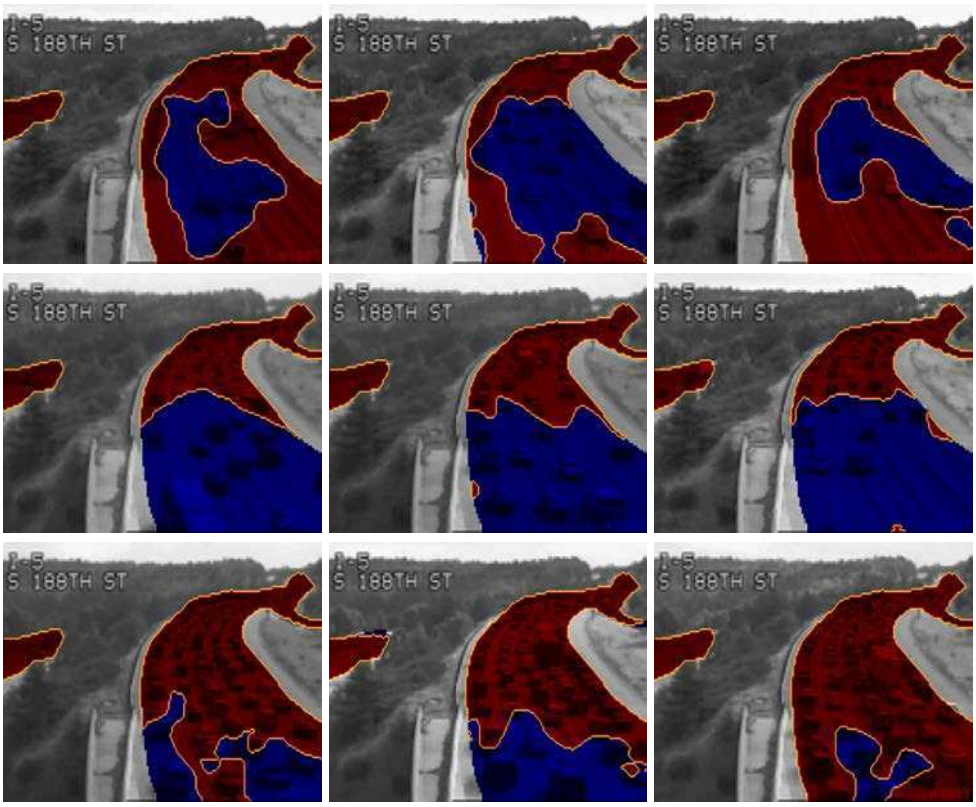


Figure 3.7: Examples of traffic segmentation using $K = 2$ visual components. Light, medium and heavy traffic conditions in the first, second and third rows, respectively.

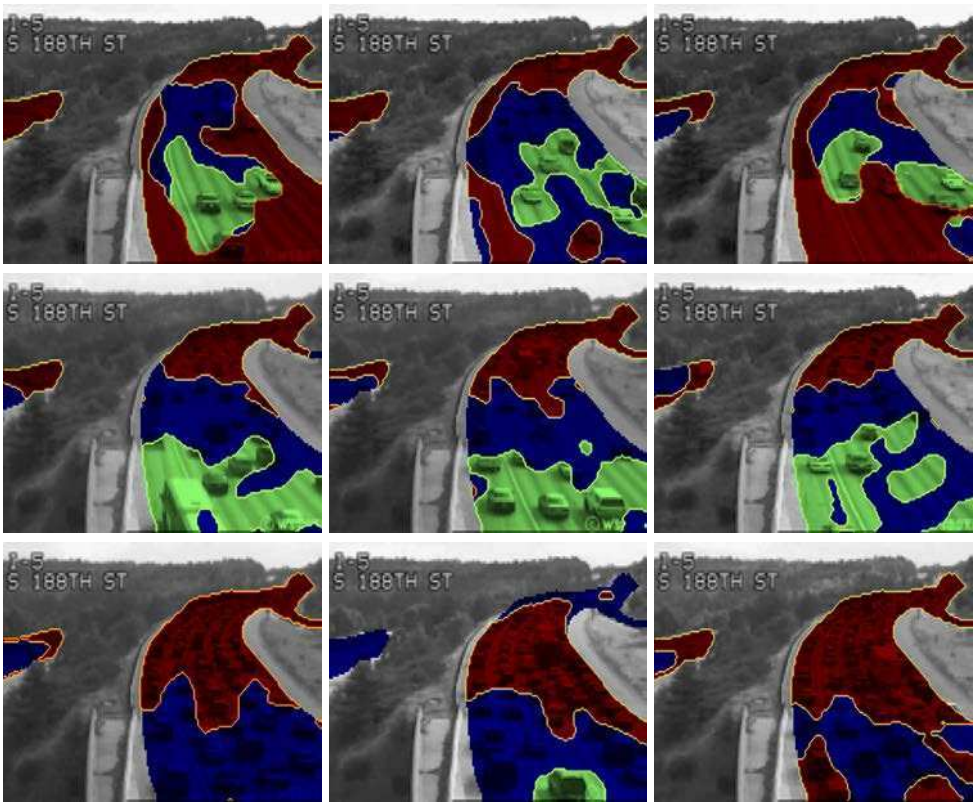


Figure 3.8: Examples of traffic segmentation using $K = 3$ visual components. Light, medium and heavy traffic conditions in first, second and third row, respectively.

ing hand gestures among 10 selected categories with three different hand postures and also under different background and illumination conditions (see Figure 3.9). The frame size of the sequences is 320×240 and have between 63 to 248 frames in total.

For the experiments in this section, we use *circle* and *wave* sequences for the first subject in the SKIG dataset. Figures 3.10 and 3.11 show example frames for these sequences.

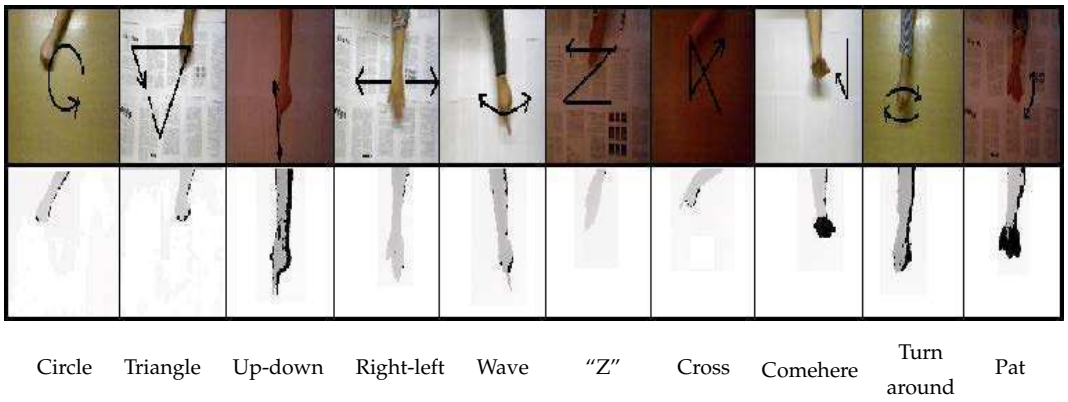


Figure 3.9: Example RGB (top) and depth (bottom) images from the SKIG dataset. Left to right: the 10 different categories of hand gestures recorded with different backgrounds. Original Figure from [LS13].

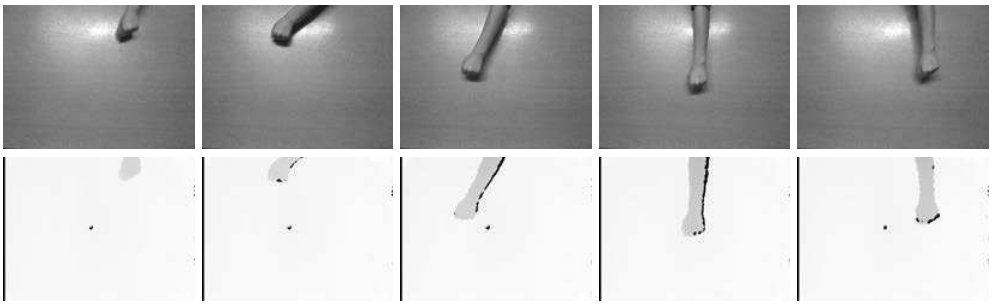


Figure 3.10: Example RGB-D images from *circle* sequence. The top row shows RGB images and corresponding depth images at the bottom row. Darker pixels mean closer distances to the sensor.

Discussion

For the experiments in this section, we estimate only two different motions, i.e. $K = 2$, due to the low complexity of motions present in the video se-

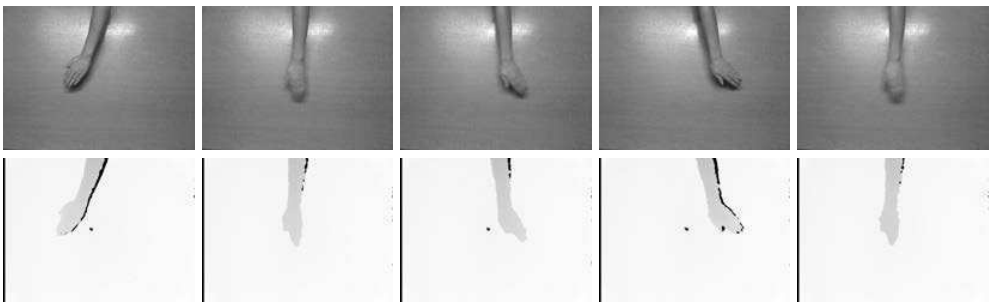


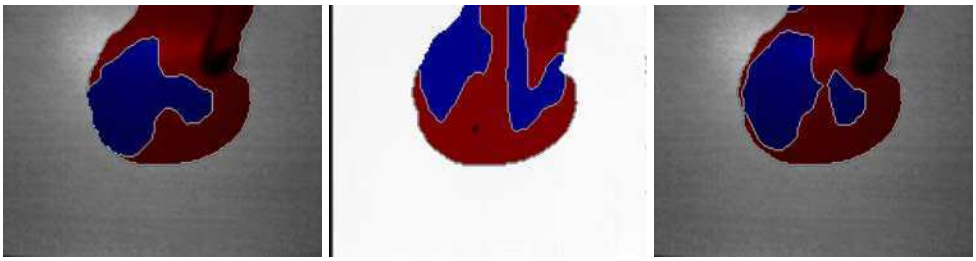
Figure 3.11: Example RGB-D images from *wave* sequence. The top row shows RGB images and corresponding depth images are shown at the bottom row. Darker pixels mean closer distances to the sensor.

quences. Also, we apply a scale of 0.5 to the videos (due to memory issues), resulting in a size of 160×120. The rest of the parameters were the same as in the Section 3.4.3. Non-colored regions are assumed with zero motion and they are estimated as pixel with low intensity variance along frames.

Figure 3.12 shows segmentation results for the *circle* sequence (see Figure 3.10) using for segmentation only the RGB or depth images (Figures 3.12a and 3.12b) and also using the combined approach on RGB-D (Figure 3.12c). As we can see, the RGB component of the video captures the circular movement of the hand better, corresponding to a slow motion in the red regions and a faster one in blue regions. In this case, the depth does not capture very well the circular motion but is able to find the areas where the hand is steady (blue regions). It should be noted that, in this sequence, the hand is always at the same distance from the sensor and then depth does not provide much information for the MDT algorithm. Then, in the RGB-D combined approach, both segmentations are intuitively combined delimiting slightly better (blue) regions where the hand is steady or move slowly.

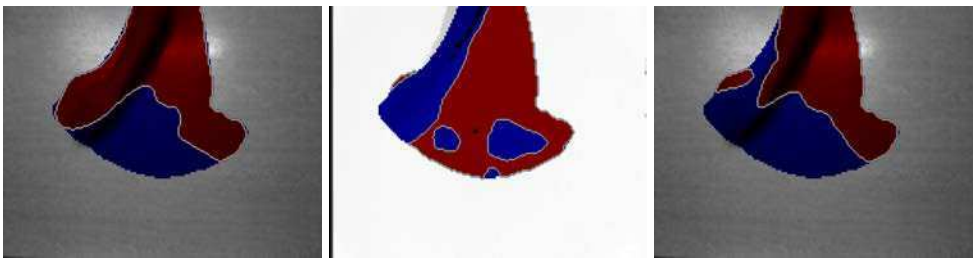
Figure 3.13 shows segmentation results for the *wave* sequence (see Figure 3.11) using for segmentation only the RGB or depth images (Figures 3.13a and 3.13b), and also using the combined approach on RGB-D (Figure 3.13c). In this sequence, in addition to hand waving from left to right, there are changes in the height at the end of the motion trail. As in the previous case, RGB separates well the fastest areas from the slowest. Depth, in this case, helps to discriminate variations in height (distance from the sensor). As a

result of the RGB-D segmentation, both properties are combined, achieving a motion segmentation that prioritizes both changes in texture and depth.



(a) RGB (b) Depth (c) RGB-D

Figure 3.12: Different motion segmentations for the *circle* sequence. (a) RGB only, (b) Depth only and (c) RGB combined with depth.



(a) RGB (b) Depth (c) RGB-D

Figure 3.13: Different motion segmentations for the *wave* sequence. (a) RGB only, (b) Depth only and (c) RGB combined with depth.

3.5 Conclusions

In this chapter we presented CPU and GPU implementations of the motion segmentation algorithm. Our GPU implementation is an adaptation from the CPU implementation using library modules that run directly over a GPU card.

The importance of this work resides in the analysis of the cases where making a full GPU implementation is a worthwhile task. Performance tests were carried out to evaluate benefits and drawbacks of porting algorithms to GPU. Our performance evaluation showed that computing time is reduced significantly with the use of GPU when video size equals or exceeds 320×240, which is the case for video applications of most practical interests. When analyzing the bottleneck of the MDT algorithm, matrix operations and specially matrix inversion are the most time-consuming tasks. Our results showed that a meaningful acceleration can be achieved with the use of a specialized GPU function to compute the inverse.

Decoupling the learning step proved to be advantageous in terms of computing performance, as can be seen in the previous sections. However, this advantage decreases with the number of components K to be classified.

Computing times for our implementation of the motion segmentation algorithm using MDT (with and without the learning step) are too far from reaching real-time performance. Besides this, we found interesting speedups that could further optimize a full GPU implementation of the MDT algorithm.

Additionally, we extended the MDT algorithm using appearance (RGB) and structure (depth) as independent cues for motion estimation in RGB-D sequences. Both estimations are merged in a combined approach that produces a segmentation that prioritizes both changes in texture and depth. This MDT algorithm for RGB-D was analysed over sequences taken from an RGB-D dataset, obtaining as a conclusion that depth information helps to discriminate visual processes.

As future work, we are interested in implementing the segmentation algorithm based on layered dynamic textures [CV09a]. This method shows

better quality results for motion segmentation, but has a more complex model.

3.6 Resumen

En este capítulo se estudiaron los procesos visual dinámico, también conocidos como texturas dinámicas y sus aplicaciones para estimar movimiento y segmentarlo.

El movimiento puede ser modelado a nivel general como un proceso estadístico visual conocido como texturas dinámicas o *dynamic textures* (DT). Este es un modelo de vídeo generativo [BL+07] donde la textura observada es un proceso variable en el tiempo comandado por un proceso de estado oculto.

Los procesos visuales dinámicos que se encuentran en los videos están estrechamente relacionados con los fenómenos de movimiento. Videos que contienen movimientos no rígidos complejos, tales como agua, fuego, humo y multitudes, entre otros, son bien caracterizados como procesos visuales y pueden ser modelados con texturas dinámicas (DTs).

Las DTs se han utilizado para la segmentación de procesos visuales en video [DCFS03, CZS+13], sin embargo, no funcionan muy bien cuando múltiples texturas dinámicas se producen en la misma escena. El uso de mixturas de texturas dinámicas o *mixture of dynamic textures* (MDT) permite un manejo simultáneo de diversos procesos visuales, posiblemente superpuestos [CV08].

En el modelo de MDT, la posibilidad de una secuencia de vídeo de pertenecer a una de varias DT se modela de forma explícita. Esto nos permite utilizar un algoritmo de *expectation-maximization* (EM) [DLR77] para clasificar secuencias de vídeo en clases y aprender sus respectivos parámetros simultáneamente. Estos parámetros se pueden utilizar posteriormente para clasificar cada píxel mediante máxima verosimilitud y así segmentar el movimiento de acuerdo a los procesos visuales presentes en el mismo.

Cuando para el análisis de movimiento un sensor RGB-D está disponible, se plantean nuevos retos interesantes. Para este fin, extendimos el algoritmo de segmentación de movimiento con MDT para aprovechar la información de profundidad. Empleamos un enfoque que trata a la estimación de movimiento independientemente para RGB y profundidad, para luego

combinar sus estimaciones y así proporcionar una segmentación que puede discriminar procesos visuales que varían en textura y profundidad al mismo tiempo.

La implementación de estos modelos estadísticos es computacionalmente demandante, por esta razón, estudiamos el impacto de los algoritmos desarrollados en GPU (*graphic processing units* para la segmentación de vídeo basada en MDT.

Hicimos dos implementaciones, una en la CPU y la otra en GPU, de un algoritmo de segmentación basado en MDT. Nuestra evaluación del rendimiento demostró que el tiempo de cálculo se reduce significativamente con el uso de GPU cuando el tamaño del vídeo es igual o mayor 320×240, que es el caso de la mayoría de aplicaciones de interés práctico.

Al analizar el cuello de botella del algoritmo de MDT, encontramos que las operaciones matriciales y en especial el calculo de la inversa, son las tareas que más tiempo consumen. Nuestros resultados mostraron que una aceleración significativa se puede lograr con el uso de una función especializada en GPU para calcular la inversa de una matriz de covarianza que se presenta en la etapa de *expectation* del algoritmo EM.

Además, en la sección 3.3.1 propusimos una modificación al algoritmo de MDT que permite la disociación de la etapa de aprendizaje. Esto ha sido demostrado ser ventajoso en términos de rendimiento computacional, como se puede ver en la Sección 3.4.3 donde se aplicó para segmentación de movimiento en tráfico de vehículos. Sin embargo, esta ventaja disminuye con el número de los componentes para ser clasificado.

Con respecto a la estimación de movimiento en secuencias de RGB-D, realizamos varias experimentos con una base de datos de reconocimiento de gestos que nos permitió distinguir diferentes zonas de movimientos en los videos de gestos (ver Sección 3.4.4). Al combinar ambas estimaciones, la de RGB por un lado y la de profundidad por el otro, vimos que se obtiene una estimación de procesos visuales que prioriza tanto los cambios en textura como en profundidad.

Chapter 4

Optical Flow and Scene Flow using graph cuts optimization

In this chapter, we present algorithms for 2D and 3D motion estimation over a pair of consecutive images RGB and RGB-D, respectively, that use graph cuts optimization.

4.1 Introduction

The motion of real world objects captured by a camera can be very hard to describe. Real world objects present deformations and very complex movements that are not well represented with a single global motion model. Thus, in order to deal with real world scenes composed of multiple objects and different motions a common solution is to assume a locally rigid motion at each visible point in a scene.

This fact, is generally addressed using dense estimations for the motion of a scene, where each point captured by a sensor has its own motion, but at the same time has to be consistent with its neighbouring points. This makes it possible to deal with missing information and large displacements, problems that often occur in real-world scenarios. A dense motion estimation is a 2D or 3D motion vector field or a flow field of the captured real world scene at each visible point.

The *optical flow* problem deals with the estimation of the motion flow

field projected into the image plane, i.e. a 2D motion field, induced by a 3D motion in the scene. This induced motion on the image plane is known as the apparent motion.

When a depth measurement of the objects in the world is available, given by an RGB-D sensor or a stereo camera (such as the ones provided by RGB-D sensors or stereo cameras), the *scene flow* estimation is a more complete problem, which enables the computation of the observed 3D motion of a real world scene.

Graph cuts is an optimization method used to solve frequent problems in computer vision, such as stereo, image segmentation and denoising with very good results.

We formulate both problems of optical flow and scene flow as a problem of energy minimization and solve them using graph cuts. To this end, one can construct a specialized graph for the energy function such that finding its minimum cut will be equivalent to finding the minimum of the energy.

It has been shown that graph cuts optimization can find a very good local minimum of the energy function for computer vision problems [SZS+08]. However, not many works use graph cuts to minimize the optical flow or scene flow energies and obtain the motion vector field. Exploring the space of all possible motions for each pixel requires increases the computational performance of the method. As a consequence, the design of efficient methods for dense motion calculation with graph cuts becomes a real challenge.

This chapter begins with a review of graph cuts optimization and its application to computer vision. Then, in Sections 4.3 and 4.4 we detail how to use graph cuts to solve optical flow and scene flow energies. Finally, in Section 4.5, we present the conclusions and discussions while proposing future work and direction.

4.2 Graph cuts

Graph cuts is a well known method from combinatorial optimization and is equivalent to the problem of finding the maximum flow or the minimum cut (max-flow/min-cut) in a graph [Cor09].

Informally, the maximum flow can be viewed as the maximum amount of water that "flows" trough the pipes between a source and a sink, where the pipes correspond to edges in a graph with capacities equal to edge weights. A cut in a graph is a partition of its vertices in two disjoint sets. The cost of the cut is the sum of the weights of the edges having one endpoint in each set. Thus the problem of finding a cut in a graph with minimum cost is know as *graph cuts*.

Let a weighted directed graph $G = \langle \mathcal{V}, \mathcal{E} \rangle$ consisting of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$ of ordered pairs (u, v), where u and v are in $\mathcal{V}$, with weight or capacity $c(u, v) \geq 0$. Also, an additional of two vertices s and t, called the *source* and the *sink*, are defined. Vertices s and t are called *terminals* and there are no incoming edges to s nor outgoing edges from t. Graph G is also called a *flow network*.

An *s-t cut* (or simply a cut) is a partition $\{S, T\}$ of the vertex set $\mathcal{V}$ such that $T = \mathcal{V} - S$ and $s \in S$ and $t \in T$. The cost of the cut is defined as:

$$cut(S, T) = \sum_{u \in S, v \in T, (u,v) \in \mathcal{E}} c(u, v) \tag{4.1}$$

Thus, the minimum *s-t cut* is the cut of G with minimum cost among all possible *s-t cuts*.

A $flow$ in G is a function $f : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ that satisfies these two properties:

Capacity constraint: $\forall u, v \in \mathcal{V} \Rightarrow \ 0 \leq f(u, v) \leq c(u, v)$

Flow conservation: $\forall u \in \mathcal{V} - \{s, t\} \Rightarrow \sum_{v \in \mathcal{V}} f(u, v) = \sum_{v \in \mathcal{V}} f(v, u)$

When $(u, v) \notin \mathcal{E}$ the flow function is $f(u, v) = 0$.

Thus, the flow value $|f|$ is defined as:

$$|f| = \sum_{v \in \mathcal{V}} f(s, v) - \sum_{v \in \mathcal{V}} f(v, s) \tag{4.2}$$

that is, the total flow out of the source s minus the flow into the source.

Thus, the *maximum flow* in a flow network G is defined as a flow with maximum value.

The Ford-Fulkerson theorem [FF62] postulates that a minimum cut in G is equivalent to finding its maximum flow in G, which can be computed

very efficiently [FF62, CG97]. In the max-flow algorithm if capacities are integers, termination is guaranteed as long as capacities are integers.

Figure 4.1 shows an example of a minimum cut and a maximum flow on network graph, the value of the maximum flow is 28 and also the cost of the minimum cut.

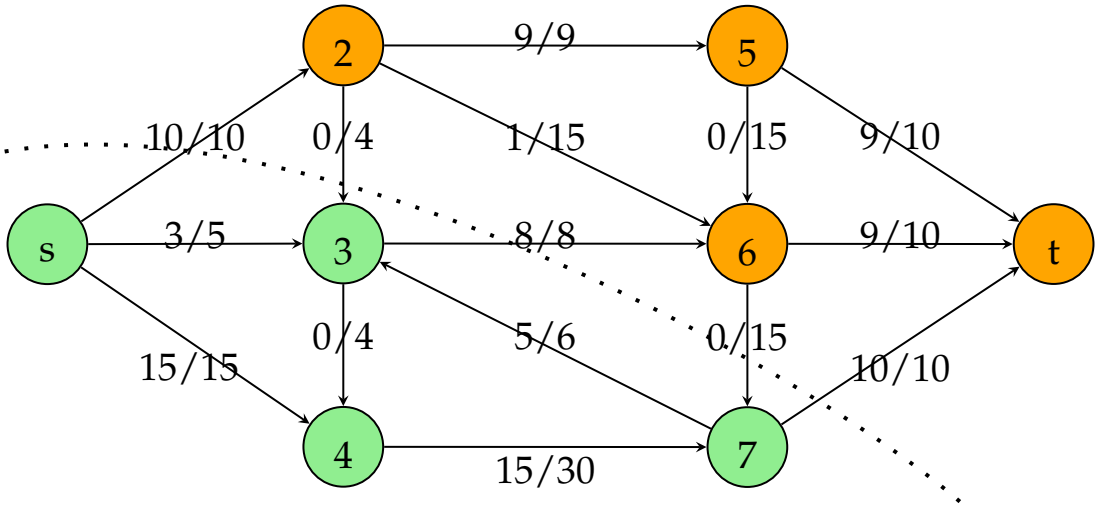


Figure 4.1: An example of a maximum flow/minimum cut on a graph. At each edge (u, v) flows and capacities are displayed of the form $f(u, v)/c(u, v)$. The value of the maximum flow is 28 and also the cost of the minimum cut. The dotted line indicates the cut division in sets S and T, also displayed in different colors.

4.2.1 Computer vision problems as energy minimization

Many computer vision problems can be formulated as a minimization of an energy function composed of two terms:

$$E = E_{data} + E_{smooth} \tag{4.3}$$

The data term E_{data} measures how well the observed data fits our model and the smoothness term E_{smooth} impose a spatial regularization over the image domain.

For instance, in image denoising we are given an image I corrupted by noise in its pixel intensities I_x and in order to obtain the original (unknown)

clean image $\tilde{I}$, we model E by:

$$E = \underbrace{\sum_{\mathbf{x}\in\Omega_I} (\tilde{I}_\mathbf{x} - I_\mathbf{x})^2}_{E_{data}} + \underbrace{\sum_{\mathbf{x},\mathbf{y}\in\mathcal{N}} (I_\mathbf{x} - I_\mathbf{y})^2}_{E_{smooth}} \tag{4.4}$$

where $\mathcal{N}$ is an usual 4-neighbour system and Ω_I the image domain.

This data term penalizes noisy image intensities and the smoothness term penalizes different intensities between neighboring pixels in the recovered image, favouring a smooth restoration.

4.2.2 Representing energies with graphs

Many graph constructions can be found in the literature for minimizing energy functions such as the one described in Equation 4.3 using max-flow algorithms [RC98, JB06, STC09].

In general, vertices correspond to pixels and are connected with edges when they are neighbours in the image or interact in some way in the original problem.

A common procedure is to create a graph G including one vertex v for each pixel in the image, and two additional terminal vertices s and t. When two pixels are adjacent, neighbours or interact with each other in the image, an edge is created (called *n-link*). Also, edges are created between pixels and terminal vertices (called *t-links*).

Edge capacities/cost are defined to represent the energy function to minimize. The cost of *t-links* are derived from the data term and represent the penalty of assigning a vertex to the terminal. Analogously, *n-link* costs are derived from the smoothness term, representing the interaction penalty between pixels regulating the discontinuity of the solution.

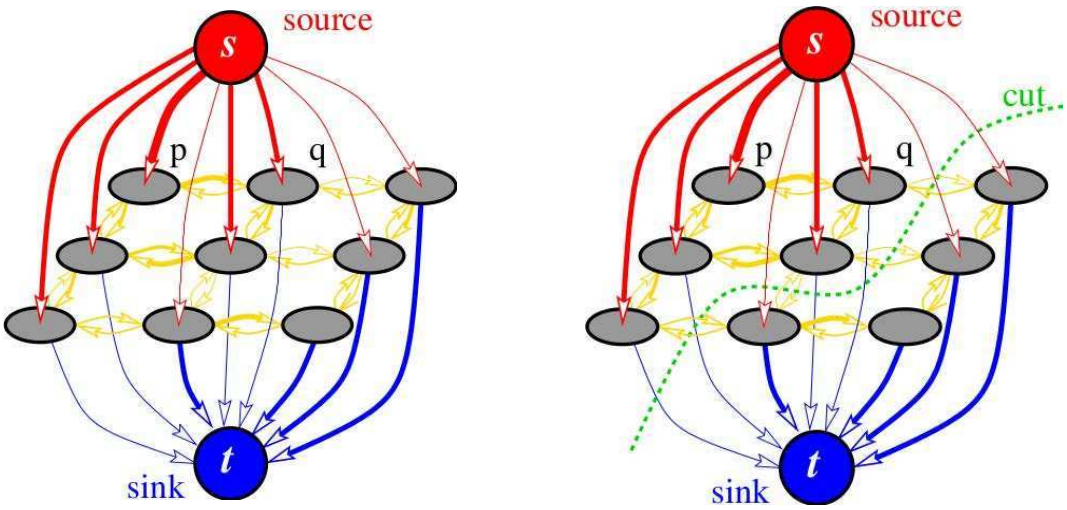
Since a cut $\{S, T\}$ is a disjoint partition of $\mathcal{V}$, it infers a *binary* labelling, i.e, vertices or pixels in partition S or T will be assigned to label s or t, respectively.

Figure 4.2 shows an example of a graph and its minimum cut for a 3x3 image with two labels. This can be an example of image segmentation, where the terminal vertex s represents the label *foreground* and terminal ver-

tex t represents the label *background*, and adjacent edges account for discontinuity in the segmentation. Thus, the cut will "break" the graph in foreground/background assignments to pixels.

The final structure of the graph will depend on the number of pixels, labels and the energy to minimize.

In this work, we use the general-purpose graph construction for minimizing an energy function for the problem of motion estimation in RGB and RGB-D, proposed by Kolmogorov and Zabih [KZ04] which has interesting properties of optimality.



(a) A graph G

(b) A cut on G

Figure 4.2: Example of a directed graph with edges weights represented by line thickness (a) and its minimum cut (b). Figure extracted from [BK04]

Energy minimization with multiple labels and preserving discontinuities

In the previous example, the energy is minimized using only two possible labels. However, many computer vision problems involve multi-label assignments, such as stereo or motion estimation (where a pixel can be assigned to different disparities or velocities), denoising (where the labels are all the possible gray values), etc.

The *multiway cut* [Vaz13] problem is a generalization of the minimum

cut to handle more than two terminal nodes (labels). Unfortunately, this problem is known to be NP-Hard and so, there is no polynomial algorithm to solve it. There are special cases where this problem can be computed efficiently: for planar graphs [STC09] and special graph constructions [XS05], when labels are consecutive integers or with a convex smoothing function [RC98, Ish03].

Having a convex function as regularizer leads to oversmoothed solutions. Most of the works that impose spatial regularization deal with the problem of discontinuities. Then, non-convex functions that preserve discontinuities are generally used. However, minimizing an energy with a discontinuity-preserving function, also, has been proven to be an NP-Hard problem [BVZ01b]. Thus, one can estimate a local minimum of the energy function via an approximation algorithm for this case.

In [BVZ01b] the authors present a very efficient algorithm called α-expansion which computes the local minimum of an energy function with discontinuity-preserving functions. This algorithm works by iteratively finding an α-expansion move for each possible label α which minimizes the energy. An α-expansion move can be computed efficiently using graph cuts with binary valued labels. Also, it has been proven that the resulting minimum of the obtained energy is within a multiplicative factor from the global minimum.

4.3 Optical flow

The optical flow is defined as the observed motion of a real world 3D object projected over the image plane. The induced motion on the image plane is known as the *apparent motion*.

In order to estimate a motion vector over the image plane, two basics assumptions are posed: (i) *brightness constancy* which states that the intensity of a point remains constant across different views; and (ii) *smooth variation* or *motion coherency* which proposes that neighboring points in a scene (or image) move smoothly with respect to each other.

The aforementioned assumptions help to resolve ambiguities related to the apparent motion but they not always hold. For instance, in the presence

of occlussions or sharp motion discontinuities, commonly encountered at object boundaries.

The motion estimation problem can be posed as an energy minimization and, as many others in computer vision, can be viewed as a visual correspondence problem, where wrong pixel correspondences are penalized, i.e. they have a high energy.

First, we must transform the original problem into a label assignment problem.

Given a pair of images I and I' taken at two time steps t and t', a set of pixels Ω_I corresponding to the image domain of I and a finite set of labels $\mathcal{L} \subset \mathbb{R}^2$ representing the space of all possible 2D motion vectors, the goal is to find a labelling function $f : \Omega_I \to \mathcal{L}$ which minimizes the energy:

$$E(f) = \sum_{\mathbf{x} \in \Omega_I} D_{\mathbf{x}}(f_{\mathbf{x}}) + \sum_{\{\mathbf{x},\mathbf{y}\} \in \mathcal{N}} V_{\mathbf{x},\mathbf{y}}(f_{\mathbf{x}}, f_{\mathbf{y}}) \tag{4.5}$$

where $\mathcal{N} \subseteq \Omega_I \times \Omega_I$ is any neighbourhood system represented as a set of pairs of adjacent pixels, and $f_{\mathbf{x}}, f_{\mathbf{y}} \in \mathcal{L}$, $\forall \mathbf{x}, \mathbf{y} \in \Omega_I$. The first term in Equation 4.5 is known as the data term E_{data}, where the function $D_{\mathbf{x}} : \mathcal{L} \to \mathbb{R}$ measures how well the label $f_{\mathbf{x}}$ fits the pixel $\mathbf{x}$ given the observed data, i.e., it measures the cost of assigning the label $f_{\mathbf{x}}$ to pixel $\mathbf{x}$. Then, the second term, also known as the smoothness term E_{smooth}, where the function $V_{\mathbf{x},\mathbf{y}} : \mathcal{L} \times \mathcal{L} \to \mathbb{R}$ must be a discontinuity-preserving function, i.e. a function that accounts for abrupt changes at object boundaries and avoids over-smoothing.

With the energy formulated as in Equation 4.5 we can construct a graph representing the energy $E(f)$ [KZ04] such that the cost of the minimum cut in G is the minimum of $E(f)$, and from which we can deduce the labelling function f.

The data term and the smoothness term are derived from the brightness constancy assumption and the smooth constraints from optical flow, explained in the next sections.

4.3.1 Data term

The *brightness constancy* assumption states that the intensity of a point in the scene remains constant across different views, that is for a pixel $\mathbf{x}$ that suffered a motion $\mathbf{v}$ from time t to t':

$$I_{\mathbf{x}} - I'_{\mathbf{x}+\mathbf{v}} = 0 \tag{4.6}$$

then, the data term $D_{\mathbf{x}}$ for the optical flow energy is defined as:

$$D_{\mathbf{x}}(\mathbf{v}) = ||I_{\mathbf{x}} - I'_{\mathbf{x}+\mathbf{v}}|| \tag{4.7}$$

where $||\cdot||$ is any norm or cost function.

Dense feature descriptors

The data term in Equation 4.7 measures how well the observed intensity is adjusted to the motion field. This constraint can be enforced using feature descriptors such as SIFT [Low99]. Computing descriptors densely at each pixel produces a multidimensional image that can be used in the data term of the energy to minimize. Thus, $I_{\mathbf{x}}$ and $I'_{\mathbf{x}+\mathbf{v}}$ represent feature vectors (with 128 dimensions in the case of SIFT), and they can be compared with an appropriate norm, such as the L_1 norm.

4.3.2 Smoothness term

The smoothness term, or the interaction penalty $V_{\mathbf{x},\mathbf{y}}$ is defined to account for motion coherency (neighbour pixels will have similar labels/motion vectors), so $V_{\mathbf{x},\mathbf{y}}$ is defined as follows:

$$V_{\mathbf{x},\mathbf{y}}(f_{\mathbf{x}}, f_{\mathbf{y}}) = k_1 \min(k_2, C^s(f_{\mathbf{x}}, f_{\mathbf{y}})) \tag{4.8}$$

where $C^s(f_{\mathbf{x}}, f_{\mathbf{y}})$ is the smoothness cost for assigning labels $f_{\mathbf{x}}, f_{\mathbf{y}} \in \mathcal{L}$ to neighbour pixels $\mathbf{x}, \mathbf{y} \in \mathcal{N}$, and $k_1, K_2 \in \mathbb{R}$ are constants that control the regularization of the solution. In particular, C^s can be defined as follows:

$$C^s(\mathbf{u}, \mathbf{v}) = ||\mathbf{u} - \mathbf{v}|| \tag{4.9}$$

where $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$ and $||\cdot||$ is any norm or cost function.

Note that the minimum in Equation 4.8 produces truncated norms that impose discontinuity in the interaction penalty function $V_{\mathbf{x},\mathbf{y}}$.

The amount of smoothing is regulated by the constants k_1 and k_2 which impose a trade off between regularization and fit to the observed data. Choosing an appropriate function $V_{\mathbf{x},\mathbf{y}}$ for the smoothness term is a key problem that affects globally the solution. Also, the resulting solution must be coherent with the human perception because a minimum of a energy function not always agrees with an human sees, unless we can model the human perception as an energy function.

Contextual information

Besides of the motion coherency assumption, there is contextual information that can be used to improve the interaction penalty cost $V_{\mathbf{x},\mathbf{y}}$. Looking only the first image I, it is very likely that neighbouring pixels $\mathbf{x}$ and $\mathbf{y}$ share similar intensity $I_{\mathbf{x}} \approx I_{\mathbf{y}}$ and then probably have a similar label.

Without violating the discontinuity persevering property of $V_{\mathbf{x},\mathbf{y}}$, one can modify the interaction penalty to vary according to pixel intensity. Then, we can weight the smoothness cost C^s by a function u measuring intensity differences between pixels.

$$V_{\mathbf{x},\mathbf{y}}(f_{\mathbf{x}}, f_{\mathbf{y}}) = u_{\mathbf{x},\mathbf{y}}\, C^s(f_{\mathbf{x}}, f_{\mathbf{y}}) \tag{4.10}$$

Then, the values for $u_{\mathbf{x},\mathbf{y}}$ should be small for large intensity differences and large for similar intensities. Also, $u_{\mathbf{x},\mathbf{y}}$ can be set according an edge detector, where $u_{\mathbf{x},\mathbf{y}}$ should be small if and edge was detected between $\mathbf{x}$ and $\mathbf{y}$. Defining, $V_{\mathbf{x},\mathbf{y}}$ in this way it helps in cases when images have low texture.

4.3.3 Optical flow computation

In this section, we describe the computation of optical flow using energy minimization via graph cuts.

Let us recall the energy function for the optical flow detailed in the previous sections.

Given a pair of images I and I' taken at two time steps t and t', a set of pixels Ω_I corresponding to the image domain of I and a finite set of labels $\mathcal{L}$ representing the space of all possible motion vectors, the goal is to find a labelling function $f : \Omega_I \to \mathcal{L}$ which minimizes the energy:

$$E(f) = \sum_{\mathbf{x} \in \Omega_I} ||I_{\mathbf{x}} - I'_{\mathbf{x}+f_{\mathbf{x}}}|| + \sum_{\mathbf{x},\mathbf{y} \in \mathcal{N}} k_1 \min(k_2, C^s(f_{\mathbf{x}}, f_{\mathbf{y}})) \tag{4.11}$$

where $C^s(f_{\mathbf{x}}, f_{\mathbf{y}})$ is the smoothness cost for assigning labels $f_{\mathbf{x}}, f_{\mathbf{y}} \in \mathcal{L}$ to neighbour pixels $\mathbf{x}$ and $\mathbf{y}$, respectively, $\mathcal{N} \subseteq \Omega_I \times \Omega_I$ is a 4-neighbourhood system, and $k_1, K_2 \in \mathbb{R}$. And we define C^s as $C^s(\mathbf{v}, \mathbf{u}) = ||\mathbf{v} - \mathbf{u}||$ where $\mathbf{v}, \mathbf{u} \in \mathbb{R}^2$ and $||\cdot||$ is any norm or cost function.

The optical flow field obtained using graph cuts directly on the energy function from Equation 4.11 is no very satisfactory and suffers from the staircase effect. Due to the discrete nature of graph cuts optimization the resulted flow field lives in a 2D space which is a discretization of $\mathbb{R}^2$. Even so, there are works in the literature that use graph cuts to minimize the optical flow energy and compute accurately motion fields. The method described next is based on similar algorithms that works with discretized flows [Coo08, LYT11, BVZ01b, SAL07].

Initialization. A label space $\mathcal{L}$ with subpixel resolution is chosen to account for possible motion vectors, defining a range value, which from labels are sampled. For example, for a subpixel resolution of 0.5 and a range value of $[-2, 2]$, the label set is $\mathcal{L} = \{-2, -1.5, -1, \ldots 1.5, 2\}$.

Coarse-to-fine. In order to deal with different scales of motions an to maintain a reduced label space, a coarse-to-fine procedure is employed. The main idea is to create sub-sampled versions of both images I and I' and estimate the optical flow at each level of the coarse-to-fine procedure to use it as initialization in the finer levels. Also, an additional median filtering between each level is employed in order to to avoid vector outliers in the flow field and gain consistency.

Decoupled motion estimation. The estimation of the optical flow using graph cuts over the Equation 4.11 is carried out decoupling the horizontal and vertical flows in the minimization. The minimization is applied in two steps using unidimensional labels. First, the minimization is performed using only horizontal motion assuming the vertical motion is zero. Second, the optimization is performed using over the vertical motion labels fixing the horizontal motion.

Flow filtering. Finally, a heat filtering is performed on the resulted flow field in order to stress the sub-pixel solution. This filtering is performed applying a convolution kernel iteratively on the image domain on each direction of the flow independently.

Data term. Note that, in the data term from Equation 4.11 the intensity value for $I'_{\mathbf{x}+f_x}$ is necessary to be obtained by means of interpolation if sub-pixel labels were chosen. Experimental results showed us that the best norm to compare intensity values is the L_1 norm. If images have color information, $I_{\mathbf{x}}$ is an 3-component RGB vector.

Dense SIFT descriptors. Additionally, dense SIFT descriptors can be employed in order to obtain a better fit to the observed data. At each pixel $I_{\mathbf{x}}$ represent a 128-component descriptor vector. SIFT descriptor are compared using the L_1 norm, so the energy function is unmodified.

Our implementation employs the code provided by Olga Veksler and Andrew Delong[1], described in the articles: [BK04, BVZ01a, KZ04].

4.3.4 Experimental results

In order to test the correctness of our method, we start testing the optical flow approach with a dataset with ground truth [BSL+11].

[1] http://vision.csd.uwo.ca/code/

There are two common error metrics to verify how close our estimated motion field is from the ground truth: the *endpoint error* (EE) [ON94] and the *angular error* (AE) [BFB94], defined as following.

Let $(u, v) \in \mathbb{R}^2$ be an estimated motion vector and $(u^{gt}, v^{gt}) \in \mathbb{R}$ the ground truth flow,

$$EE = \sqrt{(u - u^{gt})^2 + (v - v^{gt})^2} \tag{4.12}$$

$$AE = \arccos\left(\frac{u \cdot u_{gt} + v \cdot v_{gt} + 1}{\sqrt{u^2 + v^2 + 1} \cdot \sqrt{u_{gt}^2 + v_{gt}^2 + 1}}\right) \tag{4.13}$$

The angular error (AE) is the angle between two 2D vectors extended to 3D, setting to 1 the third dimension in order to measure errors in the plane and avoid division by zero. The AE is very sensitive to small errors due for small error displacements and orientations. On the contrary the endpoint error (EE) penalizes estimations over large distances. Also, the root-mean-square (RMS) for the entire flow field is commonly used. The RMS is very similar to the EE in nature, but extended to the image domain:

$$RMS = \sqrt{\frac{1}{N} \sum_{\mathbf{x} \in \Omega} (u_{\mathbf{x}} - u_{\mathbf{x}}^{gt})^2 + (v_{\mathbf{x}} - v_{\mathbf{x}}^{gt})^2} \tag{4.14}$$

where Ω is the set of pixels where the flow is defined, $N = |\Omega|$ its cardinality, $(u_{\mathbf{x}}, v_{\mathbf{x}})$ the motion vector estimated at pixel $\mathbf{x}$ and $(u_{\mathbf{x}}^{gt}, v_{\mathbf{x}}^{gt})$ the corresponding ground truth flow at $\mathbf{x}$.

Middlebury dataset

In the work of Baker et al. [BSL+11] the Middlebury benchmark dataset for optical flow is presented. The dataset is composed of eight images with motion flow ground truth that have been used to assess the performance of optical flow algorithms for many years even to the present day. Not of all them have ground truth information so we restrict our attention only to those having it.

The ground truth consist of a dense 2D motion field from a pair of RGB images. Figure 4.3 shows three example sequences from the dataset. The first image of the sequence and the corresponding ground truth flow are

displayed. The motion field follows the color convention showed in Figure 4.4 where saturated colors correspond to large vectors.

(a) Dimetrodon

(b) RubberWhale

(c) Urban3

Figure 4.3: Example sequences from the Middlebury dataset. On the left, the first image of the sequence and the corresponding ground truth flow on the right. The motion field follows the color convention showed in Figure 4.4.

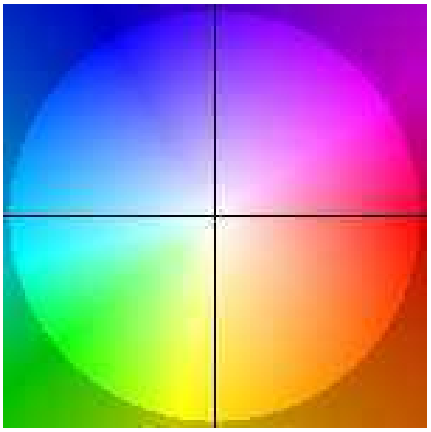


Figure 4.4: Color encoding for vectors in 2D. Saturated colors correspond to large vectors.

Discussion

For each sequence in the dataset, we ran two versions of our algorithm described in Section 4.3. First, the optical flow with graph cuts minimization called OF_{GC}, and second a modified version using dense SIFT descriptors, called OF_{GS}.

Also, we perform comparisons with three other optical flow methods: a variational optical flow algorithm based on Brox et al. [BBPW04] named VOF, SIFT-Flow [LYT11] named SF and Patch-Flow [TS12] named PF. For a review of these method and a deep related work the reader is referred to Section 2.4.

The VOF is an implementation[2] of the Brox et al. work [BBPW04] considered as one of the best performing in the state of the art. SF and PF are very similar to our method except that they do not allow subpixel accuracy.

For our methods OF_{GC} and OF_{GS} we perform 6 levels from the coarse-to-fine procedure and select labels using a vector quantization of quarter of a pixel with a maximum value of 2. For the other methods we use default parameters as defined in corresponding articles.

Tables 4.1 and 4.2 show the average angular error (AAE) and the average endpoint error (Avg. EE), respectively, for each of the method aforementioned. Note that VOF is the only method that employ continuous op-

[2]http://people.csail.mit.edu/celiu/OpticalFlow/

timization.

Results for our methods with and without using descriptors have very similar performance. This is mainly due to the nature of the motion field to be estimated. The pair of images in the dataset have relative small pixel displacements (a maximum flow of 11.65 pixels on average).

The PF method does not perform very well in this dataset because is a method developed to work with large displacements. However, errors in vector field are very similar to sift flow since both methods try to minimize the same energy function but with different approaches.

Our methods OF_{GC} and OF_{GS} performs slightly better than SF. Besides of having a similar formulation, the fact of allowing motion vectors with subpixel precision produces a more accurate vector field.

The VOF method is considered the best performing method having lowest errors in the motion field. This method employs a variational technique to minimize the optical flow energy based on continuous optimization algorithms. Achieving in this way a very precise motion field specially suited for small displacements.

However, for Hydrangea, Urban3 and Venus sequences, our method is not too far the variational method and in particular for Dimetrodon our method has lower errors.

Regarding the computational time all methods take an average of 10 seconds per sequence while our methods take around half an hour. Sadly, this makes impracticable for applications where real-time efficiency is required.

AAE	OF_{GC}	OF_{GS}	SF	PF	VOF
Dimetrodon	3.93	2.53	7.89	8.98	3.09
Grove2	9.16	9.73	10.91	12.94	6.41
Grove3	18.73	18.75	19.43	26.07	15.3
Hydrangea	3.33	3.44	3.92	5.06	2.22
RubberWhale	9.89	9.69	9.87	10.98	4.57
Urban2	8.79	9.26	11.93	18.2	5.35
Urban3	13.74	13.64	15.96	33.16	10.85
Venus	11.29	10.77	13.03	19.4	9.63

Table 4.1: Comparative results over the Middlebury dataset. Average angular errors (AAE) calculated for each image pair in the dataset. AAE units are degrees. OF_{GC} is our optical flow method with graph cuts minimization and OF_{GS} is a modification using dense SIFT descriptors. VOF is the optical flow algorithm based on Brox et al. [BBPW04], SF is SIFT-Flow [LYT11] and SF is Patch-Flow [TS12]. Note that VOF is the only method that employ continuous optimization.

Avg. EE	OF_{GC}	OF_{GS}	SF	PF	VOF
Dimetrodon	0.21	0.14	0.37	0.43	0.17
Grove2	0.31	0.35	0.48	0.58	0.16
Grove3	1.01	1.06	0.97	1.20	0.66
Hydrangea	0.34	0.35	0.34	0.44	0.18
RubberWhale	0.30	0.30	0.33	0.36	0.13
Urban2	0.79	0.88	1.16	1.46	0.34
Urban3	1.21	1.19	1.20	2.50	0.84
Venus	0.46	0.49	0.46	0.74	0.40

Table 4.2: Comparative results over the Middlebury dataset. Average endpoint error (Avg. EE) calculated for each image pair in the dataset. OF_{GC} is our optical flow method with graph cuts minimization and OF_{GS} is a modification using dense SIFT descriptors. VOF is the optical flow algorithm based on Brox et al. [BBPW04], SF is SIFT-Flow [LYT11] and SF is Patch-Flow [TS12]. Note that VOF is the only method that employ continuous optimization.

4.4 Scene flow

The scene flow can be defined as the three dimensional (3D) motion of points in the world. The name scene flow is due to the work of Vedula et al. [VBR+99] where they pose the problem as an optical flow extension to 3D.

We compute a dense scene flow estimation, extending in this way the 3D motion estimation to each pixel and enforcing motion consistency through the smoothness constraint.

The most common assumptions in scene flow come from optical flow: brightness constancy and smooth variation of the flow field. However, the third dimension incorporates new information that aids to resolve ambiguities at object borders and occlusions. Thus, another restriction is posed as *structure constancy* or *depth consistency* which relates the depth of a scene point with its velocity in the 3D scene. In this work, we assume for this additional restriction, a locally rigid translational motion model, i.e., we neglect the rotational component of the 3D rigid body motion between two consecutive frames (see Section 2.1).

We propose a simple but effective formulation of the scene flow from a pair of RGB-D images. The 3D motion estimation problem can be posed as an energy minimization, like before for the optical flow, we start the formulation as a visual correspondence problem using 3D points and requiring geometric consistent.

The energy function is constrained in the 3D space using point clouds obtained by projecting image pixels to the world and associating its color intensity. For this end it is necessary to know the intrinsic parameters of the camera. Also, we assume that images were previously rectified (lens distortion was removed).

This formulation is a natural extension of the method presented in Section 4.3. First, we must transform the original problem into a label assignment problem. As we said before, we treat the scene flow estimation problem in 3D, thus first, we need to transform the input RGB-D images into point clouds.

Given a pair of RGB-D $\{I, \mathbf{Z}\}$ and $\{I', \mathbf{Z}'\}$ captured at times t and t', we define the point clouds $\mathcal{P}$ and $\mathcal{P}'$ as the sets:

$$\mathcal{P} = \{\mathbf{X} \mid \mathbf{x} \in \Omega_I,\ \mathbf{X} = \hat{\mathbf{M}}^{-1}(\mathbf{x}, \mathbf{Z}_\mathbf{x})\} \tag{4.15}$$

$$\mathcal{P}' = \{\mathbf{X}' \mid \mathbf{x}' \in \Omega_{I'},\ \mathbf{X}' = \hat{\mathbf{M}}^{-1}(\mathbf{x}', \mathbf{Z}'_\mathbf{x})\} \tag{4.16}$$

where Ω_I and $\Omega_{I'}$ are the supports (set of pixels) of images I and I', respectively, and $\hat{\mathbf{M}}^{-1}$ the inverse of the projective function (see Section 2.1), and $\mathbf{Z}_\mathbf{x}$ and $\mathbf{Z}'_{\mathbf{x}'}$ the depths at pixel $\mathbf{x}$ and $\mathbf{x}'$, respectively.

Given a finite set of labels $\mathcal{L} \in \mathbb{R}^3$ representing the space of all possible motion vectors in 3D, the goal is to find a labelling function $f : \mathcal{P} \to \mathcal{L}$ which minimizes the energy:

$$E(f) = \sum_{\mathbf{X} \in \mathcal{P}} D_\mathbf{X}(f_\mathbf{X}) + \sum_{\{\mathbf{X},\mathbf{Y}\} \in \mathcal{N}} V_{\mathbf{X},\mathbf{Y}}(f_\mathbf{X}, f_\mathbf{Y}) \tag{4.17}$$

where $\mathcal{N} \subseteq \mathcal{P} \times \mathcal{P}$ is any neighbourhood for point pairs in the 3D space.

The first term in Equation 4.17 is known as the data term E_{data}, and $D_\mathbf{X} : \mathcal{L} \to \mathbb{R}$ measures the cost of assigning the 3D label $f_\mathbf{X}$ to point $\mathbf{X} \in \mathbb{R}^3$ given the observed data. The second term, also known as the smoothness term E_{smooth}, where the function $V_{\mathbf{X},\mathbf{Y}} : \mathcal{L} \times \mathcal{L} \to \mathbb{R}$ must be a discontinuity-preserving function, i.e. a function that accounts for abrupt changes at object boundaries and avoids oversmoothing.

Note that $\mathcal{N}$ can be defined more generally in 3D than the usual 4-neighborhood over the image plane, including as neighbors of $\mathbf{X}$ all the 3D points lying in a sphere centered at $\mathbf{X}$ with radix r.

With the energy formulated as in Equation 4.5 we can use the general graph construction for the energy $E(f)$ of Kolmogorov et al. [KZ04] such that the cost of the minimum cut in G is the minimum of $E(f)$, and from which we can deduce the labelling function f.

The data term and the smoothness term are derived from the scene flow assumptions and are explained in the next sections.

4.4.1 Data term

In the scene flow problem we have a *brightness* consistency as in optical flow, since a point in the space should be projected with the same intensity

in different images (acquired at different times), and also we have a *structure* consistency given that the back-projections of these image pixels should be near in the 3D space. This ideal situation is not always met since points can be missing at the next time step due to occlusion or sampling errors, thus, we take the nearest neighbour in the next point cloud.

Thus, we constrain the scene flow using the *rigid translational flow model* which is a rigid body motion without rotation (see Section 2.1). In order to formulate the data term, we need to defined how is the data adjustment.

Given a point $\mathbf{X} = (X, Y, Z)$ at time t and $\mathbf{X}' = (X', Y', Z')$ the same point at time t', the instantaneous motion can be described as:

$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T} \tag{4.18}$$

where $\mathbf{R}$ and $\mathbf{T}$ are the rotational and translational components of the 3D motion. Under the *rigid translation flow model* we assume that the contribution of the inter-frame rotation is negligible, so $\mathbf{R}$ is the identity and $\mathbf{T}$ will be our motion vector $\mathbf{V} = (V_x, V_y, V_z)$. Then

$$\mathbf{X}' = \mathbf{X} + \mathbf{V} \tag{4.19}$$

This rigid translation model can be used to describe the flow of all visible 3D points in a scene performing motions that can be locally approximated by translations. There are many other motion models that can describe the inter-frame 3D motion [BAHH92]. In this work, we chose the simpler but effective motion model to verify the correctness of our approach.

Let I and I' two images taken at times t and t', respectively, under the *brightness constancy assumption*, a point $\mathbf{X}$ that suffers a translational motion $\mathbf{V}$ from t to t', such that $\mathbf{X}' = \mathbf{X} + \mathbf{V}$, it is projected with the same intensity on both images:

$$I_{\mathbf{x}} - I'_{\mathbf{x}'} = 0 \tag{4.20}$$

where $\hat{\mathbf{M}}$ is the projective function that maps 3D points to the image plane, so $\hat{\mathbf{M}}(\mathbf{X}) = \mathbf{x}$ and $\hat{\mathbf{M}}(\mathbf{X}') = \mathbf{x}'$ and $I_{\mathbf{x}}$ and $I'_{\mathbf{x}'}$ its corresponding image intensities.

Therefore, we propose a novel data term for scene flow which is composed as the sum of two components accounting for intensity and structure,

defined as following:

$$D_{\mathbf{X}}(\mathbf{V}) = |I_{\hat{\mathbf{M}}(\mathbf{X})} - I'_{\hat{\mathbf{M}}(\mathbf{X}'_{nn})}| + \lambda||\mathbf{X} + \mathbf{V} - \mathbf{X}'_{nn}||_2 \tag{4.21}$$

where $\lambda \in \mathbb{R}$ is a constant that imposes a trade off between structure and intensity and $\mathbf{X}'_{nn}$ is the nearest neighbour of $\mathbf{X} + \mathbf{V}$ in the $\mathcal{P}'$:

$$\mathbf{X}'_{nn} = \arg\min_{\mathbf{X}' \in \mathcal{P}'} ||\mathbf{X} + \mathbf{V} - \mathbf{X}'||_2 \tag{4.22}$$

Figure 4.5 shows a diagram of the 3D motion model for scene flow.

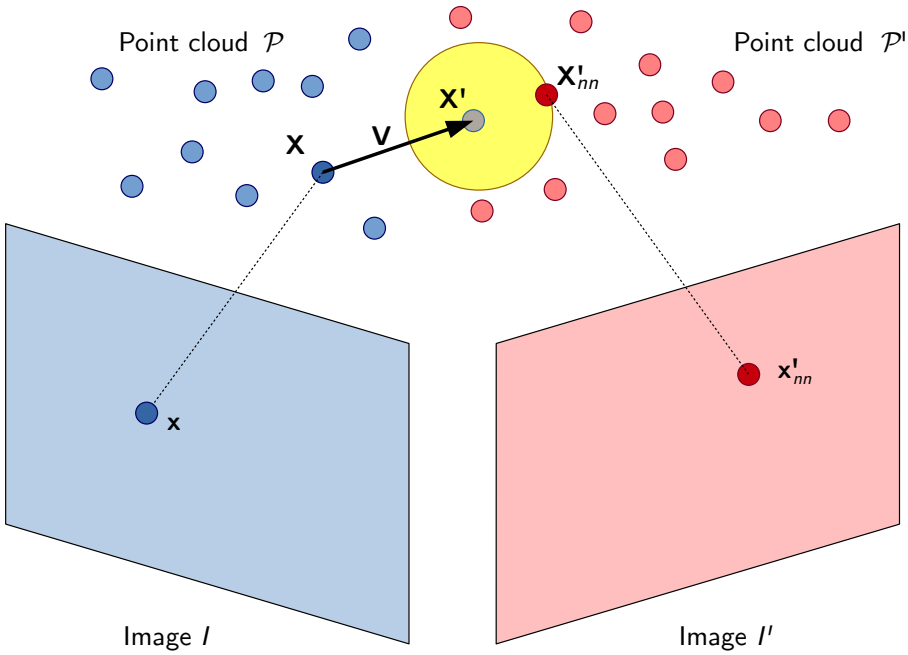


Figure 4.5: Data consistency for our scene flow energy (see text).

4.4.2 Smoothness term

The smoothness term or interaction penalty $V_{\mathbf{X},\mathbf{Y}}$ as in the case of optical flow must to be defined according to a motion coherency, i.e. neighboring points will have similar labels/motion vectors:

$$V_{\mathbf{X},\mathbf{Y}}(f_{\mathbf{X}}, f_{\mathbf{Y}}) = k_1 \min(k_2, C^s(f_{\mathbf{X}}, f_{\mathbf{Y}})) \tag{4.23}$$

where $k_1, K_2 \in \mathbb{R}$ are constants and $C^s(f_{\mathbf{X}}, f_{\mathbf{Y}}) = ||f_{\mathbf{X}} - f_{\mathbf{Y}}||_2$ now measures distances between 3D motion vectors. Note that contextual information to

account for texture can also be added using an edge detector over the image plane or simply intensity differences (see Section 4.3.2).

When we are dealing with 3D motion, we can define a spatial neighbour more general than in the image plane, including as neighbours of an 3D point $\mathbf{X}$ all the points lying in a 3D sphere centred at $\mathbf{X}$.

4.4.3 Scene flow computation

In this section, we describe the computation of scene flow using energy minimization via graph cuts.

Given a pair of RGB-D $\{I, \mathbf{Z}\}$ and $\{I', \mathbf{Z}'\}$ captured at times t and t', respectvely, and its corresponding point clouds $\mathcal{P}$ and $\mathcal{P}'$ (defined in Equation 4.15), and a finite set of labels $\mathcal{L} \in \mathbb{R}^3$ representing the space of all possible motion vectors in 3D, the goal is to find a labelling function $f : \mathcal{P} \rightarrow \mathcal{L}$ which minimizes the energy:

$$E(f) = \sum_{\mathbf{X} \in \mathcal{P}} |I_{\hat{\mathbf{M}}(\mathbf{X})} - I'_{\hat{\mathbf{M}}(\mathbf{X}'_{nn})}| + \lambda ||\mathbf{X} + f_{\mathbf{X}} - \mathbf{X}'_{nn}||_2 + \sum_{\mathbf{X},\mathbf{Y} \in \mathcal{N}} k_1 \min(k_2, C^s(f_{\mathbf{X}}, f_{\mathbf{Y}})) \tag{4.24}$$

where $\lambda, k_1, k_2 \in \mathbb{R}$ are constants that imposes a trade off between structure, intensity and smoothness of the solution, $C^s(f_{\mathbf{X}}, f_{\mathbf{Y}}) = ||f_{\mathbf{X}} - f_{\mathbf{Y}}||_2$, $\hat{\mathbf{M}}$ is the projective function that maps 3D points to the image plane. $\mathbf{X}'_{nn}$ is the nearest neighbour of $\mathbf{X} + f_{\mathbf{X}}$ in $\mathcal{P}'$ defined as:

$$\mathbf{X}'_{nn} = \arg\min_{\mathbf{X}' \in \mathcal{P}'} ||\mathbf{X} + f_{\mathbf{X}} - \mathbf{X}'||$$

.

For this end, we follow a very similar implementation as in Section 4.3.3 with the exception that labels represent 3D vectors.

Initialization. A label space $\mathcal{L}$ is chosen to account for possible 3D motion vectors, defining a range value, which from labels are sampled.

Decoupled motion estimation. The estimation of the scene flow using graph cuts over the Equation 4.24 is carried out decoupling the motion in

X, Y and Z axes, one at time in three consecutive step. First, the minimization is performed only in X, using unidimensional labels, assuming no motion in Y and Z. Then, the optimization is performed over Y assuming the motion in X is fixed and Z has no motion. Finally, the operation is repeated for Z while fixing motions in X and Y.

Neighbourhood definition The pairwise neighbourhood $\mathcal{N}$ for the interaction penalty $V_{\mathbf{X},\mathbf{Y}}$, is defined for each point $\mathbf{X}$ as all the point pairs lying in a sphere of radix r centered at $\mathbf{X}$, the radix r is a parameter of the method which imposes spatial coherency directly over the point cloud.

Experimental results showed us that the best norm to compare intensity values is the L_1 norm. If images have color information, $I_{\mathbf{x}}$ is an 3-component RGB vector.

Our implementation employs the code provided by Olga Veksler and Andrew Delong[3], described in the articles: [BK04, BVZ01a, KZ04].

4.4.4 Experimental results

In this section, we present experimental results of our scene flow method and in order to assess the performance of our method, we carry quantitative evaluation over the Middlebury stereo [SS03, SS02].

Sequences from the Middlebury stereo [SS03, SS02] dataset have been used as a scene flow ground truth in order to assess quantitatively the estimated 3D flow. Many authors [HB11, BMK13, HD07, QBDC14] have presented their numerical results over this dataset and thus we can use them to compare with our work.

Finally, we test our algorithm with sequences captured with a RGB-D sensor in order to show the performance in a real world scenario. Because of the lack of ground truth for these sequences we present discussions about the quality of the results.

[3] http://vision.csd.uwo.ca/code/

Middlebury stereo dataset

This dataset consist of two stereo sequences with depth (disparity) ground truth: "Cones" and "Teddy". They have complex geometry and textured and non-textured regions [SS03]. Also, we use the "Venus" stereo sequence from [SS02] which is a piecewise planar scene. Figure 4.6 shows the sequences analysed in this section. The first image of the sequence (left) and the corresponding ground truth flow (right) are displayed. Black pixels in the ground truth lack of disparity information.

Each image sequence was acquired in a multi-baseline stereo configuration, with the camera pointing perpendicular to the scene, consisting of nine color images with ground truth disparity map for two of them.

Since images are rectified, motion is present only in X direction with a displacement due to the stereo baseline. Then, the ground truth disparity for the primary image can be seen as the ground truth optical flow motion field. Moreover, we can transform it in a scene flow ground truth using the camera intrinsic and the depth value (disparity) at each pixel.

This adaptation from stereo ground truth to scene flow ground truth was introduced by Huguet et al. [HD07] and since then has been widely used to assess the performance of scene flow algorithms.

The intrinsics parameters of the camera are not reported with the dataset, however, they are computed as in Basha et al. [BMK13], defining a field of view (FOV) of 30 degrees, and camera center as the half of the width and height of the image (see Equations 4.25 and 4.26).

$$f_x = \frac{w}{2\tan(FOV/2)} \qquad f_y = \frac{h}{2\tan(FOV/2)} \tag{4.25}$$

$$c_x = \frac{w}{2} \qquad c_y = \frac{h}{2} \tag{4.26}$$

where f_x and f_y are the focal lengths, c_x and c_y the camera centers and w and h the width and height of the image, respectively.

Note that motion is purely horizontal with different magnitudes at different depths

We use quarter-size[4] version with a size of 450 x 375 pixels and take

[4]The Middlebury dataset also have half-size and full size version of all the images.

(a) Cones

(b) Teddy

(c) Venus

Figure 4.6: Middlebury stereo dataset. Left: The first image of the sequence. Right: ground truth flow.

images 2 and 6 of the sequence with their corresponding disparity as the pair of consecutive images I and I'.

Since the motion in the sequences is purely horizontal we do not take advantage of this knowledge and explore motion in the three dimensions.

Figure 4.7 show the 3D motion field obtained with our scene flow method. The 3D vectors are projected on the plane and depicted using the optical flow color convention (see Figure 4.4). The motion in the Z direction is represented with a "cold to warm" color scale. Note that for cones and teddy sequences, our method gives an accurate estimate of the motion field specially at the borders of the objects where discontinuities often occur. Also, our method efficiently track the motion for small objects in the scene. However, for nearby objects in depth our method fails and treat them as one. This is due to the motion vector discretization. To avoid this artifact the label space should be increased in order to use a more precise motion vector for the method. This effect is better visualized in the venus sequence where a staircase effect can be appreciated in the planar regions with smooth variations in motion an depth.

For the estimated velocities in depth or Z direction, our method only fails in the occluded regions of the sequences and gives correct estimations for non-occluded regions.

Figure 4.8 shows a 3D visualization of our scene flow estimation over a region of interest (ROI) of the teddy sequence. Motion vectors are displayed as arrows over each point of the point cloud. In Figure 4.9 we show the point cloud for the ROI in the first image translated according to the estimated 3D motion compared with the point cloud in the second image. Note that both point clouds are very similar for non-occluded point showing the precision of our method.

Comparison with other scene flow methods

For comparison with other scene flow algorithms, best performing methods of the state of the art were selected.

Basha et al. [BMK13] and Huguet et al. [HD07] are stereo-based method which computes disparity and motion at the same time from multiple view.

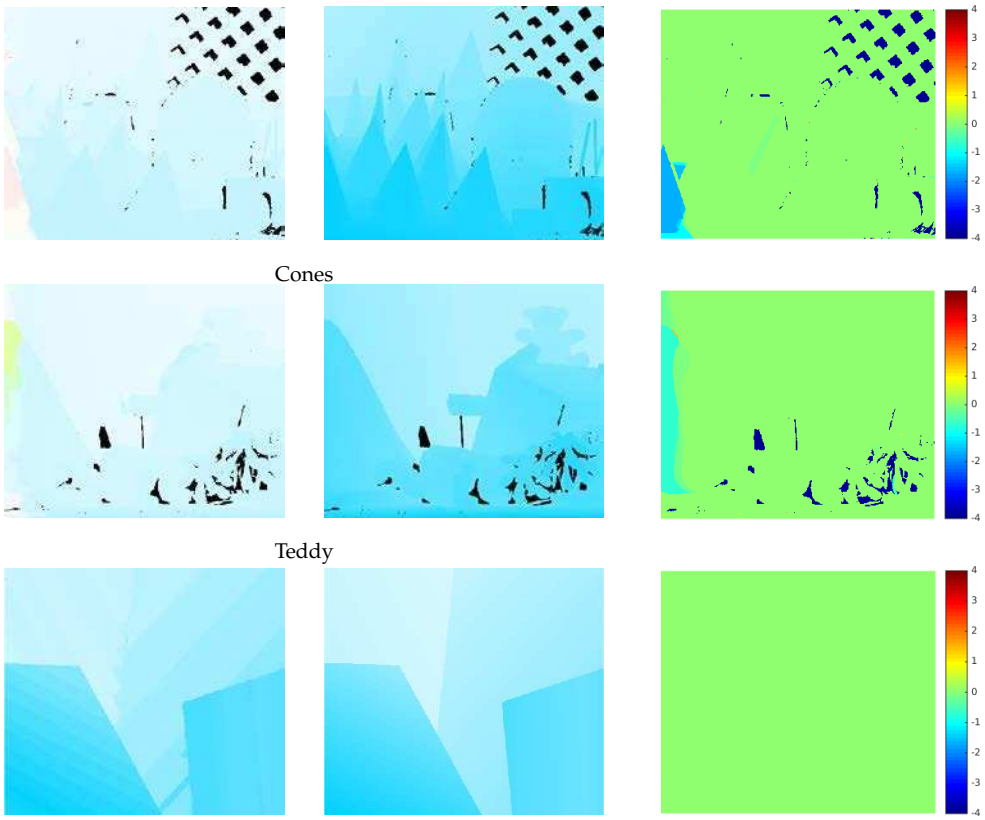
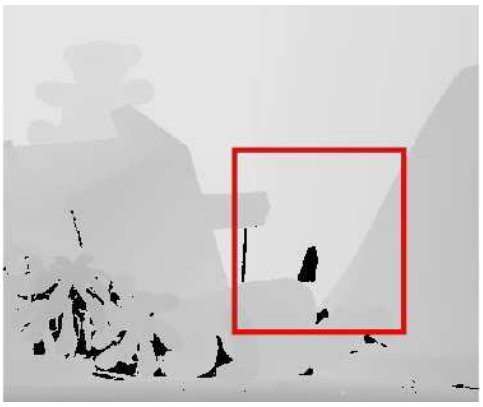
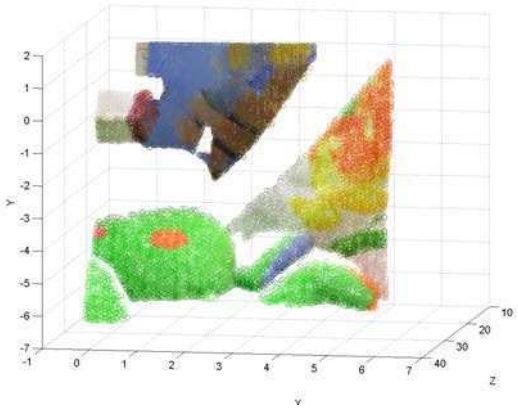


Figure 4.7: Scene flow estimations for the Middlebury dataset.

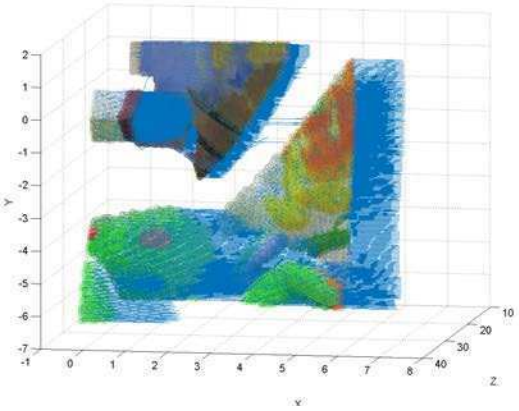
(a) First image.



(b) Depth image.

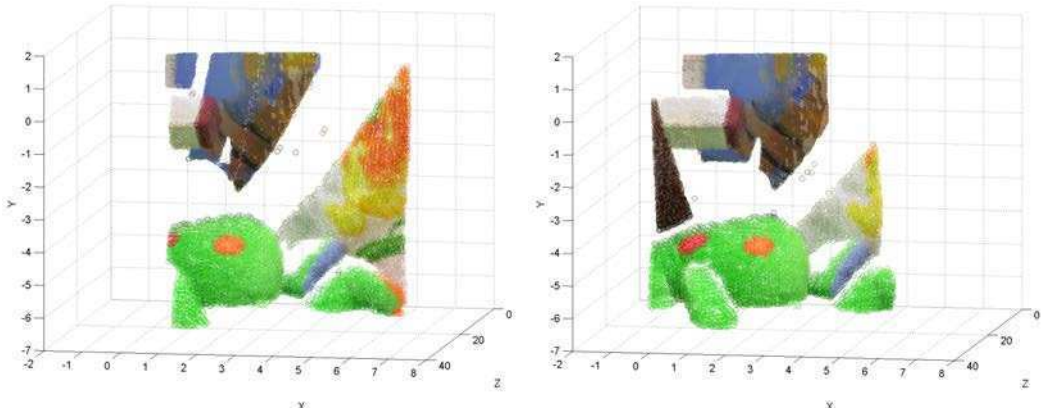


(c) Point cloud.



(d) Motion vector for each 3D point.

Figure 4.8: Example of 3D motion estimation over a ROI (in red) for the teddy sequence. The image was mirrored horizontally for better viewing.



(a) Translated 3D points. (b) Point cloud in the second image.

Figure 4.9: Translated 3D points in the first image according to their estimated motion compared with the point cloud in the second image. Point clouds correspond to the ROI of Figure 4.8.

We select the results that authors have reported for two views. The work of Quiroga et al. [QBDC14] employs a global minimization based on local motion estimations

The work of Hornáček et al. [HFR14] is the most similar to ours, since perform 3D motion estimation using matching approach with a further regularization based on graph cuts.

For a review of these method and related work see Section 2.5.

We project the estimated 3D motion field to the image plane and compute the root-mean-square error (RMS) and average angular error (AAE) (see Equations 4.12 and 4.13 from Section 4.3.4).

We initialize our method with a lineal discretization of the flow field based of maximum and minimum range values.

Table 4.3 shows results from the evaluation of scene flow methods described above.

The methods of Basha et al. [BMK13], Huguet et al. [HD07], and Quiroga et al. [QBDC14] are the top best performing since they employ a variational regularization of the flow field and so they are benefited from these se-

	Cones		Teddy		Venus	
	RMS	AAE	RMS	AAE	RMS	AAE
Basha et al. [BMK13]	0.58	0.39	0.57	1.01	**0.16**	1.58
Huguet et al. [HD07]	1.10	0.69	1.25	0.51	0.31	0.98
Hadfield et al. [HB11]	0.59	1.61	0.52	1.36	0.72	2.62
Quiroga et al. [QBDC14]	**0.45**	0.37	0.49	0.46	–	–
Hornacek et al. [HFR14]	0.54	0.52	**0.35**	0.15	0.26	0.53
Our method	0.90	**0.08**	1.07	**0.06**	0.40	**0.42**

Table 4.3: Errors for the projected optical flow from the estimated scene flow. Root mean square error (RMS) and average angular error (AAE)

quences because of their relatively small motion.

Hadfield et al. [HB11] achieves very good results in terms of RMS but not for the angular error since the probabilistic nature of this method tend to fail when there is no enough observed data. Hornacek et al. [HFR14] is the best performing method in term of RMS and AAE.

In terms of RMS error our method does not perform very well. This is due to that the space of possible motions is discretized and therefore motion vectors approximate roughly the ground truth. However, in spite of being a method inherently discrete, values of average angular errors (AAE) are the lowest because the orientations are consistent with the ground truth.

4.5 Conclusions

In this chapter we studied two main motion estimation techniques the optical flow and the scene flow. The optical flow is the 2D estimation over the image plane and the scene flow is the computation of the real world 3D motion with the help of a depth sensor.

These two method were tackled in the graph cuts framework which is a discrete optimization method that has been used to the minimization of energy functions commonly present in computer vision.

To solve these problems as a problem of energy minimization via graph

cuts is very challenging and requires very careful implementations to cope with the limitations of discrete optimization methods.

In this context, we employed a different formulation than most common methods of optical flow that use a variational optimization framework to minimize the optical flow energy function. Also, we presented a method that use different approaches, such as coarse to fine and a decoupled minimization which showed to be very helpful in order to obtain an accurate solution.

For the case of scene-flow we proposed a novel formulation of the energy function to minimize via graph cuts which, for the best of our knowledge, has not been addressed so far.

Quantitative results on a dataset with ground truth motion shown that the presented optical flow method is competitive with most common approaches for motion estimation using discrete optimization. The results for the scene flow showed that our method without much post-processing or 3D filtering of the motion field obtained very good 3d motion estimations, comparable with the best state of the art methods for scene flow.

On the downside, both methods are slow which make impossible their use in real-time applications. Also, the choice of the label set greatly influences the results as it imposes a discretization of the motion vector's space that if it not close enough to the real motion flow to estimate, outputted results are very degraded.

As a future work may be mentioned for both methods that the study of a technique that allows to define automatically, with a sound criterion, the best set of labels in order to achieve a good result without having to explore a good labelset manually.

Also, it would be very useful for both methods to have a parallel or a GPU (Graphic Processing Units) implementation in order to maximize the hardware available nowadays and thus achieving an computationally efficient motion estimation.

4.6 Resumen

Los movimientos de los objetos del mundo real, capturados por una cámara, pueden ser muy difíciles de describir. Los objetos del mundo real suelen presentar deformaciones y movimientos muy complejos que no suelen ser bien representados con un único modelo de movimiento global.

Este problema es generalmente abordado utilizando estimaciones densas para el movimiento de una escena, donde cada punto capturado por un sensor tiene su propio movimiento, pero al mismo tiempo tiene que ser coherente con sus puntos vecinos. Por lo tanto, de esta manera, podemos ser capaces de tratar con datos faltantes y grandes desplazamientos, problemas que a menudo se producen en escenarios del mundo real.

El problema de *optical flow* o flujo óptico se ocupa de la estimación del campo de vectores de movimiento proyectado en el plano de la imagen, es decir, un campo de movimiento 2D, inducido por un movimiento en 3D de la escena. Este movimiento inducido en el plano de la imagen se conoce como el movimiento aparente.

Cuando una medición de la profundidad de los objetos en el mundo está disponible, dada por un sensor RGB-D o una cámara estéreo, la estimación del *scene flow* o flujo de la escena es un problema más completo, que nos permite el cálculo del movimiento observado en 3D de una escena del mundo real.

Estos dos métodos se han abordado en el presente capítulo mediante *graph cuts*, que es un método de optimización discreta utilizado para minimizar funciones de energía comúnmente presentes en visión por computadora.

Además, presentamos un método que utiliza diferentes enfoques, como el *coarse-to-fine* y una minimización disociada que mostró ser muy útil con el fin de obtener una solución más precisa.

Los resultados cuantitativos sobre un conjunto de datos con *ground truth* muestran que el método de flujo óptico que se presenta es competitivo con la mayoría de los enfoques comunes para la estimación de movimiento mediante optimización discreta. Los resultados para el flujo de la escena mos-

traron que nuestro método sin demasiado post-procesamiento o filtrado 3D sobre el campo de vectores, obtiene muy buenas estimaciones de movimiento 3D, comparables con los mejores métodos de estado del arte. En el lado negativo, ambos métodos son lentos lo que hacen imposible su uso en aplicaciones de tiempo real. Además, la elección del conjunto de etiquetas influye en gran medida en los resultados, ya que impone una discretización del espacio del campo de vectores de movimiento que si no está suficientemente cerca para del campo de vectores de movimiento real de estimar, los resultados son perjudicados.

Como un trabajo futuro se puede mencionar para ambos métodos que el estudio de una técnica que permita definir de forma automática, con un criterio de adecuado, el mejor conjunto de etiquetas con el fin de lograr un buen resultado sin tener que explorarlo manualmente.

Además, sería muy útil para ambos métodos tener una implementación GPU (Graphic Processing Units) o paralela o con el fin de maximizar el hardware disponibles hoy en día y por lo tanto lograr una estimación de movimiento computacionalmente eficiente.

Chapter 5

Conclusions and perspective

Video motion analysis is a very challenging and a very important task in computer vision and image processing. The study of image sequences gives us a much more valuable information than static images as theirself. A proper modeling of motion in video brings opportunities to an endless number of applications and automatic tools for everyday use, such as human action recognition, hand gesture recognition, video surveillance and monitoring, among others.

The motion captured by a camera is a 2D projection over the image plane of a 3D motion from the real world, and unless another view of the scene or a depth measurement are presented, the 3D motion cannot be computed.

In this thesis, we deal with 2D motion estimation and also 3D motion estimations when an RGB-D sensor is available. For this end, two main lines of study are proposed: statistical models of motion using dynamic textures and 2D and 3D dense motion estimations using graph cuts optimization.

In chapter 3 we tackle the motion estimation with dynamic textures which models visual motions as a statistical visual process using a generative video model [BL+07], where the observed texture is a time-varying process commanded by a hidden state process.

In this context, we introduced a decoupled learning process for Mixture of Dynamic Textures (MDT) that allow us learning the parameters of the statistical distribution of an MDT as a separated process and later use it for segmentation or classification task. Decoupling the learning step has

been shown to be advantageous in terms of computing performance. And thus achieving substantial improvements that are further developed using graphics processing units (GPU) and a specific implementation matrix inversion implementation for symmetric positive definite matrices, a repeated operation considered as a "bottleneck" in this type of statistical models. However, this advantage decreases with the number of visual components K to be classified but computer times are amortized when video sizes increases.

Also, the MDT model is extended to analyze RGB-D sequences using appearance and structure as independent cues for the statistical model. The, both estimations are combined together to produce a motion segmentation that prioritize changes in texture and depth at the same time. For the best of our knowledge the MDT model has not been applied to RGB-D sequences so far.

As future work, we are interested in study a segmentation algorithm based on layered dynamic textures model [CV09a]. This method appears to be better in quality results, but has a more complex model which makes it more difficult to implement.

The other line of study of this thesis, that is complementary to dynamic textures, seeks to estimate the instantaneous motion between a consecutive pair of images for which dense motion estimation techniques were studies such as optical flow for 2D and scene flow for 3D. In order to deal with complex 3D scenes composed of multiple objects and different motions scales and also to resolve ambiguities that often occur due to occlusions and object deformations, dense estimations are required. The optical flow and scene flow methods were studied within the graph cuts framework (Chapter 4) which is a discrete optimization method that has been used to the minimization of energy functions commonly present in computer vision. In order to pose these problems as an energy minimization problem and then efficiently minimize it via graph cuts very challenging implementations must to be done in order to cope with the limitations of discrete optimization methods. This is a different approach to optical flow and scene flow because most common methods employ an energy minimization using vari-

ational optimization methods that suffers from oversmoothing and occlusion handling, problems that are best explained using discrete optimization and discontinuity-preserving regularization functions. Also, we presented a method that use different approaches, such as coarse to fine and a decoupled minimization which showed to be very helpful in order to obtain an accurate solution. We achieved competitive results against similar methods that solve the optical flow using discrete optimization.

The scene flow problem is resolved in the 3D space using geometry consistency and brightness constancy over point clouds which is a novel formulation usen in the energy function to minimize via graph cuts which. For the best of our knowledge, graph cuts for scene flow estimation has not been addressed so far. The results for the scene flow showed that our method without much post-processing or 3D filtering of the motion field obtained very good 3d motion estimations, comparable with the best state of the art methods for scene flow.

As a drawback, both methods are very demanding in terms of computer performance which make them impracticable for real-time applications. Also, it would be very useful for both methods to have efficient implementations that take advantage of parallel hardware ubiquitous nowadays is consumer laptop computers.

Additionally, the choice of the labelset greatly influences the results as it imposes a discretization of the motion vector's space that if it not close enough to the real motion flow to estimate, outputted results are very degraded. Thus, the study of a methods that allow us to define automatically, with a sound criterion, a good discretization of the space of motion vector is required.

Finally, the analysis of all these methods on RGB-D sequences is a contribution that result very useful for the scientific community because of the current rise in the use of RGB-D sensors.

Capítulo 5

Conclusiones y perspectiva

El análisis de movimiento en vídeo es un gran desafío y una tarea muy importante en visión computadora y procesamiento de imágenes.

El estudio sobre secuencias de imágenes nos da una información mucho más valiosa que las imágenes estáticas por si solas. Un modelado adecuado del movimiento en vídeo nos trae un sinfín de oportunidades para aplicaciones y herramientas automáticas para el uso diario, tales como el reconocimiento de acciones humanas, el reconocimiento de gestos, análisis de cámaras de vigilancia y monitoreo, entre otros.

El movimiento capturado por una cámara es una proyección 2D sobre el plano de la imagen de un movimiento 3D del mundo real. A menos que se presente otra vista de la escena o una medición de la profundidad, el movimiento 3D no se puede calcular exactamente.

En esta tesis, nos ocupamos de la estimación de movimiento 2D y en 3D cuando un sensor RGB-D está disponible. Para este fin, se proponen dos líneas principales de estudio: procesos estadísticos visuales de movimiento utilizando texturas dinámicas y estimación densa de movimiento 2D y 3D utilizando *graph cuts*.

En el Capítulo 3 abordamos la estimación de movimiento con texturas dinámicas donde modelamos los procesos visuales en video como un proceso estadístico visual utilizando un modelo generativo de vídeo, donde la textura observada es un proceso variable en el tiempo al mando de un proceso de estado oculto.

En este contexto, se introdujo un proceso de aprendizaje disociado para mixtura de texturas dinámicas (MDT) que nos permiten conocer los parámetros de la distribución estadística de una MDT como un proceso aparte y luego utilizarlo para tareas de segmentación o clasificación. La disociación de la etapa de aprendizaje ha demostrado que es ventajosa en términos de rendimiento computacional. Logrando así mejoras sustanciales que se desarrollan utilizando GPU (Graphic Processing Units) y una implementación específica para la inversa de una matriz simétrica definida positiva que es una operación considerada como un çuello de botella.[en] este tipo de modelos estadísticos. Sin embargo, esta ventaja disminuye con el número de componentes visuales a clasificar, aunque los tiempo de de ejecución se amortizan cuando el tamaño del video aumenta.

Además, extendimos el modelo de MDT para analizar secuencias RGB-D utilizando apariencia y estructura como independientemente para el modelo estadístico, luego ambas estimaciones se combinan para producir una segmentación de movimiento que prioriza variaciones en textura y profundidad al mismo tiempo. Para lo mejor de nuestro conocimiento el modelo MDT no se ha aplicado a secuencias de RGB-D hasta el momento.

Como trabajo futuro, estamos interesados en el estudio de un algoritmo de segmentación basado en el modelo dinámico de texturas en capas [CV09a]. Este método parece ser mejor en los resultados de calidad, pero tiene un modelo estadístico más complejo que lo hace más difícil de implementar.

La otra línea de estudio de esta tesis, que es complementaria a texturas dinámicas, trata de estimar el movimiento instantáneo entre un par consecutivo de imágenes. Donde nos centramos en las técnicas de estimación de movimiento denso mediante *optical flow* o flujo óptico para 2D y el *scene flow* o flujo de la escena para 3D. Con el fin de hacer frente a escenas 3D complejas compuestas de múltiples objetos y movimientos diferentes y también para resolver las ambigüedades que a menudo se producen debido a las oclusiones y deformaciones de los objetos, se requieren estimaciones densas. Los métodos de flujo óptico y de flujo de la escena fueron estudiados en el marco cortes *graph cuts* (Capítulo 4) que es un método de optimización

discreta que se ha utilizado para la minimización de funciones de energía comúnmente presentes en visión por computadora. Con el fin de plantear estos problemas como un problema de minimización de energía y resolverlos eficientemente a través de una minimización con *graph cuts* se deben realizar implementaciones desafiantes con el fin de lidiar con las limitaciones de los métodos de optimización discreta. Este es un enfoque diferente para el flujo óptico y el flujo de la escena dado que la mayoría de los métodos comunes emplean una minimización de energía usando métodos de optimización variacional los cuales sufren de sobre-suavizado (*oversmoothing*) y falta manejo de oclusiones, problemas que se explican mejor mediante optimización discreta y funciones de regularización que preservan discontinuidades. Además, presentamos un método que utiliza diferentes enfoques, como el *coarse-to-fine* y una minimización desacoplada, que mostró a ser muy útil con el fin de obtener una solución más precisa. Hemos logrado resultados competitivos contra métodos similares que resuelven el flujo óptico mediante optimización discreta.

El problema de flujo de la escena se resuelve en el espacio 3D usando las propiedades de consistencia geométrica y de constancia de brillo sobre las nubes de puntos, que es una novedosa formulación de la función de energía a minimizar por *graph cuts*. Para lo mejor de nuestro conocimiento, *graph cuts* en la estimación del flujo de la escena no se ha abordado hasta el momento. Los resultados para el flujo de la escena mostraron que nuestro método sin mucho post-procesamiento o filtrado 3D del campo de movimiento obtiene muy buenas estimaciones de movimiento 3D, comparables con los mejores métodos del estado del arte en flujo de la escena.

Como desventaja, ambos métodos son muy exigentes en términos de rendimiento computacional que los hacen impracticable para aplicaciones en tiempo real.

Como trabajo futuro, sería muy útil para ambos métodos tener una implementación eficiente que se aproveche del hardware paralelo omnipresente hoy en día es computadoras portátiles.

Además, la elección del conjunto de etiquetas influye mucho en los resultados, ya que impone una discretización del espacio del campo de vectores

de movimiento que si no está lo suficientemente cerca del flujo de movimiento real a estimar, los resultados son perjudicados. Por lo tanto, el estudio de métodos que nos permitan definir de forma automática, con un criterio adecuado, es un requerimiento para una buena discretización del espacio de vectores de movimiento.

Por último, el análisis de todos estos métodos en secuencias RGB-D es una contribución que resulta de gran utilidad para la comunidad científica debido al creciente uso de sensores RGB-D.

Bibliography

[BAHH92] James R Bergen, Patrick Anandan, Keith J Hanna, and Rajesh Hingorani. Hierarchical model-based motion estimation. In *Computer Vision – ECCV92*, pages 237–252. Springer, 1992.

[BB95] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995.

[BBM09] Thomas Brox, Christoph Bregler, and Jagannath Malik. Large displacement optical flow. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 41–48. IEEE, 2009.

[BBPW04] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *Computer Vision-ECCV 2004*, pages 25–36. Springer, 2004.

[BD96] Aaron Bobick and James Davis. Real-time recognition of activity using temporal templates. In *Applications of Computer Vision, 1996. WACV'96., Proceedings 3rd IEEE Workshop on*, pages 39–42. IEEE, 1996.

[BD02] Gary R Bradski and James W Davis. Motion segmentation and pose recognition with motion history gradients. *Machine Vision and Applications*, 13(3):174–184, 2002.

[BFB94] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994.

[BHF+15] Mariano Bianchi, Nadia Heredia, Francisco Gómez Fernández, Alvaro Pardo, and Marta Mejail. Two applications of RGB-D descriptors in computer vision. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications - 20th Iberoamerican Congress, CIARP 2015, Montevideo, Uruguay, November 9-12, 2015, Proceedings*, pages 236–244, 2015.

[Bis06] Christopher M Bishop. *Pattern Recognition*, volume 89. Machine Learning, 2006.

[BJ01] Yuri Y Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001.

[BK04] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, September 2004.

[BKK+13] Anders Glent Buch, Daniel Kraft, Joni-Kristian Kamarainen, Henrik Gordon Petersen, and Norbert Kruger. Pose estimation using local structure-specific shape and appearance context. In *Int. Conf. Robotics and Automation (ICRA)*, pages 2080–2087. IEEE, 2013.

[BL+07] Christopher M Bishop, Julia Lasserre, et al. Generative or discriminative? getting the best of both worlds. *Bayesian statistics*, 8:3–24, 2007.

[BM92] P.J. Besl and Neil D. McKay. A method for registration of 3-d

shapes. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 14(2):239–256, 1992.

[BM04] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[BM11] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2011.

[BMK10] T. Basha, Y. Moses, and N. Kiryati. Multi-view scene flow estimation: A view centered variational approach. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1506–1513. IEEE, June 2010.

[BMK13] Tali Basha, Yael Moses, and Nahum Kiryati. Multi-view scene flow estimation: A view centered variational approach. *International journal of computer vision*, 101(1):6–21, 2013.

[Bou12] R Bouckaert. Matrix inverse with cuda and cublas. http://www.cs.waikato.ac.nz/~remco/, 2012.

[BP98] Jean-Yves Bouguet and Pietro Perona. Camera calibration from points and lines in dual-space geometry. In *5th European Conference on Computer Vision, Freiburg, Germany*, 1998.

[BSFG09] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009.

[BSL+11] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[BVZ01a] Yuri Boykov, Olga Veksler, and Ramin Zabih. Efficient approximate energy minimization via graph cuts. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 20(12):1222–1239, November 2001.

[BVZ01b] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[BWF+05] Andrés Bruhn, Joachim Weickert, Christian Feddern, Timo Kohlberger, and Christoph Schnorr. Variational optical flow computation in real time. *Image Processing, IEEE Transactions on*, 14(5):608–615, 2005.

[BWKS05] Andrés Bruhn, Joachim Weickert, Timo Kohlberger, and Christoph Schnörr. Discontinuity-preserving computation of variational optic flow in real-time. In *Scale Space and PDE Methods in Computer Vision*, pages 279–290. Springer, 2005.

[BWS05] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[CG97] Boris V Cherkassky and Andrew V Goldberg. On implementing the push—relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997.

[CGN14] Haiyang Chao, Yu Gu, and Marcello Napolitano. A survey of optical flow techniques for robotics navigation applications. *Journal of Intelligent & Robotic Systems*, 73(1-4):361–372, 2014.

[CGZZ10] Qin Cai, David Gallup, Cha Zhang, and Zhengyou Zhang. 3d deformable face tracking with a commodity depth camera. In *ECCV 2010*, pages 229–242. Springer, 2010.

[Cha09] A. Chan. Synthetic dynamic texture segmentation database. http://www.svcl.ucsd.edu/projects/motiondytex/db/dytex_synthdb.zip, July 2009.

[CLO+12] Michael Calonder, Vincent Lepetit, Mustafa Ozuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua. Brief: Computing a local binary descriptor very fast. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1281–1298, 2012.

[CNM83] Stuart K. Card, Allen Newell, and Thomas P. Moran. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1983.

[Coo08] Tristrom Cooke. Two applications of graph-cuts to image processing. In *Digital Image Computing: Techniques and Applications (DICTA), 2008*, pages 498–504. IEEE, 2008.

[Cor09] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[CSS06] Ralph Costantini, Luciano Sbaiz, and Sabine Susstrunk. Dynamic texture synthesis: Compact models based on luminance-chrominance color representation. In *Image Processing, 2006 IEEE International Conference on*, pages 2085–2088. IEEE, 2006.

[CV05a] Antoni B Chan and Nuno Vasconcelos. Classification and retrieval of traffic video using auto-regressive stochastic processes. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 771–776. IEEE, 2005.

[CV05b] Antoni B Chan and Nuno Vasconcelos. Probabilistic kernels for the classification of auto-regressive visual processes. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 846–851. IEEE, 2005.

[CV08] Antoni B Chan and Nuno Vasconcelos. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(5):909–926, 2008.

[CV09a] Antoni B Chan and Nuno Vasconcelos. Layered dynamic textures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(10):1862–1879, 2009.

[CV09b] Antoni B Chan and Nuno Vasconcelos. Variational layered dynamic textures. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1062–1069. IEEE, 2009.

[CZS+13] Jie Chen, Guoying Zhao, Mikko Salo, Esa Rahtu, and Matti Pietikäinen. Automatic dynamic texture segmentation using local descriptors and optical flow. *Image Processing, IEEE Transactions on*, 22(1):326–339, 2013.

[DCFS03] Gianfranco Doretto, Daniel Cremers, Paolo Favaro, and Stefano Soatto. Dynamic texture segmentation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1236–1242. IEEE, 2003.

[DCWS03] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109, 2003.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[Dor05] Gianfranco Doretto. *DYNAMIC TEXTURES: modeling, learning, synthesis, animation, segmentation, and recognition*. University of California at Los Angeles, 2005.

[DS03] Gianfranco Doretto and Stefano Soatto. Editable dynamic textures. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–137. IEEE, 2003.

[FBK15] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. Optical flow modeling and computation: a survey. *Computer Vision and Image Understanding*, 134:1–21, 2015.

[FC07] Sándor Fazekas and Dmitry Chetverikov. Analysis and performance evaluation of optical flow features for dynamic texture recognition. *Signal Processing: Image Communication*, 22(7):680–691, 2007.

[FF62] LR Ford and Delbert Ray Fulkerson. *Flows in networks*, volume 1962. Princeton Princeton University Press, 1962.

[FH06] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient belief propagation for early vision. *International journal of computer vision*, 70(1):41–54, 2006.

[FLPM14] Francisco Gómez Fernández, Zicheng Liu, Alvaro Pardo, and Marta Mejail. Automatic camera-screen localization. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications - 19th Iberoamerican Congress, CIARP 2014, Puerto Vallarta, Mexico, November 2-5, 2014. Proceedings*, pages 588–595, 2014.

[GY06] Minglun Gong and Yee-Hong Yang. Disparity flow estimation using orthogonal reliability-based dynamic programming. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 70–73. IEEE, 2006.

[HA85] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[HB11] Simon Hadfield and Richard Bowden. Kinecting the dots: Particle based scene flow from depth sensors. In *Computer Vision*

(ICCV), 2011 IEEE International Conference on, pages 2290–2295. IEEE, 2011.

[HB14] Simon Hadfield and Richard Bowden. Scene particles: Unregularized particle-based scene flow estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(3):564–576, 2014.

[HBK+14] Michael Hornáček, Frederic Besse, Jan Kautz, Andrew Fitzgibbon, and Carsten Rother. Highly overparameterized optical flow using patchmatch belief propagation. In *Computer Vision–ECCV 2014*, pages 220–234. Springer, 2014.

[HD07] Frédéric Huguet and Frédéric Devernay. A variational method for scene flow estimation from stereo sequences. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–7. IEEE, 2007.

[HFR14] Michael Hornacek, Andrew Fitzgibbon, and Carsten Rother. Sphereflow: 6 dof scene flow from rgb-d pairs. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3526–3533. IEEE, 2014.

[HRF13] Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d flow: Dense 3-d motion estimation using color and depth. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2276–2282. IEEE, 2013.

[HS81] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, August 1981.

[HWHC11] Min-Yu Huang, Shih-Chieh Wei, Bormin Huang, and Yang-Lang Chang. Accelerating the kalman filter on a gpu. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pages 1016–1020. IEEE, 2011.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[IK08] Serdar Ince and Janusz Konrad. Occlusion-aware optical flow estimation. *Image Processing, IEEE Transactions on*, 17(8):1443–1451, 2008.

[Ish03] Hiroshi Ishikawa. Exact optimization for markov random fields with convex priors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1333–1336, 2003.

[Ist03] Robert SH Istepanian. Microarray processing: current status and future directions. *IEEE transactions on Nanobioscience*, 2(4):173–175, 2003.

[JB06] Olivier Juan and Yuri Boykov. Active graph cuts. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 1023–1029. IEEE, 2006.

[JH99] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[Jor98] Michael Irwin Jordan. *Learning in graphical models*, volume 89. Springer Science & Business Media, 1998.

[KH02] Changick Kim and Jenq-Neng Hwang. Fast and automatic video object segmentation and tracking for content-based applications. *Circuits and Systems for Video Technology, IEEE Transactions on*, 12(2):122–129, 2002.

[KT05] Pushmeet Kohli and Philip HS Torr. Efficiently solving dynamic markov random fields using graph cuts. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 922–929. IEEE, 2005.

[KZ04] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? *IEEE transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, February 2004.

[LCKW94] M.R. Luettgen, W. Clem Karl, and A.S. Willsky. Efficient multiscale regularization with applications to the computation of optical flow. *Image Processing, IEEE Transactions on*, 3(1):41–64, Jan 1994.

[LK81] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 674–679, 1981.

[Low99] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LS13] Li Liu and Ling Shao. Learning discriminative representations from rgb-d video data. In *IJCAI*, 2013.

[LTN+11] Hatem Ltaief, Stanimire Tomov, Rajib Nath, Peng Du, and Jack Dongarra. A scalable high performant cholesky factorization for multicore with gpu accelerators. In *High Performance Computing for Computational Science–VECPAR 2010*, pages 93–101. Springer, 2011.

[LYT11] C. Liu, J. Yuen, and A. Torralba. SIFT flow: Dense correspondence across scenes and its applications. *IEEE Trans Pattern Anal Mach Intell*, 33(5):978–994, May 2011.

[MCLC13] Adeel Mumtaz, Emanuele Coviello, Gert RG Lanckriet, and Antoni B Chan. Clustering dynamic textures with the hierarchical em algorithm for modeling video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(7):1606–1621, 2013.

[MCT09] Erik Murphy-Chutorian and Mohan M Trivedi. Head pose estimation in computer vision: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):607–626, 2009.

[MSKS12] Yi Ma, Stefano Soatto, Jana Kosecka, and S Shankar Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer Science & Business Media, 2012.

[MYN07] Jonathan Mooser, Suya You, and Ulrich Neumann. Real-time object tracking for augmented reality combining graph cuts and optical flow. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–8. IEEE Computer Society, 2007.

[NP92] Randal C Nelson and Ramprasad Polana. Qualitative recognition of motion using temporal texture. *CVGIP: Image understanding*, 56(1):78–89, 1992.

[NVI12] NVIDIA. Cuda cublas library, January 2012. Version 4.1.

[ON94] Michael Otte and H-H Nagel. Optical flow estimation: advances and comparisons. In *Computer Vision—ECCV'94*, pages 49–60. Springer, 1994.

[PN97] Ramprasad Polana and Randal Nelson. *Temporal texture and activity recognition*. Springer, 1997.

[QBDC14] Julian Quiroga, Thomas Brox, Frédéric Devernay, and James Crowley. Dense semi-rigid scene flow estimation from rgbd images. In *Computer Vision–ECCV 2014*, pages 567–582. Springer, 2014.

[QDC13] Julian Quiroga, Frédéric Devernay, and James Crowley. Local/-global scene flow estimation. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 3850–3854. IEEE, 2013.

[RC98] Sébastien Roy and Ingemar J Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *Computer Vision, 1998. Sixth International Conference on*, pages 492–499. IEEE, 1998.

[RCV13] Arunkumar Ravichandran, Rizwan Chaudhry, and Rene Vidal. Categorizing dynamic textures using a bag of dynamical systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(2):342–353, 2013.

[RG99] Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural computation*, 11(2):305–345, 1999.

[RMWF10] Clemens Rabe, Thomas Müller, Andreas Wedel, and Uwe Franke. Dense, robust, and accurate motion field estimation from stereo image sequences in real-time. In *Computer Vision–ECCV 2010*, pages 582–595. Springer, 2010.

[SAL07] Nurul Arif Setiawan, Dhi Aurrahman, and Chil Woo Lee. Optical flow in dynamic graph cuts. In *Communications, Computers and Signal Processing, 2007. PacRim 2007. IEEE Pacific Rim Conference on*, pages 288–291. IEEE, 2007.

[SBK10] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *Computer Vision–ECCV 2010*, pages 438–451. Springer, 2010.

[SBM+11] Patrik Sundberg, Thomas Brox, Michael Maire, Pablo Arbeláez, and Jitendra Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2233–2240. IEEE, 2011.

[SDW01] Stefano Soatto, Gianfranco Doretto, and Ying Nian Wu. Dynamic textures. In *Computer Vision, 2001. ICCV 2001. Proceed-*

ings. Eighth IEEE International Conference on, volume 2, pages 439–446. IEEE, 2001.

[SFVG04] Christoph Strecha, Rik Fransens, and Luc Van Gool. A probabilistic approach to large displacement optical flow and occlusion detection. In *Statistical methods in video processing*, pages 71–82. Springer, 2004.

[SJB00] Hagen Spies, Bemd Jähne, and John L Barron. Dense range flow from depth and intensity data. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 1, pages 131–134. IEEE, 2000.

[SKH08] Alexander Shekhovtsov, Ivan Kovtun, and Václav Hlaváč. Efficient mrf deformation model for non-rigid image matching. *Computer Vision and Image Understanding*, 112(1):91–99, 2008.

[SS02] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[SS03] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–195. IEEE, 2003.

[STC09] Frank R Schmidt, Eno Toppe, and Daniel Cremers. Efficient planar graph cuts with applications in computer vision. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 351–356. IEEE, 2009.

[SZS+08] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(6):1068–1080, 2008.

[TH98] Miroslav Trajković and Mark Hedley. Fast corner detection. *Image and Vision Computing*, 16(2):75–87, 1998.

[TS12] Mariano Tepper and Guillermo Sapiro. Decoupled coarse-to-fine matching and nonlinear regularization for efficient motion estimation. In *ICIP*, pages 1517–1520, 2012.

[TSDS10] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European Conf. on Computer Vision (ECCV)*, pages 356–369. Springer, 2010.

[Vaz13] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[VBR+99] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 722–729. IEEE, 1999.

[WBV+11] Andreas Wedel, Thomas Brox, Tobi Vaudrey, Clemens Rabe, Uwe Franke, and Daniel Cremers. Stereoscopic scene flow computation for 3d motion understanding. *International Journal of Computer Vision*, 95(1):29–51, 2011.

[WM95] Joseph Weber and Jitendra Malik. Robust computation of optical flow in a multi-scale differential framework. *International Journal of Computer Vision*, 14(1):67–81, 1995.

[WRV+08] Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke, and Daniel Cremers. Efficient dense scene flow from sparse or dense stereo data. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 739–751, Berlin, Heidelberg, 2008. Springer-Verlag.

[XJM12] Li Xu, Jiaya Jia, and Yasuyuki Matsushita. Motion detail preserving optical flow estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1744–1757, 2012.

[XS05] Jiangjian Xiao and Mubarak Shah. Motion layer extraction in the presence of occlusion using graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1644–1659, 2005.

[YWLS04] Lu Yuan, Fang Wen, Ce Liu, and Heung-Yeung Shum. Synthesizing dynamic texture with closed-loop linear dynamic system. In *Computer Vision-ECCV 2004*, pages 603–616. Springer, 2004.

[ZB12] Jieyu Zhang and John L. Barron. Optical Flow at Occlusion. In *Canadian Conference on Computer and Robot Vision*, 2012.

[Zha94] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.