VISIÓN POR COMPUTADORA

MOVIMIENTO, ESTRUCTURA Y DETECCIÓN

JOAQUÍN SALAS

INSTITUTO POLITÉCNICO NACIONAL
CICATA QUERÉTARO

Visión por Computadora: Movimiento, Estructura y Detección

Joaquín Salas
CICATA Querétaro
Instituto Politécnico Nacional
Cerro Blanco 141, Colinas del Cimatario, 76090,
Querétaro, México
jsalasr@ipn.mx

Marzo, 2016

Visión por Computadora: Movimiento, Estructura y Detección
Joaquín Salas

Primera edición:2016

Instituto Politécnico Nacional
Luis Enrique Erro s/n
Unidad Profesional "Adolfo López Mateos"
Zacatenco, Deleg. Gustavo A. Madero
CP 07738, Ciudad de México

Instituto Politécnico Nacional
CICATA Querétaro
Cerro Blanco 141
Colinas del Cimatario
CP 76090, Querétaro, México

ISBN (impreso) 978-607-414-521-2
ISBN (electrónico) 978-607-414-525-0
http://www.cicataqro.ipn.mx

Diseño de Portada: Alma Lucero Flores Ramírez
Revisión: Ramón Martínez de Velasco

Dedicado a
María Elena, Daniel, Linda y Samuel,
por su compañía en el camino.

Índice general

Prefacio

Este documento surge de notas del curso de Visión por Computadora que se imparte en el Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada Querétaro del Instituto Politécnico Nacional. Para desarrollar este material, he ido a diversas fuentes primarias, tratado de seguir su hilo discursivo, y complementado el material con otras referencias, ejercicios, e ilustraciones que he desarrollado. He tratado, y espero haberlo logrado, de dar la correcta cita a los autores de los trabajos, más cuando en la línea principal de desarrollo he utilizado mayormente notación similar a la que ellos han propuesto en sus trabajos. Mi idea ha sido explicar el material de la manera más sencilla posible, haciendo la cita apropiada para que el estudiante pueda ir a la referencia primaria del trabajo.

El material ha sido desarrollado con varios objetivos en mente. Por un lado, pretende dar una presentación balanceada entre aquellos trabajos que han resistido la prueba del tiempo y aquellos nuevos desarrollos que ofrecen perspectivas prometedoras. En la exposición de los artículos, el material revisado es, por un lado destilado en su esencia, y por otro lado detallado, tratando de lograr una comprensión que permita su uso y extensión. Otra razón importante para escribir este documento es construir una plataforma electrónica que vaya evolucionando hacia exposiciones más interactivas, no lineales, donde haya una combinación de fundamentos y actividades prácticas.

El documento se desarrolla de la siguiente manera. En la primera parte se hace una revisión de algunas técnicas de Análisis de Movimiento. Se comienza con los fundamentos establecidos por Lucas-Kanade[11] y los complementos hechos por Shi-Tomasi [14]. Esos trabajos establecen algunos criterios para obtener propiedades que naturalmente ocurren en las imágenes, particularmente útiles para realizar seguimiento de características. Enseguida, estudiamos el algoritmo de Horn-Schunck[8], donde, con el propósito de volver tratable el problema, se introduce el uso de restricciones de variación suave de los campos de movimiento, de términos de regularización, y de soluciones iterativas. Esto se complementa con las ideas de Farnebäck[4], actualmente implementadas en la librería de OpenCV[1].

En la segunda parte estudiamos el tema de la Reconstrucción Tridimensional a partir de las imágenes. Primero comenzamos la exposición de la idea de establecer la geometría epipolar mediante el algoritmo de los ocho puntos[7], el cual requiere la aplicación de normalización para obtener soluciones robustas. Una alternativa a este algoritmo es la factorización introducida por Tomasi-Kanade[16]. Si bien el primero se aplica en situaciones más generales donde ocurre proyección perspectiva del mundo a la imagen, la segunda, basada en la suposición de proyección ortográfica, ofrece soluciones robustas para una variedad importante de casos prácticos. En ambos casos, los objetos estudiados son rígidos. Gotardo-Martinez [6] propusieron recientemente un esquema de factorización donde se emplea la suposición de que la reconstrucción resulta de la combinación lineal de formas básicas y trayectorias suaves. Esta parte termina con el estudio de técnicas que han estado cobrando fuerza con la introducción de cámaras de profundidad baratas, confiables y rápidas. Algunos de los métodos

resultantes permiten la reconstrucción tridimensional de escenarios de escalas medianas[12], y la detección de personas en tiempo real[15].

En la tercera y última parte, estudiamos la detección de objetos. Las técnicas más sobresalientes parecen utilizar descriptores basados en el uso de información del gradiente. En ese sentido, Lowe[10] propuso las llamadas características SIFT, las cuales describen localmente propiedades de la imagen que son invariantes a rotación, cambio de escala, y parcialmente invariantes a transformaciones afines y cambios de iluminación. La búsqueda de detectores eficaces y eficientes tiene su materialización en el propuesto por Viola-Jones[17]. Su esquema para la obtención de descriptores sencillos, su aplicación en cascadas de detectores débiles y la propuesta de cálculo de descriptores mediante la imagen integral, ha estado evolucionando. Un ejemplo de estos últimos resultados son los esquemas multicanales de Dóllar[3]. Otro resultado sobresaliente en la detección de objetos lo ofrece el uso de histogramas de gradiente orientados, inicialmente propuesto por Dalal-Triggs[2]. Su uso en la descripción de partes de objetos propuesto por Felzenszwalb *et al.*[5] ha permitido su aplicación en situaciones donde hay oclusiones. Más recientemente, mediante el uso de técnicas basadas en redes neuronales convolucionales (CNN) se han obtenido resultados sobresalientes. Introducidas por Yann Lecun[9] en el contexto del reconocimiento de dígitos escritos manualmente, las CNN se utilizan hoy con bastante éxito para la detección de objetos en lo general[13]. Aquí estudiamos los procesos de cálculo de la función de discriminación que representan, su entrenamiento, la definición de arquitecturas básicas y algunos aspectos importantes para su uso antes y durante su entrenamiento.

Mi agradecimiento a quienes han participado, a través de sus comentarios, en el enriquecimiento de este trabajo. Igualmente mi agradecimiento a los autores de los trabajos que aquí se mencionan, por su obra y por permitirnos a través de ella lograr un mejor entendimiento del fenómeno de la obtención de información a partir de imágenes.

Bibliografía

[1] Bradski, G.: *The OpenCV Library*. Dr. Dobb's Journal of Software Tools, 2000.

[2] Dalal, Navneet y Bill Triggs: *Histograms of Oriented Gradients for Human Detection*. En *IEEE Conference on Computer Vision and Pattern Recognition*, volumen 1, páginas 886–893, 2005.

[3] Dollár, Piotr, Ron Appel, Serge Belongie y Pietro Perona: *Fast Feature Pyramids for Object Detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014.

[4] Farnebäck, Gunnar: *Two-Frame Motion Estimation based on Polynomial Expansion*. En *Image Analysis*, páginas 363–370. Springer, 2003.

[5] Felzenszwalb, Pedro, Ross Girshick, David McAllester y Deva Ramanan: *Object Detection with Discriminatively Trained Part-based Models*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645, 2010.

[6] Gotardo, Paulo y Aleix Martinez: *Non-Rigid Structure from Motion with Complementary Rank-3 Spaces*. En *IEEE Conference on Computer Vision and Pattern Recognition*, páginas 3065–3072, 2011.

[7] Hartley, Richard: *In Defense of the Eight-Point Algorithm*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(6):580–593, 1997.

[8] Horn, Berthold y Brian Schunck: *Determining Optical Flow*. En *Technical Symposium East*, páginas 319–331. International Society for Optics and Photonics, 1981.

[9] LeCun, Yann, Léon Bottou, Yoshua Bengio y Patrick Haffner: *Gradient-based Learning Applied to Document Recognition*. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[10] Lowe, David: *Distinctive Image Features from Scale-Invariant Keypoints*. International Journal of Computer Vision, 60(2):91–110, 2004.

[11] Lucas, Bruce y Takeo Kanade: *An Iterative Image Registration Technique with an Application to Stereo Vision*. En *International Joint Conferences on Artificial Intelligence*, volumen 81, páginas 674–679, 1981.

[12] Newcombe, Richard, Andrew Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim y Andrew Fitzgibbon: *KinectFusion: Real-Time Dense Surface Mapping and Tracking*. En *IEEE International Symposium on Mixed and Augmented Reality*, páginas 127–136, 2011.

[13] Ouyang, Wanli, Xiaogang Wang, Cong Zhang y Xiaokang Yang: *Factors in Finetuning Deep Model for Object Detection with Long-Tail Distribution*. En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 864–873, 2016.

[14] Shi, Jianbo y Carlo Tomasi: *Good Features to Track*. En *IEEE Conference on Computer Vision and Pattern Recognition*, páginas 593–600, 1994.

[15] Shotton, Jamie, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook y Richard Moore: *Real-time Human Pose Recognition in Parts from Single Depth Images*. Communications of the ACM, 56(1):116–124, 2013.

[16] Tomasi, Carlo y Takeo Kanade: *Shape and Motion from Image Streams under Orthography: A Factorization Method*. International Journal of Computer Vision, 9(2):137–154, 1992.

[17] Viola, Paul y Michael Jones: *Rapid Object Detection using a Boosted Cascade of Simple Features*. En *IEEE Conference on Computer Vision and Pattern Recognition*, volumen 1, páginas I–511, 2001.

Parte I

Análisis de Movimiento

CAPÍTULO 1

Seguimiento de Características

Un problema importante en Visión por Computadora es determinar el desplazamiento de pequeñas porciones de una imagen, la cual puede ser parte de un video o un elemento en una secuencia de imágenes. Una algoritmo popular para resolver este problema es el propuesto por Lucas y Kanade[6], el cual inicialmente surgió como una parte central para la búsqueda de la correspondencia en un par estéreo. Este algoritmo fue posteriormente analizado por Shi y Tomasi[11], quienes obtuvieron algunas conclusiones importantes sobre él. En particular, su análisis puso énfasis en el comportamiento de los valores singulares y la inestabilidad numérica que surgía al momento de realizar la búsqueda de las correspondencias cuando se tomaba un modelo que incluía traslaciones y transformaciones afines, *i.e.*, que incluye rotaciones, cambios de escala y sesgos. De entonces, mucho se ha estudiado sobre las propiedades de este algoritmo[2]. Hoy en día la idea sigue vigente, y muestra de ello lo constituye la abundante y reciente literatura sobre el tema[8].

Así pues, partiendo de la intuición proporcionada por Shi y Tomasi[11], en el sentido de que las formulaciones más elaboradas que toman en cuenta transformaciones afines parecen ser numéricamente inestables, en este documento derivamos el seguidor para el caso sencillo de una traslación. Luego, revisamos la matriz de la estructura, esencial en la formulación de Lucas y Kanade, la cual brinda información sobre cuáles características son más robustas para ser seguidas, *i.e.*, cuya distribución de niveles de intensidad facilita su localización en la siguiente imagen. Enseguida, mostramos un ejemplo de la implementación del seguidor utilizando algunos recursos disponibles en Matlab y OpenCV. Luego, buscamos desarrollar un poco de intuición respecto al proceso del seguidor para la búsqueda de los parámetros y su velocidad de convergencia. Marcadamente notamos la similaridad del seguidor con la formulación de Newton-Raphson[9] para la búsqueda de raíces. Finalmente, visitamos el problema de la construcción de trayectorias sobre todos los cuadros de una secuencia de video, detallando algunas complicaciones tales como la asignación de una trayectoria por cada punto del mundo.

1.1. Derivación

La derivación que aquí presentamos se basa en el reporte técnico de Shi y Tomasi[11]. Normalmente, la representación de una imagen en una computadora es discreta, y se da a través de una matriz. En contraposición a la presentación de Shi y Tomasi[11], quienes presentaron su análisis considerando a las imágenes como funciones continuas, aquí hacemos

la sustitución de integrales por sumatorias y derivadas por diferencias. En adición, asumimos que los valores de las imágenes para coordenadas no enteras se obtienen por interpolación. Así, dadas dos imágenes $I : \mathcal{R}^2 \to \mathcal{R}$ y $J : \mathcal{R}^2 \to \mathcal{R}$ una medida de disimilaridad entre pequeñas porciones de ellas, $\mathbf{W}$, podría estar dada por la suma de las diferencias al cuadrado (*SSD*). Es decir,

$$\epsilon_0(\mathbf{d}) = \sum_{\mathbf{x}\in\mathbf{W}} \left[J(\mathbf{x}+\mathbf{d}) - I(\mathbf{x})\right]^2, \tag{1.1}$$

donde $\mathbf{x} = [x, y]^T$ corresponde a una posición en la imagen, y $\mathbf{d} = [d_x, d_y]^T$ es el desplazamiento que la característica en la ventana $\mathbf{W}$ tiene entre la imagen I y la imagen J.

Note que el término $\mathbf{d}$ no puede ser resuelto inmediatamente, pues se encuentra como argumento de la función. Una forma de resolver este problema es linealizando la función $J(\mathbf{x}+\mathbf{d})$ utilizando su expansión en términos de la serie de Taylor[1], tal que

$$J(\mathbf{x}+\mathbf{d}) = J(\mathbf{x}) + \mathbf{g}^T(\mathbf{x})\mathbf{d} + \mathcal{O}(\mathbf{x}), \tag{1.3}$$

donde $\mathbf{g}(\mathbf{x}) = \nabla_{\mathbf{x}} J(\mathbf{x})$, y $\mathcal{O}(\mathbf{x})$ aglutina aquellos elementos de orden cuadrático o superior, los cuales se asumen prescindibles. Por su lado, la derivada de (1.3) está dada por

$$\nabla_{\mathbf{d}} = \mathbf{g}(\mathbf{x}) + \nabla_{\mathbf{d}}\mathcal{O}(\mathbf{x}). \tag{1.4}$$

Volviendo a nuestro objetivo de obtener el desplazamiento óptimo $\mathbf{d}$, vemos que este puede ser encontrado derivando (1.1) con respecto a $\mathbf{d}$ y resolviendo la ecuación resultante con respecto a cero, lo cual resulta en

$$\frac{\partial \epsilon}{\partial \mathbf{d}} = 2 \sum_{\mathbf{x}\in\mathbf{W}} \left[J(\mathbf{x}+\mathbf{d}) - I(\mathbf{x})\right] \nabla_{\mathbf{d}} J(\mathbf{x}+\mathbf{d}). \tag{1.5}$$

Enseguida, sustituyendo la expansión de Taylor (1.3) y su correspondiente derivada con respecto a $\mathbf{d}$ en (1.4), tenemos que una aproximación a $\dfrac{\partial \epsilon}{\partial \mathbf{d}}$ está dada por

$$\frac{\partial \epsilon}{\partial \mathbf{d}} \approx \sum_{\mathbf{x}\in\mathbf{W}} \left[J(\mathbf{x}) + \mathbf{g}(\mathbf{x})^T\mathbf{d} - I(\mathbf{x})\right] \mathbf{g}(\mathbf{x}) = 0. \tag{1.6}$$

Reordenando los términos de la ecuación anterior, con los términos en $\mathbf{d}$ del lado derecho y sin ella del lado izquierdo de la igualdad, y sacando $\mathbf{d}$ de la integral, dado que no depende de $\mathbf{x}$, tenemos que

$$\begin{aligned} \sum_{\mathbf{x}\in\mathbf{W}} \left[J(\mathbf{x}) - I(\mathbf{x})\right] \mathbf{g}(\mathbf{x}) &= -\sum_{\mathbf{x}\in\mathbf{W}} \mathbf{g}(\mathbf{x})(\mathbf{g}(\mathbf{x})^T\mathbf{d}), \\ &= -\left[\sum_{\mathbf{x}\in\mathbf{W}} \mathbf{g}(\mathbf{x})\mathbf{g}(\mathbf{x})^T\right]\mathbf{d}. \end{aligned} \tag{1.7}$$

[1]La expansión en serie de Taylor de una función $f(x_0+h)$ es igual a

$$f(x_0+h) = \sum_{n=0}^{\infty} f^{(n)}(x_0)h^n/n! = f(x_0) + f'(x_0)h + f''(x_0)h^2/2! + \cdots + f^{(n)}(x_0)h^n/n! + \ldots \tag{1.2}$$

En forma matricial, la expresión anterior puede reescribirse como

$$\mathbf{e} = -\mathbf{Z}\mathbf{d}, \text{ ó } \mathbf{d} = -\mathbf{Z}^{-1}\mathbf{e}, \tag{1.8}$$

donde $\mathbf{Z}$ y $\mathbf{e}$ están dados por

$$\mathbf{Z} = \sum_{\mathbf{x}\in\mathbf{W}} \mathbf{g}(\mathbf{x})\mathbf{g}(\mathbf{x})^T, \tag{1.9}$$

y

$$\mathbf{e} = \sum_{\mathbf{x}\in\mathbf{W}} [J(\mathbf{x}) - I(\mathbf{x})]\, \mathbf{g}(\mathbf{x}). \tag{1.10}$$

Esto permite encontrar el desplazamiento óptimo dentro de la ventana $\mathbf{W}$. Una vez en ese punto, la estructura local de los niveles de intensidad puede hacer posible que la función de disimilaridad se reduzca aún más. Esto da pie a la definición de una formulación de búsqueda iterativa que tendría la forma

$$\mathbf{d}_{k+1} = \mathbf{d}_k - \mathbf{Z}^{-1}\mathbf{e}. \tag{1.11}$$

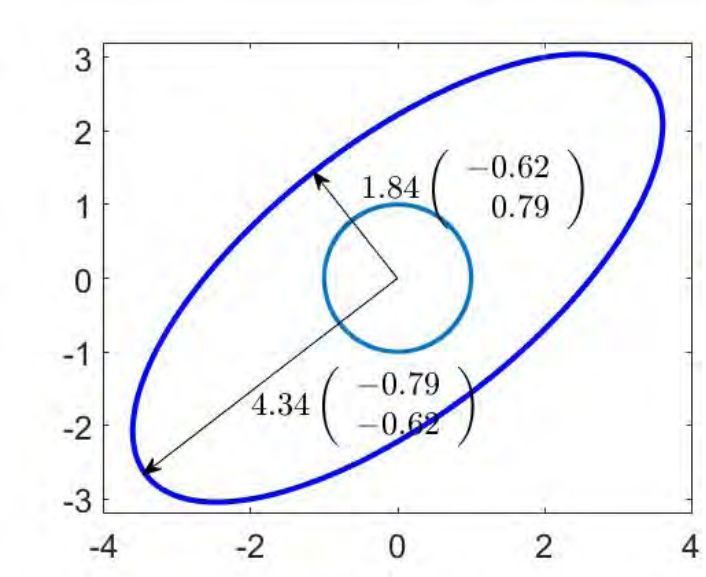


Figura 1.1: Una transformación lineal $\mathbf{A}$ modifica los puntos de un círculo unitario en una elipse cuyos ejes están caracterizados por los eigenvalores y eigenvectores de $\mathbf{A}$. Los ejes de la elipse se orientan en la dirección de los eigenvectores y la magnitud de los ejes principales coincide con los valores singulares. Los valores numéricos se encuentran en (1.12).

1.2. Matriz de Estructura

La matriz $\mathbf{Z}$ podría ser vista como una matriz de covarianza, donde el promedio del valor del gradiente es cero. En ocasiones está matriz es llamada *tensor estructural* o *matriz de segundos momentos*. Algunas de las propiedades interesantes de esta matriz incluyen que es simétrica y positiva semi-definida[12]. Esto significa, entre otras cosas, que la matriz es

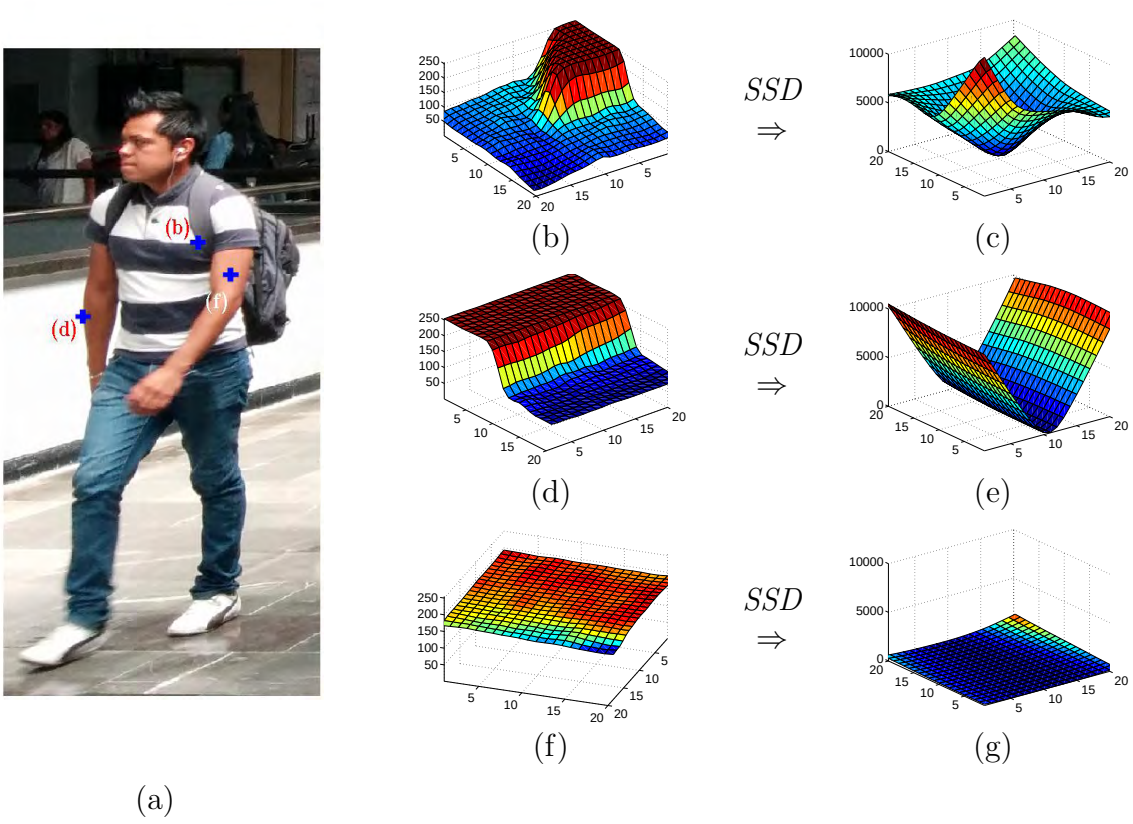
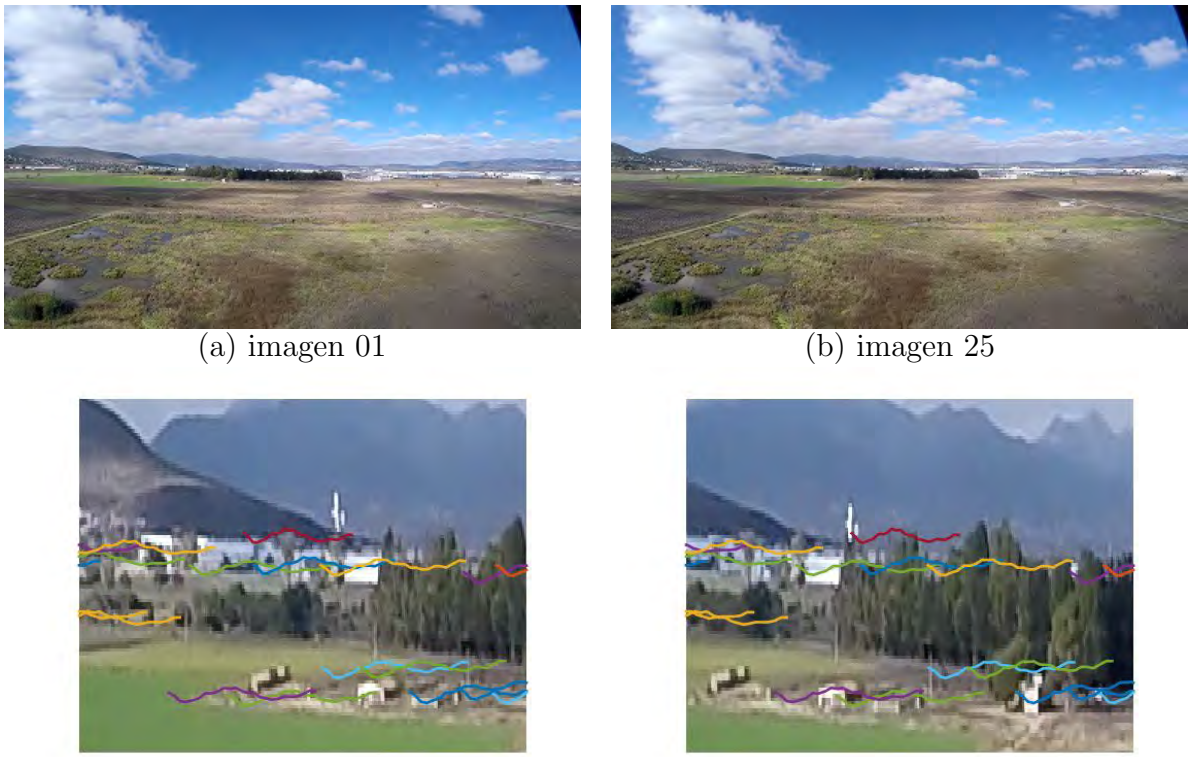


Figura 1.2: Estructura de las Características. La posibilidad de identificar una característica a lo largo de una secuencia tiene mucho que ver con su discriminabilidad, a excepción de oclusiones, por supuesto. Así, una esquina tendrá más posibilidades de ser distinguida entre cuadros, la posición de un punto sobre una línea tendrá ambigüedad sobre ella, y un punto en una región plana será muy poco distinguible. Esto es conocido como el *problema de la apertura*. En (b), (d) y (f) se muestran los perfiles de intensidad de algunas regiones de la imagen (a). Luego en (c), (e) y (g) se muestran las correspondientes evaluaciones de (1.1) para esas regiones. Una esquina presenta una SSD en forma de cuenca, un contorno una `V`, y una región homogénea una planicie.

invertible y sus eigenvalores son positivos. En general, una matriz transforma los puntos en el círculo unitario en una elipse, cuyos ejes principales son caracterizados por su eigensistema. Así, las direcciones de los ejes principales de la elipse y los eigenvectores coinciden, y la magnitud de los ejes está dada por los valores singulares[4]. Por ejemplo, considere la matriz $\mathbf{Z} = \begin{pmatrix} 2 & 3 \\ 3 & 0.5 \end{pmatrix}$, cuya descomposición en valores singulares está dada por

$$\begin{aligned} \mathbf{Z} &= \mathbf{USV}^T, \\ \begin{pmatrix} 2 & 3 \\ 3 & 0.5 \end{pmatrix} &= \begin{pmatrix} -0.79 & -0.62 \\ -0.62 & 0.79 \end{pmatrix} \begin{pmatrix} 4.34 & 0 \\ 0 & 1.84 \end{pmatrix} \begin{pmatrix} -0.79 & -0.62 \\ 0.62 & -0.79 \end{pmatrix}. \end{aligned} \tag{1.12}$$

La Figura 1.1 ilustra los valores numéricos de este ejemplo.



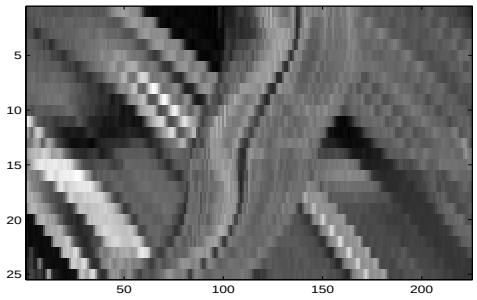
(a) imagen 01 (b) imagen 25

(c) (d)

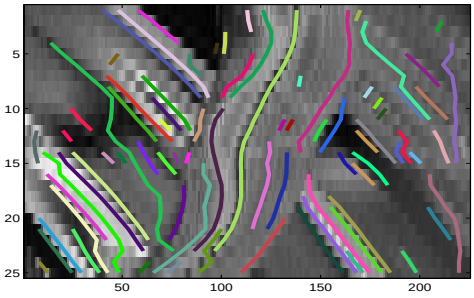
Figura 1.3: Implementación del seguidor de Lucas-Kanade en OpenCV. Algunos puntos de la imagen son considerados como susceptibles de ser seguidos. Luego, entre un cuadro y el siguiente se busca su correspondencia. Aquí se ilustra el resultado usando el seguidor de Lucas-Kanade (ver el Algoritmo 1). En la secuencia, una cámara rota de izquierda (a) a derecha (b). Un detalle de la imagen es amplificado ilustrando la trayectoria seguida por algunas características (mejor vista a color).

La matriz $\mathbf{Z}$ puede decirnos mucho sobre la distribución de los niveles de intensidad en una ventana $\mathbf{W}$. Por ejemplo, considere la imagen en la Figura 1.2, y las ventanas asociadas a diversas regiones de ella. Cuando el punto que deseamos seguir se encuentra en una esquina, la alta discriminabilidad de la región de la imagen donde el punto se encuentra facilita su localización. Por otro lado, si el punto se encuentra sobre una línea, es imposible resolver la ambigüedad de la correspondencia a lo largo de ella. Un caso más complicado se presenta cuando no hay referencia local sobre la cual identificar al punto. En este último caso, no es posible resolver el problema de la correspondencia más que sobre los puntos de la región. Esto es conocido como el *problema de la apertura*. En la Figura 1.2(a), (b) y (c), los valores singulares (σ_1, σ_2) son respectivamente $(490.1, 352.6)$, $(520.2, 5.0)$, y $(1.7, 0.7)$. Es decir, cualitativamente en una esquina ambos valores singulares son grandes; en una línea, uno de los valores singulares es grande y el otro pequeño; finalmente, en una región plana, ambos valores singulares son pequeños.

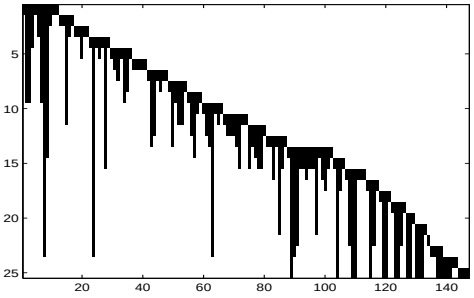
Varios esquemas han sido propuestos para determinar la bondad de una característica



(a)



(b)



(c)

Figura 1.4: Seguimiento de características a lo largo de la secuencia. En (a) se muestra la evolución de una imagen, que sólo tiene una hilera, a través del tiempo. En el desarrollo de la secuencia se aprecian oclusiones y acreciones ocasionadas por el objeto un objeto que aparece en la parte central. En la secuencia, detectamos las características buenas en cada imagen y las seguimos (b). Cuando las nuevas características coincidían con los seguimientos, se preferían estos últimos. El proceso continúa así hasta el fin de la secuencia. En (c) se ilustra la matriz de llenado $\mathbf{F}$, que indica el número de la característica (columna) y el número de imagen en la que la característica fue vista (hilera). Un problema es que algunas de las características hacen referencia al mismo punto en el mundo debido a oclusiones intermedias o fallas en el seguidor.

para ser seguida. Por ejemplo, Harris y Stephens[5] propusieron usar la operación[2]

$$c = \sigma_1\sigma_2 - \kappa(\sigma_1 + \sigma_2)^2 = \texttt{det}(\mathbf{Z}) - \kappa\texttt{trace}^2(\mathbf{Z}), \tag{1.13}$$

donde κ es un parámetro ajustable, `det` corresponde al determinante[3], y `trace` corresponde a la traza[4]. Por otro lado, Shi y Tomasi[11] usan un criterio, donde evalúan los valores singulares, dado por

$$\text{mín}(\sigma_1, \sigma_2) \leq \tau, \tag{1.14}$$

[2]Recordar que para matrices simétricas (en general Hermitianas) los valores singulares y los eigenvalores son iguales en virtud de que la descomposición de similaridad $\mathbf{A} = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^{-1} = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^{T}$ resulta ser una descomposición en valores singulares válida[7].

[3]El determinante de una matriz de $\mathbf{Z}_{2\times 2} = \begin{pmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{pmatrix}$ es igual a $z_{11}z_{22} - z_{21}z_{12}$

[4]La traza es igual a la suma de los elementos de la diagonal $\sum_{i=1}^{n} z_{ii}$

para un cierto valor de umbral τ.

1.3. Búsqueda del Óptimo

La Ec. (1.11) guarda cierta similaridad con el método de Newton-Raphson[9], el cual se utiliza para la localización de cruces por cero de una función. La matriz $\mathbf{Z}$ sumariza las direcciones del gradiente en una cierta vecindad. Si el gradiente se orienta predominantemente en una dirección, el valor singular correspondiente será grande. Alternativamente, en la dirección más homogenea, el valor singular correspondiente será pequeño. Sin embargo, (1.11) utiliza la inversa $\mathbf{Z}^{-1}$, lo cual implica que las direcciones de búsqueda se amplifican inversamente a los valores singulares de $\mathbf{Z}$. Es decir, entre más pequeño el valor singular de $\mathbf{Z}$, el correspondiente desplazamiento en la dirección de su eigenvector respectivo es mayor. Esto sigue guardando similitud con el método de Newton-Raphson, donde una posición con derivada cercana a cero puede hacer que el valor de la innovación sea muy grande. Por otro lado, el término $\mathbf{e}_k$ corresponde a un vector en la orientación del gradiente que es amplificado en función de la diferencia entre las imágenes $\mathbf{I}$ y $\mathbf{J}$. La dirección y magnitud del cambio de $\mathbf{d}_k$ está dado en función de la transformación del vector $\mathbf{e}_k$ y la transformación $\mathbf{Z}^{-1}$.

Similar al método de Newton-Raphson[5], el cual converge a la solución en forma cuadrática cerca de la región donde se encuentra la solución[9], el seguidor de Lucas-Kanade es bastante efectivo. Sin embargo, al igual que su contraparte, la configuración local del espacio de búsqueda puede hacer imposible su convergencia. En [1], Anandan analiza las superficies de búsqueda que se generan en esquinas, líneas, y regiones homogéneas. Para ello, varió $\mathbf{d}$ en (1.1) para esos tres casos (ver Figura 1.2). Así, es posible observar que, para las esquinas, la región de búsqueda presenta una región de convergencia donde la estructura del mínimo es muy notoria desde cualquier dirección de inicio. Por otro lado, en el caso de una línea, la presencia de un mínimo está muy marcada solamente en una dirección, mientras que en la otra dirección el gradiente es prácticamente cero. En esas condiciones, lo más probable es que el seguidor terminará perdiéndose por la presencia de una derivada muy cercana a cero. Finalmente, en las regiones homogéneas no se tendrá confiabilidad en la localización de la correspondencia pues no hay referencias, traduciéndose en un gradiente que es prácticamente cero en cualquier dirección.

1.4. Ilustración del Seguimiento de Características

Recientemente, Kota Yamaguchi desarrolló *mexopencv*[13], una interfase que permite la implementación de rutinas de OpenCV[3] en Matlab. El Algoritmo 1 muestra un ejemplo de pseudocódigo donde se observan las funciones a llamar para obtener la posición de las

[5]El método de Newton-Raphson se utiliza para localizar las raíces o cruces por cero de una función. El método se basa en el esquema iterativo

$$x_{n+1} \leftarrow x_n - f(x_n)/f'(x_n),$$

donde x_n y x_{n+1} corresponden a los valores actual y siguiente de x, y $f(x)$ y $f'(x)$ corresponden a la función y su derivada.

Llamada : $< \mathbf{M}, \mathbf{F} > \leftarrow$ `Seguir_Características` $(\mathcal{I})$
Entradas: Una secuencia de imágenes $\mathcal{I} = \mathbf{I}_1, \ldots, \mathbf{I}_m$
Salidas : La matriz de mediciones $\mathbf{M}_{2(m-1)\times\kappa}$, conteniendo la posición de las κ características en la secuencia y la matriz de llenado $\mathbf{F}_{(m-1)\times\kappa}$ indicando en cuáles posiciones de la secuencia aparecen las características

`// Inicializa variables de medicion` $\mathbf{M}$ `y llenado` $\mathbf{F}$
$\mathbf{M} \leftarrow \{\}; \mathbf{F} \leftarrow \{\}$;
`// Procesa las` m `imágenes de la secuencia` $\mathcal{I}$
for $i \leftarrow 1 : m$ **do**
  `// Obtener las posiciones` $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_k\}$ `de las características en la imagen` $\mathbf{I}$ `que son buenas para seguir`
  $\mathbf{P}_{2\times k} =$ `cv.goodFeaturesToTrack` $(\mathbf{I}_i)$;
  if $i = 1$ **then**
    `// Inicializa la variable con características` $\mathbf{Q}$
    $\mathbf{Q}_{2\times\kappa} \leftarrow \mathbf{P}$;
    `// Inicializa la primera hilera de la matriz de mediciones` $\mathbf{M}$
    $\mathbf{M}_{1:2,1:k} \leftarrow \mathbf{P}$;
    `// Inicializa la primera hilera de la matriz de llenado` $\mathbf{F}$
    $\mathbf{F}_{1,1:k} \leftarrow \mathbf{1}_{1\times k}$;
  end
  else
    `// Seguir las características de la imagen` $\mathbf{I}_{i-1}$ `en la imagen` $\mathbf{I}_i$ `donde` $\mathbf{Q}'_{2\times k'}$ `representa la nueva posición de las características y` $\mathbf{s}$ `es una variable que indica si las características fueron seguidas o no.`
    $[\mathbf{Q}', \mathbf{s}] =$ `cv.calcOpticalFlowPyrLK` $(\mathbf{I}_{i-1}, \mathbf{I}_i, \mathbf{Q})$;
    `// Si hay una característica en` $\mathbf{P}$ `muy cerca a la característica en` $\mathbf{Q}$`, que fue seguida, prefiere esta última. La matriz` $\mathbf{P}'$ `contiene las características nuevas`
    $\mathbf{P}'_{2\times k'} \leftarrow$ `Duplicados` $(\mathbf{P}, \mathbf{Q}', \mathbf{s})$;
    `// Actualiza` $\mathbf{F}$`, incluyendo las nuevas` k' `entradas`
    $\mathbf{F} \leftarrow$ `Actualizar_Matriz_Llenado` $(i, \mathbf{F}, \mathbf{s}, \mathbf{1}_{1\times k'})$;
    `// Completa la matriz de mediciones` $\mathbf{M}$ `utilizando las medidas en` $\mathbf{Q}'$`, de las características que fueron seguidas, y las nuevas características` $\mathbf{P}'$
    $\mathbf{M} \leftarrow$ `Actualizar_Matriz_Mediciones` $(i, \mathbf{M}, \mathbf{Q}', \mathbf{s}, \mathbf{P}')$;
    `// Define las nuevas características para seguir` $\mathbf{Q}$
    $\mathbf{Q} \leftarrow$ `Características_por_Seguir` $(i, \mathbf{M}, \mathbf{F})$;
  end
end

Algorithm 1: Algoritmo para seguir características sobre una secuencia de imágenes $\mathcal{I}$. El algoritmo regresa una matriz $\mathbf{M}$ donde se registran las posiciones de las características a lo largo de la secuencia y una matriz $\mathbf{F}$ de llenado, donde se dice cuáles posiciones contienen mediciones en qué imágenes.

características a lo largo de una secuencia de imágenes $\mathcal{I}$ utilizando el seguidor de Lucas-Kanade. Es decir, en cada imagen se estima cuáles características tienen buenas posibilidades de ser seguidas de cuadro a cuadro.

En muchas aplicaciones, tales como la recuperación de la estructura a partir del movimiento, estamos interesados en seguir la historia de la característica conforme la secuencia se desarrolla. Es decir, atendiendo a que la característica corresponde a un punto en el mundo, nos interesa determinar su posición en la imagen a lo largo de la secuencia. El problema se dificulta pues la característica puede cambiar su apariencia visual debido a condiciones tales como el punto de vista, o la iluminación. Asimismo puede ocurrir que la característica quede ocluida. En ese sentido, el algoritmo es flexible a deformaciones de la característica. Sin embargo, corre el riesgo de tomar estructuras que cambien demasiado a lo largo del tiempo, posiblemente perdiendo la forma que originalmente se estaba siguiendo.

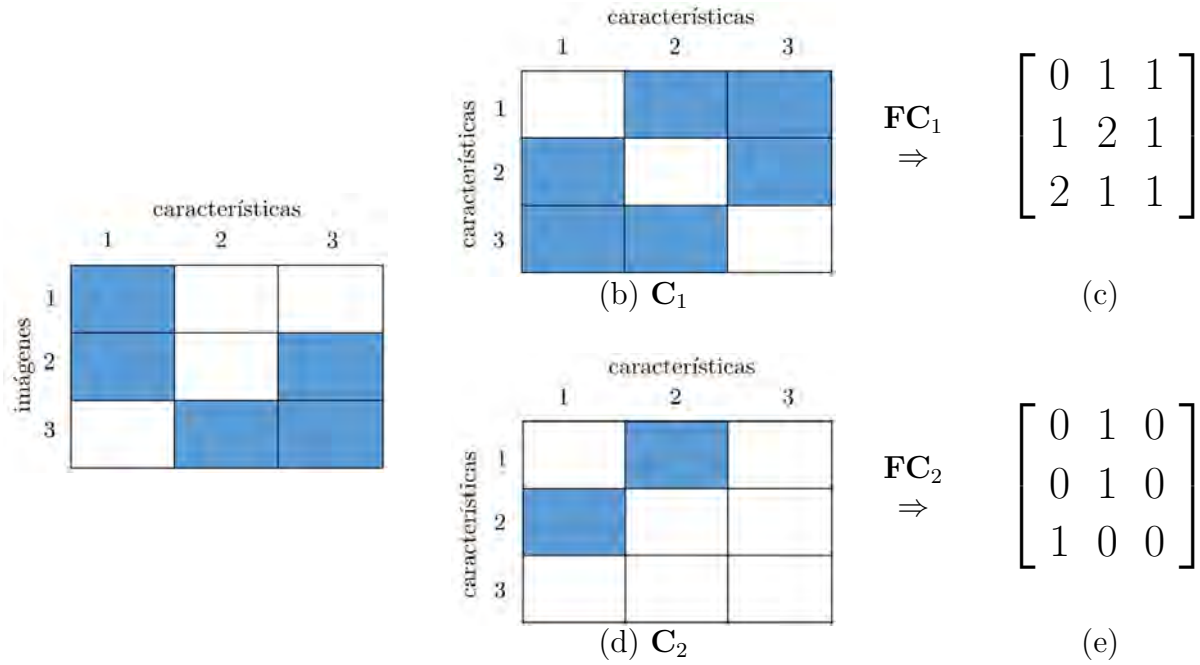
Para ilustrar la situación, imaginemos que obtenemos una línea en particular para cada una de las imágenes de la secuencia que aparece en la Figura 1.3. Esto puede ser ilustrado con una imagen que muestra la evolución de los niveles de intensidad de esa línea a lo largo de la secuencia (ver Figura 1.4(a)). En este problema reducido, supóngase que tenemos la tarea de realizar el seguimiento de características a lo largo de la secuencia. En este caso, la función de disimilaridad podría modificarse y plantearse como

$$\epsilon_2 = \sum_{x \in \mathbf{W}} \left[J(x+d) - I(x) \right]^2, \tag{1.15}$$

donde J y I son funciones unidimensionales sobre x, y $d \in \mathcal{R}$ es el desplazamiento. Inicialmente, las características de buena calidad serían detectadas. Luego, en la segunda imagen, las características serían seguidas usando el esquema de Lucas-Kanade. Por supuesto, algunas de las características podrían perderse debido a motivos tales como que la característica pudiera cambiar mucho visualmente o quedar ocluida. Supóngase que en la segunda imagen se identifican nuevas características. Si las características nuevas y las seguidas están muy cercanas podría preferirse continuar con las segundas. Este proceso pudiera repetirse hasta terminar con la secuencia. Al final, se tendría un resultado similar al que se ilustra en la Figura 1.4(b). De manera complementaria, en la Figura 1.4(c) se representa la matriz de llenado $\mathbf{F}_{n \times m}$. Las hileras de $\mathbf{F}$ representan los cuadros de la secuencia, mientras que las columnas representan las características. Las entradas de $\mathbf{F}(j,i)$ son uno cuando la característica i es detectada en el cuadro j y cero en caso contrario. La matriz de mediciones $\mathbf{M}_{2n \times m}$ es complementaria a la matriz $\mathbf{F}$ e indica la posición de una característica $\mathbf{x} \in \mathcal{R}^2$ a lo largo de la secuencia.

1.5. Compactación de Observaciones

Las características corresponden a puntos en el mundo y, debido a oclusiones o fallas en el seguidor, en ocasiones las matrices $\mathbf{M}$ y $\mathbf{F}$ contienen columnas diferentes para el mismo punto. Dada una matriz de compactación binaria $\mathbf{C}$, que indica en $\mathbf{C}(i,j) = 1$ cuando las trayectorias i y j pertenecen al mismo punto, un problema interesante consiste en poner sobre la misma columna de las matrices $\mathbf{M}$ y $\mathbf{F}$ las observaciones que corresponden al mismo punto en el mundo. Eso permitiría incrementar la robustez de la reconstrucción virtual de



(a) $\mathbf{F}$ (b) $\mathbf{C}_1$ (c) (d) $\mathbf{C}_2$ (e)

Figura 1.5: Compactación de Observaciones. En (a), (b) y (d), el rectángulo azul representa 1 y el rectángulo blanco representa 0. Cuando en $\mathbf{C}_1$ se establece una asociación entre las características 1 y 3, las cuales se traslapan temporalmente en la segunda hilera de $\mathbf{F}$, esto resulta en un valor superior a 1 en el producto $\mathbf{FC}_1$. Por otro lado, en $\mathbf{C}_2$ esta asociación no se establece, resultando en una matriz $\mathbf{FC}_2$ con valores menores o iguales a 1.

objetos y disminuir el espacio que ocupan esas matrices. Ricco y Tomasi [10] proponen una solución, donde simultáneamente resuelven el problema de recuperar la estructura a partir de movimiento y compactan las matrices de observación y llenado. Ellos logran la compactación aplicando restricciones fotométricas, temporales y geométricas.

La restricción fotométrica implica que las características se ven igual bajo la medida de similaridad. Es decir, que el error de la función de error

$$\epsilon_1(\mathbf{A}, \mathbf{d}) = \sum_{\mathbf{x} \in \mathbf{W}} \left[J(\mathbf{A}\mathbf{x} + \mathbf{d}) - I(\mathbf{x}) \right]^2, \tag{1.16}$$

donde $\mathbf{A}_{2\times 2}$ corresponde a una transformación afín, es relativamente pequeño.

La restricción temporal implica que las trayectorias i y j no se traslapan en el tiempo, lo cual se traduce en que $\|\mathbf{FC}\|_\infty = 1$ [6]. Es decir, considere la hilera k de $\mathbf{F}$, la cual contiene información sobre qué características fueron observadas en la imagen k-ésima. Su multiplicación por las columnas de $\mathbf{C}$ debería dar, a lo más, uno. Esto implica que en la columna de $\mathbf{C}$ respectiva no hay más de una asociación entre las características i y j al tiempo que ambas características ocurren simultáneamente en la secuencia (ver Figura 1.5). Una condición temporal adicional busca el establecimiento de una relación de transitividad. Es decir, si las trayectorias i y j se unen, y las trayectorias j y k se unen, las trayectorias i y k deben

[6] $\lim_{p\to\infty} \|f\|_p = \|f\|_\infty = \max\{|f_1|, \ldots, |f_n|\}$

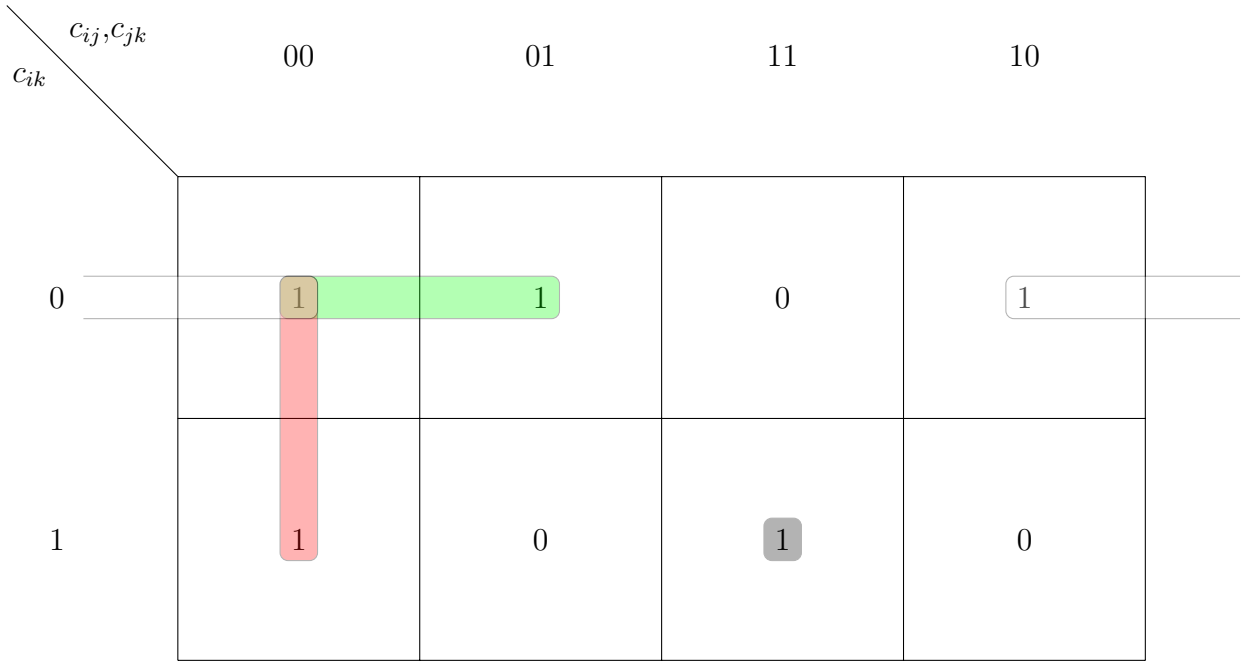


Figura 1.6: Relación de transitividad en la restricción temporal para la compactación. Se ilustra en un diagrama de Karnaugh las restricciones que son aceptables. En adición se agrupan los términos para generar la expresión lógica $(1-c_{ij})(1-c_{jk})+(1-c_{ij})(1-c_{ik})+(1-c_{jk})(1-c_{ik})+c_{ij}c_{jk}c_{ik}\geq 1$.

unirse, o en términos de una ecuación, tenemos (ver Figura 1.6)

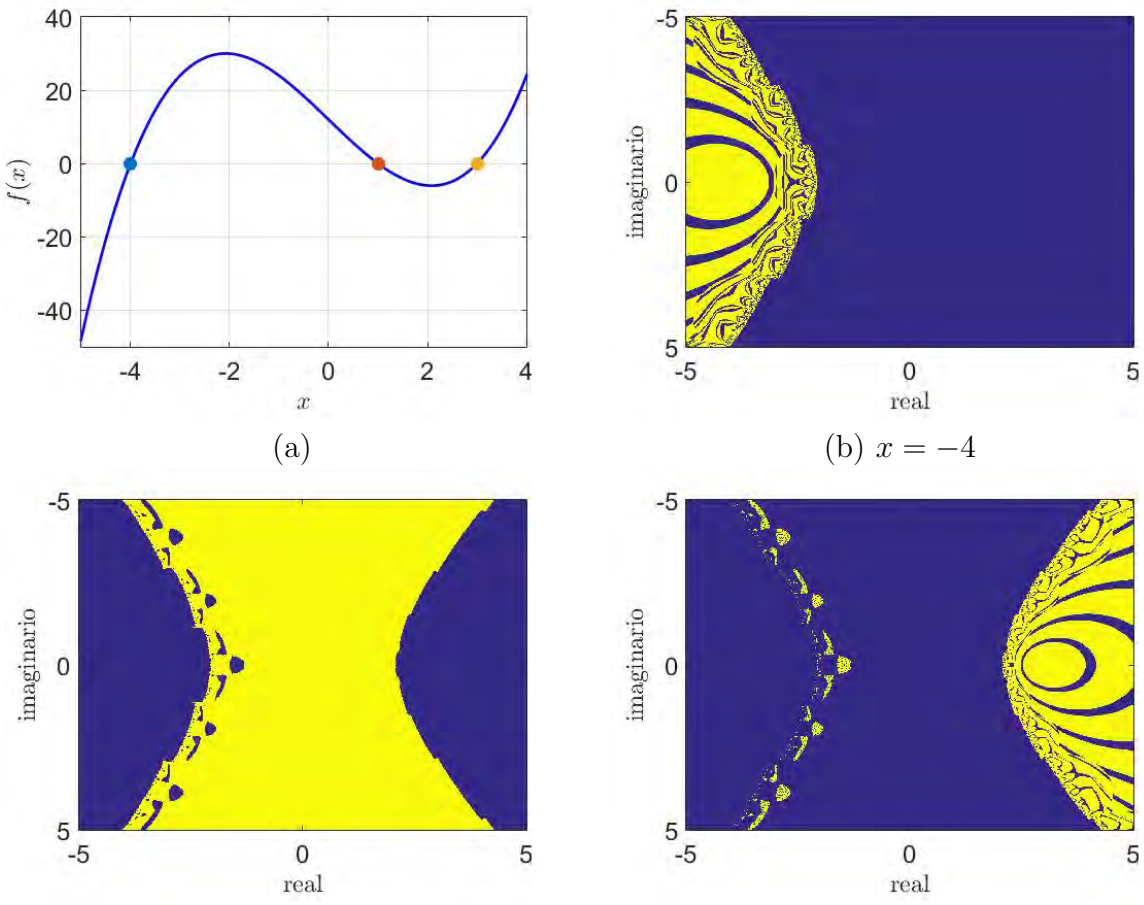
$$(1-c_{ij})(1-c_{jk})+(1-c_{ij})(1-c_{ik})+(1-c_{jk})(1-c_{ik})+c_{ij}c_{jk}c_{ik}\geq 1 \forall i,j,k, \qquad (1.17)$$

donde c_{ij} es una entrada de la matriz $\mathbf{C}$.

Finalmente, la restricción de geometría es aplicada asegurándose que la trayectoria reconstruida aproxima las observaciones de ambos segmentos. En términos matemáticos, se evalúa que $c_{ij}(r_{ik}-r_{jk})=0$. Es decir, si i y j se mantienen separados $c_{ij}=0$, o si son mezclados, las hileras correspondientes de $\mathbf{R}$ en la factorización $\mathbf{M}=\mathbf{L}\mathbf{R}^T$ deben ser idénticas. El método de reconstrucción por factorización será estudiado en el Capítulo 4.

Sumario

Hay un paralelismo muy interesante entre la suma de las diferencias al cuadrado (SSD), las distancias Euclidianas, y la expresión de verosimilitud de funciones estocásticas que siguen un modelo Gaussiano. En ese sentido, SSD trabaja mejor cuando los cambios entre las imágenes son pequeños y el ruido en la imagen sigue un modelo Gaussiano. Con esa suposición, aquí se ha mostrado cómo diversos autores han derivado una formulación similar a Newton-Raphson



(a) (b) $x = -4$

(c) $x = 1$ (d) $x = 3$

Figura 1.7: Cuenca de convergencia. El método de Newton-Raphson tiene cuencas de convergencia hacia $x = -4$((b), $x = 1$ (c), y $x = 3$ (d)) para la función $f(x) = x^3 - 2x - 11x + 12$ (a) que dependen del punto inicial de búsqueda.

para aproximar los parámetros de deformación y desplazamiento de pequeñas regiones en la imagen. Las características que son fácilmente seguibles tienden a parecerse a una esquina y ello vuelve más manejable el problema de encontrar su correspondencia. En otros casos, el problema de la apertura nos indica porqué podemos tener dificultades para seguir a una característica. Aun así, deformaciones visuales bruscas, cambios de iluminación y oclusiones pueden hacer que el seguidor falle. El seguidor de Lucas-Kanade ha sido eficientemente implementado en una variedad de plataformas. En general, proporciona resultados rápidos aunque sobre características dispersas. Uno de los problemas abiertos es la construcción de trayectorias que guarden la historia de cada punto en el mundo. Esto ayudaría en la solución de problemas visuales. Marcadamente, la reconstrucción tridimensional a partir de movimiento.

Ejercicios

1. En (1.1), ¿cúales serían valores razonables para $w(\mathbf{x})$ en una ventana W de 5×5?

2. ¿Cúales serían valores óptimos para los términos $\mathbf{d}$ y $\mathbf{A}$ en (1.18)?

$$\epsilon_1 = \sum_{\mathbf{x} \in \mathbf{W}} \left[J(\mathbf{A}\mathbf{x} + \mathbf{d}) - I(\mathbf{x}) \right]^2, \tag{1.18}$$

   donde $\mathbf{A}_{2\times 2}$ corresponde a una transformación afín. Puede asumir nuevamente que los valores de J y I correspondientes a coordenadas no enteras pueden obtenerse por interpolación.

3. ¿Cuál sería una descomposición en valores singulares válida para la matriz $A_1 = (1, 1)$ y para la matriz $A_2 = (1, 1)^T$?

4. La búsqueda de raíces de la función $f(x) = x^3 - 2x - 11x + 12$ mediante el método de Newton-Raphson tiene cuencas de convergencia a los valores -4, 1, y 3 dependiendo del punto inicial. Esto se ilustra en la Figura 1.7. Dadas las similaridades del método de Newton-Raphson y el método de Lucas-Kanade, ¿sugiere esto algo sobre la existencia de cuencas de convergencia para este último método?

5. Calcula la matriz $\mathbf{P}$ en la construcción del sistema $\mathbf{P}\mathbf{c} = \mathbf{1}$ derivado de la función en (1.17), donde $\mathbf{c}$ es un vector que ordena los elementos de la matriz $\mathbf{C}$ como una columna, $\mathbf{P}$ es una matriz cuyas entradas están en el conjunto $\{-1, 0, 1\}$ y $\mathbf{1}$ es un vector de unos.

6. Implementa el seguidor de características basado en

$$\epsilon_2(d) = \sum_{x \in W} \left[J(x + d) - I(x) \right]^2. \tag{1.19}$$

7. Compara las funciones de evaluación de características de Harris y Stephens, y de Shi y Tomasi. ¿Cuándo dan igual/diferente resultado?

8. Implementa el seguidor de características basado en el Algoritmo 1.

Bibliografía

[1] Anandan, Padmanabhan: *A Computational Framework and an Algorithm for the Measurement of Visual Motion*. International Journal of Computer Vision, 2(3):283–310, 1989.

[2] Baker, Simon y Iain Matthews: *Lucas-Kanade 20 Years on: A Unifying Framework*. International Journal of Computer Vision, 56(3):221–255, 2004.

[3] Bradski, G.: *The OpenCV Library*. Dr. Dobb's Journal of Software Tools, 2000.

[4] Golub, Gene y Charles Van Loan: *Matrix Computations*, volumen 3. JHU Press, 2012.

[5] Harris, Chris y Mike Stephens: *A Combined Corner and Edge Detector*. En *Alvey Vision Conference*, volumen 15, página 50, 1988.

[6] Lucas, Bruce y Takeo Kanade: *An Iterative Image Registration Technique with an Application to Stereo Vision*. En *International Joint Conferences on Artificial Intelligence*, volumen 81, páginas 674–679, 1981.

[7] Moler, Cleve: *Numerical Computing with MATLAB*. SIAM, 2008.

[8] Oron, Shaul, Aharon Bar-Hillel y Shai Avidan: *Extended Lucas Kanade Tracking*. En *European Conference on Computer Vision*, 2014.

[9] Press, William: *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007.

[10] Ricco, Susanna y Carlo Tomasi: *Simultaneous Compaction and Factorization of Sparse Image Motion Matrices*. En *European Conference on Computer Vision*, páginas 456–469. Springer, 2012.

[11] Shi, Jianbo y Carlo Tomasi: *Good Features to Track*. En *IEEE Conference on Computer Vision and Pattern Recognition*, páginas 593–600, 1994.

[12] Strang, Gilbert: *Introduction to Linear Algebra*. Wellesley-Cambridge Press, Massachusetts, 2003.

[13] Yamaguchi, Kota: *mexopencv*. `http://vision.is.tohoku.ac.jp/~kyamagu/software/mexopencv/`, October 2014.

CAPÍTULO 2

Estimación de Movimiento

En Visión por Computadora, el flujo óptico describe a un conjunto de vectores (denso cuando se calcula un vector por cada pixel en la imagen y disperso cuando es el caso del capítulo anterior) que nos da información sobre el movimiento relativo de los objetos y del espectador. El flujo óptico es un elemento que puede ser muy relevante para determinar el arreglo espacial de los objetos, para identificar las discontinuidades que permitieran segmentar las imágenes en regiones que pertenecen a diferentes objetos, o eventualmente para recuperar la estructura de los objetos. Sin embargo, sin disminuir su importancia, conviene tomar en cuenta que lo que percibe la cámara es el reflejo de la luminosidad en la escena. Por ello, en una situación particular la interpretación del flujo óptico puede no corresponder con el mundo físico. Por ejemplo, en las imágenes de una escena podemos percibir que no hay cambios de luminosidad aún cuando los objetos se están moviendo. En ese sentido, Horn y Schunck[5] mencionan el caso de una esfera homogénea que gira mientras es iluminada por una fuente constante. Una situación diferente se presenta cuando un objeto permanece estático aún cuando su imagen cambia. Éste sería el caso, por ejemplo, cuando se tienen objetos especulares sobre los cuales la fuente de luz cambia de posición.

En este documento estudiamos dos estrategias para la recuperación del flujo óptico. Por un lado, tenemos el algoritmo de Horn y Schunck[5], el cual es importante como un ejemplo seminal que estableció muchas de las suposiciones que aún hoy forman la base de una gran cantidad de algoritmos, e introdujo aproximaciones basadas en la regularización, de tal manera que se ha establecido como una referencia en el tema. Por otro lado, estudiamos el algoritmo de Farnebäck[3], el cual se reconoce por su eficiencia y del cual hay implementaciones muy optimizadas y ampliamente disponibles[1].

2.1. La Suposición de Suavidad

En imágenes de intensidad partimos del conocimiento de la cantidad de luz reflejada por la escena e incidente en cierta posición del sensor de la cámara. La cuestión que queremos determinar es ¿hacia dónde se desplazaron los puntos del mundo en subsiguientes imágenes? Horn y Schunck[5] desarrollaron uno de los métodos más populares para el cálculo del flujo óptico bajo la suposición de que el patrón de brillo cambiaba suavemente en casi cualquier lugar de la imagen. El trabajo de Horn y Schunck trata sobre objetos cuyo reflejo de luz está asociado a su movimiento. Es decir, objetos sin sombras cuya iluminación es uniforme sobre toda la superficie. En su caso, asumen que la reflectancia varía suavemente, que no

hay discontinuidades espaciales y no hay oclusiones.

2.1.1. La Ecuación del Flujo

Sea $\mathbf{I}(x, y, t) = k$ una imagen de intensidad, el objetivo es determinar qué sucede cuando el objeto que está en la escena se desplaza tanto espacial como temporalmente de (x, y, t) a $(x + dx, y + dy, t + dt)$. Su expansión en términos de la serie de Taylor (ver (1.2)) puede expresarse como

$$\mathbf{I}(x + dx, y + dy, t + dt) = \mathbf{I}(x, y, t) + dx\frac{\partial \mathbf{I}}{\partial x} + dy\frac{\partial \mathbf{I}}{\partial y} + dt\frac{\partial \mathbf{I}}{\partial t} + \mathcal{O}(dx, dy, dt), \tag{2.1}$$

donde $\mathcal{O}(dx, dy, dt)$ agrupa a los términos cuadráticos y superiores. Una suposición esencial en el método de Horn y Schunck es que el valor de reflexión del punto del objeto en el mundo que genera la imagen en (x, y, t) permanece constante. Es decir,

$$\mathbf{I}(x, y, t) = \mathbf{I}(x + dx, y + dy, t + dt). \tag{2.2}$$

Con ello, y asumiendo que los factores de orden superior al lineal en la serie de Taylor son despreciables, tenemos, al dividir todos los términos de (2.1) entre dt

$$\frac{dx}{dt}\frac{\partial \mathbf{I}}{\partial x} + \frac{dy}{dt}\frac{\partial \mathbf{I}}{\partial y} + \frac{\partial \mathbf{I}}{\partial t} = 0. \tag{2.3}$$

Para simplificar la expresión, se puede realizar el siguiente cambio de variables

$$u = \frac{dx}{dt}, v = \frac{dy}{dt}, I_x = \frac{\partial \mathbf{I}}{\partial x}, I_y = \frac{\partial \mathbf{I}}{\partial y} \text{ y } I_t = \frac{\partial \mathbf{I}}{\partial t}, \tag{2.4}$$

donde u y v representan el desplazamiento espacial de puntos del objeto en el tiempo, I_x e I_y representan el cambio de nivel de intensidad sobre la imagen, e I_t representa el cambio de intensidad de la imagen en el tiempo. Con esto, (2.3) puede reescribirse como

$$I_x u + I_y v + I_t = 0, \tag{2.5}$$

la cual es conocida como la *ecuación de flujo*.

2.1.2. Regularización

La ecuación (2.5) tiene dos incógnitas, u y v. Horn y Schunck hacen la observación de que los puntos de un objeto en movimiento, excepto por los lugares cerca de sus bordes, tienen velocidades similares, y el campo de velocidad del brillo en la imagen varía suavemente. Ellos proponen expresar esta observación mediante el uso de los siguientes gradientes

$$\|\nabla u\|^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2, \text{ y } \|\nabla v\|^2 = \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2, \tag{2.6}$$

y sugieren incluirla como una restricción de la ecuación de flujo. De esta forma, la función objetivo puede definirse como

$$E = \int\int \left((I_x u + I_y v + I_t)^2 + \alpha^2(\|\nabla u\|^2 + \|\nabla v\|^2)\right) dxdy, \tag{2.7}$$

donde α es un multiplicador de Lagrange que define qué tan importante es la suavidad de la imagen, comparada con las mediciones que se hagan. Este funcional puede ser solucionado utilizando cálculo variacional [1].

Sea L el funcional definido por la expresión

$$L = (I_x u + I_y v + I_t)^2 + \alpha^2(\|\nabla u\|^2 + \|\nabla v\|^2). \tag{2.11}$$

Entonces, las restricciones existentes en la ecuación de Euler-Lagrange nos permiten definir el sistema lineal dado por las ecuaciones

$$\frac{\partial L}{\partial u} - \frac{d}{dx}\frac{\partial L}{\partial u_x} - \frac{d}{dy}\frac{\partial L}{\partial u_y} = 0, \text{ y } \frac{\partial L}{\partial v} - \frac{d}{dx}\frac{\partial L}{\partial v_x} - \frac{d}{dy}\frac{\partial L}{\partial v_y} = 0. \tag{2.12}$$

Considerando que $\|\nabla u\|^2 = u_x^2 + u_y^2$, la expresión anterior tiene la solución

$$I_x(I_x u + I_y v + I_t) - \alpha^2 \nabla u = 0, \text{ y } I_y(I_x u + I_y v + I_t) - \alpha^2 \nabla v = 0, \tag{2.13}$$

donde $\nabla = \dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2}$ es el operador Laplaciano. Horn y Schunck proponen que ∇u y ∇v sean aproximados por $\kappa(\overline{u} - u)$ y $\kappa(\overline{v} - v)$ respectivamente, donde $\overline{u}$ y $\overline{v}$ son los promedios pesados de u y v sobre la vecindad ilustrada en la Figura 2.2, y κ es un factor de proporcionalidad igual a 3. Con este cambio de variable, las expresiones en (2.13) pueden ser reescribirse como

$$I_x(I_x u + I_y v + I_t) - \alpha^2(\overline{u} - u) = 0, \text{ y } I_y(I_x u + I_y v + I_t) - \alpha^2(\overline{v} - v) = 0, \tag{2.14}$$

[1] El mapeo de $f : x \to y$ es una función. En contraste, el mapeo $g : f(x) \to y$ es un funcional y x es un parámetro de la función. Un ejemplo tomado de [4] explica: *Considere todas las trayectorias que pueden trazarse desde un punto A a un punto B, ambos en un plano $x - y$. Supóngase que una partícula viaja por alguna de esas trayectorias y en cada punto tiene una velocidad $v(x, y)$. Uno podría plantearse la definición de un funcional w asociando a cada trayectoria el tiempo t que toma recorrerla, i.e., algo como $w : v(x, y) \to t$.*

El cálculo variacional trata con el tema de encontrar los valores óptimos en funcionales. Mucho de su énfasis es resolver la ecuación de Euler-Lagrange, la cual expresa el valor del funcional L a lo largo de la trayectoria definida entre x_1 y x_2, evaluada de acuerdo al comportamiento de $y(x)$. Es decir[4],

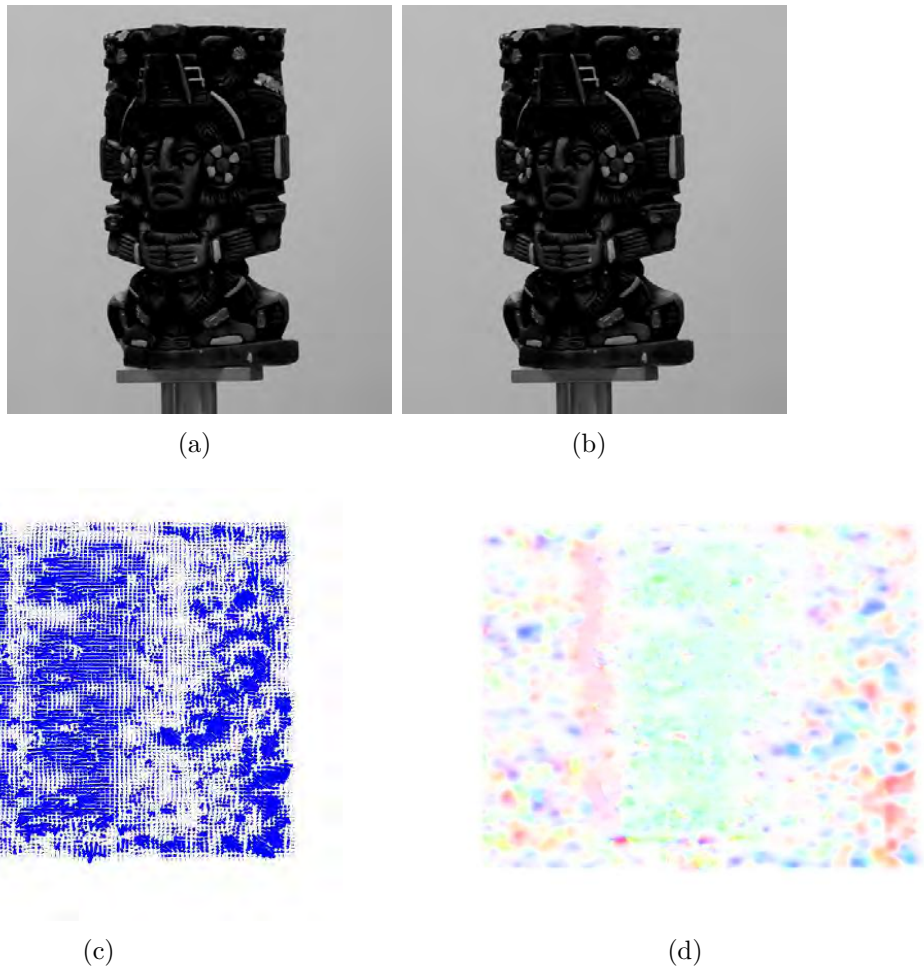
$$J[y] = \int_{x_1}^{x_2} L[x, y(x), y'(x)]dx, \tag{2.8}$$

donde x_1, y x_2 son constantes, y $y(x)$, $y'(x)$ y $L[x, y(x), y'(x)]$ son derivables. Es posible probar[4] que la expresión

$$L_y(x, y(x), y'(x)) - \frac{d}{dx}L_{y'}(x, y(x), y'(x)) = 0, \tag{2.9}$$

es una restricción que se establece en (2.8). Finalmente, para nuestros propósitos se tiene que si $y \in \mathcal{R}^n$, entonces se puede obtener un sistema de ecuaciones diferenciales dado por

$$\frac{\partial L(x, y(x), y'(x))}{\partial y_i} - \frac{d}{dx}\frac{\partial L_{y'}(x, y(x), y'(x))}{\partial y_i'} = 0, \text{ para } i = 1, \ldots, n. \tag{2.10}$$



(a) (b)

(c) (d)

Figura 2.1: Flujo óptico denso usando el algoritmo de Horn y Schunck. En (c) se muestra los vectores correspondientes al flujo óptico (imágenes proporcionadas por Joaquín Santoyo). Similarmente, en (d) se codifica en color la información de magnitud y fase ((d) fue obtenida con el código de visualización en `https://hci.iwr.uni-heidelberg.de/Correspondence_Visualization`)

para un valor apropiado de α. Poniendo las incógnitas del lado izquierdo y las constantes del derecho, llegamos al sistema lineal

$$(I_x^2 + \alpha^2)u + I_x I_y v = \alpha^2 \overline{u} - I_x I_t, \text{ y } I_x I_y u + (I_y^2 + \alpha^2)v = \alpha^2 \overline{v} - I_y I_t, \tag{2.15}$$

el cual tiene como solución

$$u = \frac{(I_y^2 + \alpha^2)\overline{u} - I_x I_y \overline{v} - I_x I_t}{\alpha^2 + I_x^2 + I_y^2}, \text{ y } v = \frac{-I_x I_y \overline{u} + (I_x^2 + \alpha^2)\overline{v} - I_y I_t}{\alpha^2 + I_x^2 + I_y^2}. \tag{2.16}$$

Adicionando y sustrayendo $I_x^2 \overline{u}$ del numerador de la primera expresión, y adicionando y sustrayendo $I_y^2 \overline{v}$ del numerador de la segunda expresión, (2.16) puede ser reducida a

$$u = \overline{u} - \frac{I_x(I_x \overline{u} + I_y \overline{v} + I_t)}{\alpha^2 + I_x^2 + I_y^2}, \text{ y } v = \overline{v} - \frac{I_y(I_x \overline{u} + I_y \overline{v} + I_t)}{\alpha^2 + I_x^2 + I_y^2}. \tag{2.17}$$

Tras lo cual Horn y Schunck plantean el proceso iterativo

$$u_{k+1} = \overline{u}_k - u_k, \text{ y } v_{k+1} = \overline{v}_k - v_k, \tag{2.18}$$

que va actualizando y propagando los valores del flujo sobre toda la imagen.

Llamada : $< \mathbf{U}, \mathbf{V} > \leftarrow$ `Flujo_Optico_Horn_Schunck` $(\mathbf{I}_1, \mathbf{I}_2, \alpha, m)$
Entradas: Imágenes cosecutivas $\mathbf{I}_1$ e $\mathbf{I}_2$, coeficiente de suavizado α, y número de iteraciones para el cálculo del flujo óptico.
Salidas : Flujo óptico $< \mathbf{U}, \mathbf{V} >$ conteniendo la estimación del desplazamiento de un pixel en $\mathbf{x} = (x, y)$ de la imagen $\mathbf{I}_1$ a la imagen $\mathbf{I}_2$.

`// Inicializa aproximación a operador Laplaciano`

$$\mathbf{K} = \begin{pmatrix} 1/12 & 1/6 & 1/12 \\ 1/6 & -1 & 1/6 \\ 1/12 & 1/6 & 1/12 \end{pmatrix};$$

`// Inicializa el valor del flujo óptico`

$\mathbf{U} \leftarrow \mathbf{0}$; $\mathbf{V} \leftarrow \mathbf{0}$;

`// Calcular las diferencias`

$< \mathbf{I}_x, \mathbf{I}_y, \mathbf{I}_t > \leftarrow$ `Calcular_Diferencias` $(\mathbf{I}_1, \mathbf{I}_2)$;

`// Realiza` m `iteraciones para aproximar el flujo óptico`

for $i \leftarrow 1 : m$ **do**

`// Calcular los promedios locales pesados`

$\overline{\mathbf{U}}_k \leftarrow$ `Convolución` $(\mathbf{U}_k, \mathbf{K})$; $\overline{\mathbf{V}}_k \leftarrow$ `Convolución` $(\mathbf{V}_k, \mathbf{K})$;

`// Calcular el flujo óptico siguiendo (2.17) y (2.18)`

$$\mathbf{U}_{k+1} \leftarrow \overline{\mathbf{U}}_k - \frac{\mathbf{I}_x \cdot (\mathbf{I}_x \cdot \overline{\mathbf{U}}_k + \mathbf{I}_y \cdot \overline{\mathbf{V}}_k + \mathbf{I}_t)}{\alpha^2 + \mathbf{I}_x \cdot \mathbf{I}_x + \mathbf{I}_y \cdot \mathbf{I}_y}; \; \mathbf{V}_{k+1} \leftarrow \overline{\mathbf{V}}_k - \frac{\mathbf{I}_y (\mathbf{I}_x \overline{\mathbf{U}}_k + \mathbf{I}_y \overline{\mathbf{V}}_k + \mathbf{I}_t)}{\alpha^2 + \mathbf{I}_x \cdot \mathbf{I}_x + \mathbf{I}_y \cdot \mathbf{I}_y};$$

end

Algorithm 2: Algoritmo para calcular el flujo óptico entre dos imágenes $\mathbf{I}_1$ y $\mathbf{I}_2$ mediante el esquema de Horn y Schunck con un factor de suavizado α. Después de m-iteraciones, el algoritmo regresa dos matrices $\mathbf{U}$ y $\mathbf{V}$ con el desplazamiento horizontal y vertical del pixel la posición $\mathbf{x} = (x, y)$. La operación $\cdot$ refleja una multiplicación punto a punto. La división también es una operación punto a punto en este algoritmo.

2.2. Algoritmo de Farnebäck

Una forma de obtener el desplazamiento denso entre dos imágenes, incluida en las rutinas de OpenCV, es la estimación del movimiento basado en expansión polinomial[3]. En este método, porciones pequeñas de ambas imágenes son aproximadas por un polinomio cuadrático. Enseguida se analiza qué ocurre cuando el polinomio se transforma entre lo que es captado en una imagen y la siguiente. Y finalmente, se realiza la inferencia a porciones de la imagen donde no había información mediante un mecanismo conocido como convolución normalizada. A continuación detallamos las bases sobre las cuales se desarrolla el método.

1/12	1/6	1/12
1/6	-1	1/6
1/12	1/6	1/12

Figura 2.2: Aproximación al operador Laplaciano propuesta por Horn y Schunck[5].

2.2.1. Expansión Polinomial con Factores de Incertidumbre

Una pequeña región en la imagen puede ser representada por una forma polinomial cuadrática de la siguiente forma

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c, \tag{2.19}$$

donde $\mathbf{A}$ es simétrica, $\mathbf{b}$ es un vector, c es un escalar y $\mathbf{x} = (x, y)$ es una posición en la imagen. La expansión de esta expresión resulta en la forma

$$f(\mathbf{x}) = r_1x^2 + r_2xy + r_3y^2 + r_4x + r_5y + r_6, \tag{2.20}$$

lo cual corresponde a una suma de términos en el conjunto $\{x^2, xy, y^2, x, y, 1\}$ escalados por los coeficientes r_i. Dado un conjunto de n puntos $\mathbf{x}_i$, para $i = 1, \ldots, n$, su evaluación en ese conjunto puede ser organizado como una matriz $\mathbf{B}_{n\times 6}$, la cual tendría la forma

$$\mathbf{B} = \begin{pmatrix} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ & & \vdots & & & \\ x_n^2 & x_ny_n & y_n^2 & x_n & y_n & 1 \end{pmatrix}. \tag{2.21}$$

Las columnas de $\mathbf{B}$ pueden ser vistas como la base de un espacio lineal sobre la cual los coeficientes r_j, para $j = 1, \ldots, 6$, se proyectan. Es decir, si $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_6)$, $\mathbf{r} = (r_1, \ldots, r_6)^T$, y $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^T$, entonces uno podría formular el sistema lineal como

$$\sum_{i=1}^{6} \mathbf{b}_i r_i = \mathbf{f}. \tag{2.22}$$

Optimización por Mínimos Cuadrados

El problema entonces es identificar los coeficientes $\mathbf{r}$ que sirven para representar las superficies de porciones de la imagen, representadas por $\mathbf{f}$, por las funciones base definidas en $\mathbf{B}$. Esto puede representarse por la expresión

$$\arg\min_{\mathbf{r}} \|\mathbf{B}\mathbf{r} - \mathbf{f}\|, \tag{2.23}$$

cuya solución de mínimos cuadrados está expresada por

$$\mathbf{r} = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{f}. \tag{2.24}$$

$$\arg\min_{\mathbf{r}} \|\mathbf{W}(\mathbf{B}\mathbf{r} - \mathbf{f})\|. \tag{2.25}$$

cuya expresión de mínimos cuadrados podría escribierse respectivamente como

$$\begin{aligned} \mathbf{r} &= ((\mathbf{W}\mathbf{B})^T\mathbf{W}\mathbf{B})^{-1}(\mathbf{W}\mathbf{B})^T\mathbf{W}\mathbf{f}, \\ &= (\mathbf{B}^T\mathbf{W}^T\mathbf{W}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}^T\mathbf{W}\mathbf{f}, \\ &= \mathbf{N}^{-1}\mathbf{D}, \end{aligned} \tag{2.26}$$

donde

$$\mathbf{N}_{6\times 6} = \mathbf{B}^T\mathbf{W}^T\mathbf{W}\mathbf{B} \quad \text{y} \quad \mathbf{D}_{6\times 1} = \mathbf{B}^T\mathbf{W}^T\mathbf{W}\mathbf{f}. \tag{2.27}$$

Con las expresiones anteriores pudieramos representar una porción de la imagen por medio de una superficie cuadrada donde supondriamos una medida de incertidumbre para los valores de cada observación.

Convolución Normalizada

Farnebäck [2] obtiene los vectores de desplazamiento del flujo óptico denso mediante el proceso que se denomina *convolución normalizada*. La convolución normalizada es un proceso donde se mezclan el ajuste por mínimos cuadrados, las funciones particulares bajo las cuales se hace el análisis (en este caso una parábola) y el manejo de incertidumbre. En la convolución normalizada se tienen los siguientes componentes:

- Por un lado, se tiene la *señal*, $\mathbf{f}_{n\times 1}$, que en nuestro caso son los pixeles que integran una porción de la imagen, cuyos elementos son ordenados como un vector.
- Por otro lado, se tiene la *función base*, $\mathbf{B}_{n\times 6}$, que define el espacio sobre el cual la señal es proyectada. En ocasiones la función base también recibe el nombre de *operador*.
- Luego se tiene la *certidumbre* $\mathbf{c}_{n\times 1}$ sobre los valores de la señal $\mathbf{f}$.
- Finalmente, se tiene la *aplicabilidad* $\mathbf{a}_{n\times 1}$, equivalente a la certidumbre pero definida sobre las observaciones en el operador $\mathbf{B}$.

El vector $\mathbf{a}$ proviene del ordenamiento en columnas de la matriz de aplicabilidad $\mathcal{A}$, que define el tamaño y estructura de la vecindad sobre la cual se hará el análisis. Por su parte, el vector $\mathbf{c}$ proviene de extraer de la matriz de certidumbre, $\mathcal{C}$, la cual está definida sobre toda la imagen $\mathcal{F}$, la porción correspondiente al área de análisis. Las matrices de aplicabilidad,

$\mathcal{A}$, y certidumbre, $\mathcal{C}$, sirven para obtener los valores de la matriz diagonal de incertidumbre en la medición, $\mathbf{W}$, cuyos elementos se definen como

$$\mathbf{W}_{ii}^2(\xi_0) = \mathcal{A}(\mathbf{x}_i)\mathcal{C}(\xi_0 + \mathbf{x}_i), \tag{2.28}$$

para una vecindad centrada en ξ_0 y elementos de posición $\mathbf{x}_i$ en la vecindad. La posición de centrado, ξ_0, constituye el foco del análisis, y la vecindad se define en función de la estructura de $\mathcal{A}$. Aquí se ha hecho una equivalencia en la operación y notación entre $\mathbf{W}^2$ y $\mathbf{W}^T\mathbf{W}$.

En la convolución normalizada[6] hay una separación entre los datos $\mathcal{F}(\xi)$ y la certidumbre de su valor $\mathcal{C}(\xi)$, el operador que se utiliza para realizar el filtraje $\mathbf{B}(\mathbf{x})$ y el operador equivalente a la certidumbre, denominado aplicabilidad $\mathcal{A}(\mathbf{x})$. Por ejemplo, $\mathcal{F}(\xi)$ puede ser una matriz con mediciones y $\mathcal{C}(\xi)$ una matriz que contiene para cada posición de $\mathcal{F}(\xi)$ un número entre cero y uno, que refleja la confianza en la medición.

La relación entre la expresión (2.26) y una convolución es delineada en [2] y puede sumarizarse como sigue. Considere la expresión

$$\mathbf{r} = \tilde{\mathbf{B}}^T\mathbf{f}, \tag{2.29}$$

donde $\tilde{\mathbf{B}}^T = (\mathbf{B}^T\mathbf{W}^2\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}^2$ corresponde a la solución de mínimos cuadrados. Por su parte, cada valor r_i de $\mathbf{r}$ puede expresarse como

$$r_i = \tilde{\mathbf{b}}_i^T\mathbf{f}. \tag{2.30}$$

Haciendo referencia a cada uno de los elementos de los vectores $\tilde{\mathbf{b}}_i$ y $\mathbf{f}$ con el índice de posición $\mathbf{u}$ y haciendo la operación para cada elemento $\mathbf{x}$ de la señal $\mathbf{f}$, se tiene

$$r_i(\mathbf{x}) = \sum_{\mathbf{u}} \tilde{b}_i(\mathbf{u})\mathcal{F}(\mathbf{x} + \mathbf{u}), \tag{2.31}$$

la cual técnicamente corresponde más a una correlación que a una convolución.

En la Figura 2.3 presentamos un ejemplo de lo que se denomina *promediado normalizado*, donde una imagen de 306×408, asumiendo una función base $\mathbf{B}_{n\times 1} = \mathbf{1}$, es convolucionada con función de aplicabilidad $\mathcal{A}$ Gaussiana de 11×11 con desviación estándar $\sigma = 2$, y se asume que $\mathcal{C}$ en conjunción con $\mathcal{F}$ ha sido seleccionada para contener sólo el 10 % de las muestras originales. Las entradas de $\mathcal{C}$ donde $\mathcal{F}$ está definida valen uno, y cero donde no. Este ejemplo es resultado de ejecutar el Algoritmo 3.

2.2.2. Ejemplo de Cálculo de Expansión Polinomial

A continuación elaboramos un ejemplo similar al desarrollado por Farnebäck[2]. Sea una imagen $\mathcal{F}$ y su certidumbre $\mathcal{C}$ definidas como

$$\mathcal{F} = \begin{pmatrix} 4 & 3 & 0 & 9 & 1 \\ 7 & 3 & 6 & 3 & 8 \\ 0 & 5 & 4 & 6 & 0 \\ 3 & 4 & 5 & 8 & 4 \\ 1 & 6 & \textcircled{1} & 8 & 9 \\ 0 & 2 & 1 & 0 & 5 \\ 1 & 8 & 8 & 0 & 6 \end{pmatrix} \quad \mathcal{C} = \begin{pmatrix} 0.3 & 0.2 & 0.1 & 0.5 & 0.4 \\ 0.6 & 0.7 & 0.1 & 0.1 & 0.1 \\ 0.7 & 0.1 & 0.6 & 0.5 & 0.5 \\ 0.1 & 0.4 & 0.2 & 0.6 & 0.6 \\ 0.7 & 0.8 & \textcircled{0.2} & 0.1 & 0.5 \\ 0.8 & 0.3 & 0.4 & 0.4 & 0.8 \\ 0.7 & 0.2 & 0.1 & 0.6 & 0.5 \end{pmatrix}, \tag{2.32}$$

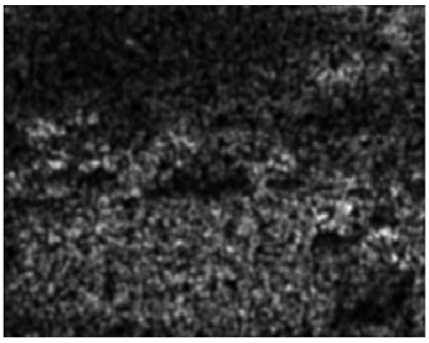
Llamada : $\mathbf{G} \leftarrow$ `Convolución_Normalizada` $(\mathcal{F}, \mathcal{C}, \mathbf{B})$
Entradas: Imagen $\mathcal{F}_{m,n}$, correspondiente certidumbre sobre el valor de cada pixel en la imagen $\mathcal{C}_{m \times n}$ y función base $\mathbf{B}$.
Salidas : Imagen $\mathbf{G}_{m \times n}$ con resultado de aplicar la convolución normalizada a $\mathbf{F}$ bajo la certidumbre $\mathbf{C}$ y el modelo de representación local $\mathbf{B}$.

`// tamaño de la ventana de aplicabilidad`
$k \leftarrow \alpha$; $l \leftarrow 2k + 1$;
`// Define la aplicabilidad como los valores de una distribución normal sobre una ventana de` $n \times n$
$\mathcal{A}_{l \times l} \leftarrow \mathcal{N}(\mathbf{0}, \Sigma)$; $\mathbf{a}_{l^2 \times 1} \leftarrow vec(\mathcal{A})$;
`// Centrar la operación de convolución en cada punto` $(\overline{x}, \overline{y})$ `en la imagen[6].`
for $\overline{y} = k + 1 : n - k$ **do**
  for $\overline{x} = k + 1 : m - k$ **do**
    `// Toma la porción de la imagen correspondiente a la aplicabilidad`
    $\mathbf{f} = vec(\mathcal{F}(y - k : y + k, x - k : x + k))$;
    `// Toma la porción de la certidumbre correspondiente a la aplicabilidad`
    $\mathbf{c} = vec(\mathcal{C}(y - k : y + k, x - k : x + k))$;
    `// Construye la matriz de pesos`
    $\mathbf{W}_a = diag(\mathbf{a})$; $\mathbf{W}_c = diag(\mathbf{c})$;
    `// Obtener la convolución normalizada`
    $\mathbf{N} = \mathbf{B}^T \mathbf{W}_a \mathbf{W}_c \mathbf{B}$; $\mathbf{D} = \mathbf{B}^T \mathbf{W}_a \mathbf{W}_c \mathbf{f}$;
    `// Si` $\mathbf{N}^{-1}$ `existe`
    $\mathbf{G}(y, x) \leftarrow \mathbf{N}^{-1} \mathbf{D}$;
  end
end

Algorithm 3: Algoritmo de convolución normalizada. Dada una imagen $\mathcal{F}$, la certidumbre sobre los valores de cada pixel que la integra $\mathcal{C}$ y el modelo de transformación $\mathbf{B}$, la convolución normalizada sopesa la evidencia presente utilizando una región de aplicabilidad ponderada $\mathcal{A}$.

(a) Imagen Original

(b) Muestras disponibles, $\mathcal{F}$



(c) Convolución

(d) Convolución Normalizada

Figura 2.3: Convolución Normalizada. Supóngase que de una imagen sólo se tiene una muestra de datos. La convolución normalizada permite aproximar los datos faltantes asumiendo un valor de certidumbre para el análisis. Aquí se asume una función base $\mathbf{B} = 1$, una función de aplicabilidad $\mathcal{A}$ Gaussiana. Las muestras disponibles de $\mathcal{F}$ seleccionadas a través de la función de certidumbre $\mathcal{C}$, equivalen al 10 % de las muestras originales. La certidumbre $\mathcal{C}$ vale uno donde $\mathcal{F}$ está definida, y cero donde no.

donde el número en un círculo indica dónde se estará centrando el análisis.

La aplicabilidad $\mathcal{A}$ delimita la región espacial sobre la cual se realizan los cálculos. En el caso de este ejemplo, la aplicabilidad $\mathcal{A}$ se define sobre una vecindad de 3×3 con los siguientes valores

$$\mathcal{A} = \begin{pmatrix} 0.07 & 0.13 & 0.07 \\ 0.13 & 0.2 & 0.13 \\ 0.07 & 0.13 & 0.07 \end{pmatrix}, \tag{2.33}$$

donde la suma de los coeficientes es igual a 1. Dada una base polinomial de la forma $\{x^2, xy, y^2, x, y, 1\}$, el operador $\mathbf{B}$ que aglutina las funciones base en la vecindad dada por el espacio

$$\mathcal{X} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad \text{y} \quad \mathcal{Y} = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}, \tag{2.34}$$

y la aplicabilidad $\mathbf{a}$, se definen como

$$\mathbf{B} = \begin{pmatrix} 1 & 1 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 \\ 1 & -1 & 1 & 1 & -1 & 1 \\ 1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \text{y} \quad \mathbf{a} = \begin{pmatrix} 0.07 \\ 0.13 \\ 0.07 \\ 0.13 \\ 0.20 \\ 0.13 \\ 0.07 \\ 0.13 \\ 0.07 \end{pmatrix}, \tag{2.35}$$

donde los valores de x y y se obtienen respectivamente de las matrices $\mathcal{X}$ y $\mathcal{Y}$.

Igualmente, la señal $\mathbf{f}$ y la certidumbre $\mathbf{c}$ tendrán los siguientes valores

$$\mathbf{f} = \begin{pmatrix} 4 \\ 6 \\ 2 \\ 5 \\ 1 \\ 1 \\ 8 \\ 8 \\ 0 \end{pmatrix} \quad \text{y} \quad \mathbf{c} = \begin{pmatrix} 0.4 \\ 0.8 \\ 0.3 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.6 \\ 0.1 \\ 0.4 \end{pmatrix}. \tag{2.36}$$

Aplicando (2.26), y dado que

$$\mathbf{W}^2 = \texttt{diag}(0.034, 0.104, 0.023, 0.030, 0.057, 0.061, 0.052, 0.021, 0.031), \tag{2.37}$$

el cual se obtiene con (2.28), se tiene

$$\begin{aligned} \mathbf{r} &= (\mathbf{B}^T\mathbf{W}^2\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}^2\mathbf{f}, \\ &= \begin{pmatrix} 0.19 & -0.01 & 0.11 & 0.00 & 0.02 & 0.19 \\ -0.01 & 0.11 & -0.01 & 0.02 & -0.03 & -0.01 \\ 0.11 & -0.01 & 0.23 & -0.03 & -0.08 & 0.23 \\ 0.00 & 0.02 & -0.03 & 0.19 & -0.01 & 0.00 \\ 0.02 & -0.03 & -0.07 & -0.01 & 0.23 & -0.08 \\ 0.19 & -0.01 & 0.23 & 0.00 & -0.08 & 0.35 \end{pmatrix}^{-1} \begin{pmatrix} 0.67 \\ -0.27 \\ 1.21 \\ -0.49 \\ -0.34 \\ 1.44 \end{pmatrix} = \begin{pmatrix} -1.23 \\ -1.59 \\ 2.44 \\ -2.07 \\ 0.13 \\ 3.15 \end{pmatrix}. \end{aligned} \tag{2.38}$$

Conviene notar que la relación entre los coeficientes de $\mathbf{r}$ y los elementos de la expresión cuadrática es

$$c = r_1, \quad \mathbf{b} = \begin{pmatrix} r_2 \\ r_3 \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} r_4 & r_6/2 \\ r_6/2 & r_5 \end{pmatrix}. \tag{2.39}$$

2.2.3. Estimación del Desplazamiento

Una vez que las regiones alrededor de cada pixel han sido representadas por una función cuadrática, el flujo óptico se calcula con base en su deformación entre cuadros sucesivos. Es decir, sea la expresión cuadrática

$$f_1(\mathbf{x}) = \mathbf{x}^T \mathbf{A}_1 \mathbf{x} + \mathbf{b}_1^T \mathbf{x} + c_1. \tag{2.40}$$

Supongamos ahora que se presenta un desplazamiento $\mathbf{d}$. Esto puede expresarse como la función f_2 tal que [3]

$$\begin{aligned} f_2(\mathbf{x}) &= f_1(\mathbf{x}+\mathbf{d}) \\ &= (\mathbf{x}+\mathbf{d})^T \mathbf{A}_1 (\mathbf{x}+\mathbf{d}) + \mathbf{b}_1^T (\mathbf{x}+\mathbf{d}) + c_1, \\ &= \mathbf{x}^T \mathbf{A}_1 \mathbf{x} + (\mathbf{b}_1 + 2\mathbf{A}_1\mathbf{d})^T \mathbf{x} + \mathbf{d}^T \mathbf{A}_1 \mathbf{d} + \mathbf{b}_1^T \mathbf{d} + c_1, \\ &= \mathbf{x}^T \mathbf{A}_2 \mathbf{x} + \mathbf{b}_2^T \mathbf{x} + c_2, \end{aligned} \tag{2.41}$$

recordando que $\mathbf{A}_1$ es simétrica, por construcción, y donde se establecen las igualdades

$$\mathbf{A}_2 = \mathbf{A}_1, \mathbf{b}_2 = \mathbf{b}_1 + 2\mathbf{A}_1\mathbf{d}, c_2 = \mathbf{d}^T \mathbf{A}_1 \mathbf{d} + \mathbf{b}_1^T \mathbf{d} + c_1. \tag{2.42}$$

Farnebäck hace la observación de que el desplazamiento puede ser calculado directamente de la ecuación anterior, resultando en

$$\mathbf{d} = \tfrac{1}{2}\mathbf{A}_1^{-1}(\mathbf{b}_2 - \mathbf{b}_1). \tag{2.43}$$

Estimador de Desplazamiento

En la práctica, se utilizan aproximaciones a cuadráticas en ambas imágenes. Por ello, el mejor estimador para la matriz $\mathbf{A}$ resulta de obtener el valor medio entre ambas aproximaciones, tal como[3]

$$\mathbf{A}(\mathbf{x}) = \frac{\mathbf{A}_1(\mathbf{x}) + \mathbf{A}_2(\mathbf{x})}{2}. \tag{2.44}$$

Así, si se hace el cambio de variable

$$\Delta\mathbf{b}(\mathbf{x}) = (\mathbf{b}_2(\mathbf{x}) - \mathbf{b}_1(\mathbf{x}))/2, \tag{2.45}$$

tenemos que el valor de $\mathbf{d}(\mathbf{x})$ del sistema lineal, construido de (2.43),

$$\mathbf{A}(\mathbf{x})\mathbf{d}(\mathbf{x}) = \Delta\mathbf{b}(\mathbf{x}), \tag{2.46}$$

puede ser encontrado minimizando la expresión

$$e(\mathbf{x}) = \sum_{\Delta\mathbf{x}} w(\Delta\mathbf{x}) \|\mathbf{A}(\mathbf{x}+\Delta\mathbf{x})\mathbf{d}(\mathbf{x}+\Delta\mathbf{x}) - \Delta\mathbf{b}(\mathbf{x}+\Delta\mathbf{x})\|^2, \tag{2.47}$$

donde $\Delta\mathbf{x}$ representa los índices de una vecindad alrededor de $\mathbf{x}$. El valor óptimo se obtiene derivando con respecto a $\mathbf{d}$ e igualando a cero, lo que resulta en la expresión de mínimos cuadrados

$$\mathbf{d}(\mathbf{x}) = \left(\sum_{\Delta\mathbf{x}} w(\Delta\mathbf{x}) \mathbf{A}(\mathbf{x}+\Delta\mathbf{x})^T \mathbf{A}(\mathbf{x}+\Delta\mathbf{x}) \right)^{-1} \sum_{\Delta\mathbf{x}} w(\Delta\mathbf{x}) \mathbf{A}(\mathbf{x}+\Delta\mathbf{x})^T \Delta\mathbf{b}(\mathbf{x}+\Delta\mathbf{x}). \quad (2.48)$$

Finalmente, una solución iterativa requerirá la incorporación de conocimiento *a priori*. Es decir, después de un cierto número de imágenes, el desplazamiento inicial puede ser diferente de cero. En este caso, el valor inicial estimado del desplazamiento puede estar dado por

$$\tilde{\mathbf{x}} = \mathbf{x} + \tilde{\mathbf{d}}(\mathbf{x}), \quad (2.49)$$

donde $\tilde{\mathbf{d}}(\mathbf{x})$ es el desplazamiento hasta el cuadro anterior. Esto modifica las expresiones (2.44) y (2.45) en las expresiones

$$\mathbf{A}(\mathbf{x}) = \frac{\mathbf{A}_1(\mathbf{x}) + \mathbf{A}_2(\tilde{\mathbf{x}})}{2}, \quad (2.50)$$

y

$$\Delta\mathbf{b}(\mathbf{x}) = (\mathbf{b}_2(\tilde{\mathbf{x}}) - \mathbf{b}_1(\mathbf{x}))/2 + \mathbf{A}(\mathbf{x})\tilde{\mathbf{d}}(\mathbf{x}). \quad (2.51)$$

Fuentes de Incertidumbre

Farnebäck[2] considera tres fuentes de incertidumbre para el cálculo del desplazamiento, los cuales tienen que ver con los valores de desplazamiento, el desplazamiento estimado y la cercanía con los bordes de la imagen. Aquí modificamos el primer criterio para relacionarlo con la similaridad entre las vecindades de donde surgen las expansiones polinomiales, las cuales se expresan en (2.47). Una posible expresión puede ser

$$\mathcal{C}_1(\mathbf{x}) = \exp\left(-\alpha e(\mathbf{x})\right), \quad (2.52)$$

para una constante α. Por otro lado, limita la cantidad de movimiento que puede haber en una imagen a

$$\mathcal{C}_2(\mathbf{x}) = \begin{cases} 1 & d_{\text{mín}} \leq \|\mathbf{d}(\mathbf{x})\| \leq d_{\text{máx}}, \\ 0 & \text{en otro caso.} \end{cases} \quad (2.53)$$

Finalmente, limita la certidumbre al tamaño de la región de aplicabilidad. Si el tamaño de la región $\mathcal{A}$ es de $N \times N$, entonces la condición de certidumbre está dada por

$$\mathcal{C}_3(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \text{ está a menos de } (N-1)/2 \text{ pixeles de la orilla,} \\ 0 & \text{en otro caso.} \end{cases} \quad (2.54)$$

Y la certidumbre está dada por la expresión

$$\mathcal{C}(\mathbf{x}) = \mathcal{C}_1(\mathbf{x})\mathcal{C}_2(\mathbf{x})\mathcal{C}_3(\mathbf{x}). \quad (2.55)$$

Llamada : $\langle \mathcal{U}, \mathcal{V} \rangle \leftarrow$ `Flujo_Optico_Farnebäck` $(\mathbf{I}_l, \mathbf{I}_r, m)$
Entradas: Imágenes consecutivas $\mathbf{I}_l$ e $\mathbf{I}_r$ y número de iteraciones m
Salidas : Flujo óptico $< \mathcal{U}, \mathcal{V} >$ conteniendo la estimación del desplazamiento de un pixel en $\mathbf{x} = (x, y)$ de la imagen $\mathbf{I}_l$ a la imagen $\mathbf{I}_r$

```
// Inicializa el flujo óptico resultante
```
$\mathcal{U} \leftarrow \mathbf{0}$; $\mathcal{V} \leftarrow \mathbf{0}$;
```
// Calcular la expansión polinomial de las imágenes
```
$\langle \mathbf{A}_l, \mathbf{b}_l, c_l \rangle \leftarrow$ `Expansion_Polinomial`$(\mathbf{I}_l)$;
$\langle \mathbf{A}_r, \mathbf{b}_r, c_r \rangle \leftarrow$ `Expansion_Polinomial`$(\mathbf{I}_r)$;
```
// Realiza un cierto número de iteraciones m
```
for $i \leftarrow 1 : m$ **do**
```
    // Calcular A y Δb de acuerdo a (2.44) y (2.45)
```
    $\langle \mathbf{A}, \Delta\mathbf{b} \rangle \leftarrow$ `Aproximar_Términos` $(\mathbf{A}_l, \mathbf{A}_r, \mathbf{b}_l, \mathbf{b}_r)$;
```
    // Obtener el vector de desplazamiento d resolviendo el sistema
       lineal expresado en (2.48)
```
    $\mathbf{d} \leftarrow$ `Calcular_Desplazamiento` $(\mathbf{A}, \Delta\mathbf{b})$;
```
    // Calcular el valor de certidumbre mediante (2.55)
```
    $\mathcal{C} \leftarrow$ `Estimar_Incertidumbre` $(\mathbf{A}, \Delta\mathbf{b}, \mathbf{d})$;
```
    // Obtener el promediado normalizado (descrito en §2.2.1) a los
       componentes del desplazamiento usando aplicabilidad Gaussiana
       y la certidumbre C.
```
    $\langle \mathbf{U}, \mathbf{V} \rangle \leftarrow$`Convolución_Normalizada` $(\mathbf{d}, \mathcal{C})$;
```
    // Actualiza el flujo óptico resultante
```
    $\mathcal{U} \leftarrow \mathcal{U} + \mathbf{U}$; $\mathcal{V} \leftarrow \mathcal{V} + \mathbf{V}$;
```
    // Calcular la expansión polinomial de las imágenes
```
    $\langle \mathbf{A}_r, \mathbf{b}_r, c_r \rangle \leftarrow$ `Expansion_Polinomial`$(\mathbf{I}'_r, \mathcal{U}, \mathcal{V})$;
end

Algorithm 4: Algoritmo de Farnebäck para calcular el flujo óptico entre dos imágenes $\mathbf{I}_l$ y $\mathbf{I}_r$. El algoritmo regresa dos matrices $\mathcal{U}$ y $\mathcal{V}$ con el desplazamiento horizontal y vertical del pixel la posición $\mathbf{x} = (x, y)$. En el algoritmo original existe la búsqueda del flujo utilizando multiples escalas y un modelo subyacente del movimiento, ambos aspectos no son tratados en el algoritmo que se presenta.

2.3. Implementación

El Algoritmo 2 presenta unas líneas de pseudocódigo ilustrando el esquema de Horn y Schunck para el cálculo del flujo óptico entre dos imágenes. En la Figura 2.1 se ilustran diferentes fases del funcionamiento del algoritmo para una par de imágenes de una estatua que rota sobre su eje vertical.

El algoritmo de Farnebäck puede ser sumarizado por las líneas de pseudocódigo en el Algoritmo 4. El algoritmo ha sido implementado en muy diversas plataformas. Por ejemplo, se encuentra disponible en OpenCV, donde podría utilizarse en conjunción con el *mexopencv* mediante una llamada a la rutina $\mathbf{F} = \texttt{cv.calcOpticalFlowFarneback}(\mathbf{I}_1, \mathbf{I}_2)$, donde $\mathbf{I}_1$ y $\mathbf{I}_2$ serían dos imágenes. Alternativamente, uno podría utilizar la implementación disponible en Matlab de MathWorks (ver Figura 2.4).

Sumario

El algoritmo de Horn y Schunck parte del supuesto de que prácticamente en todos lados de la imagen, excepto en las fronteras de movimiento, el flujo varía suavemente. Ellos introducen una función objetivo que incluye una codependencia entre la variación espacial de los objetos y el flujo óptico, por un lado, y la restricción de suavidad por el otro. Ambas suposiciones aún son utilizadas ampliamente en la comunidad de Visión por Computadora.

El algoritmo de Farnebäck para el cálculo denso de flujo óptico se basa en la expansión polinomial de pequeñas regiones de la imagen. Una característica sobresaliente del método es que en su formulación se separa lo que se mide de la certidumbre que se tiene sobre lo que es medido. El algoritmo permite la incorporación de diversos modelos de movimiento. OpenCV lo incorpora como parte de las rutinas implementadas en la librería. Una buena implementación del algoritmo se encuentra en Matlab de MathWorks (ver Figura 2.4).

Ejercicios

1. Muestra la equivalencia entre

$$u = \frac{(I_y^2 + \alpha^2)\overline{u} - I_x I_y \overline{v} - I_x I_t}{\alpha^2 + I_x^2 + I_y^2}, \text{ y } v = \frac{-I_x I_y \overline{u} + (I_x^2 + \alpha^2)\overline{v} - I_y I_t}{\alpha^2 + I_x^2 + I_y^2}, \tag{2.56}$$

   y

$$u = \overline{u} - \frac{I_x(I_x\overline{u} + I_y\overline{v} + I_t)}{\alpha^2 + I_x^2 + I_y^2}, \text{ y } v = \overline{v} - \frac{I_y(I_x\overline{u} + I_y\overline{v} + I_t)}{\alpha^2 + I_x^2 + I_y^2}. \tag{2.57}$$

2. ¿Bajo qué condiciones funciona/falla el algoritmo de Horn y Schunck?

3. ¿Qué criterios utilizarías para segmentar una escena con base en el tipo de movimiento? Justifica tu respuesta.

4. Prueba el algoritmo de Horn y Schunck, y de Farnebäck, para un movimiento fronto-paralelo de un objeto rígido frente a una cámara.

5. Obtén la solución de los coeficientes de la expansión polinomial cuadrática para estructuras de la forma

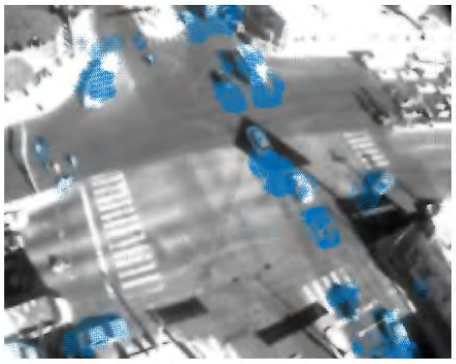
$$\mathcal{F} = \begin{pmatrix} 15 & 15 & 1 & -1 & 0 \\ 14 & 14 & -2 & 0 & 0 \\ 15 & 14 & 1 & -2 & 1 \\ -1 & 0 & 16 & 15 & 16 \\ -1 & 0 & \textcircled{15} & 13 & 14 \\ 1 & -1 & 13 & 15 & 15 \\ 0 & -1 & 14 & 14 & 14 \end{pmatrix} \quad \mathcal{C} = \begin{pmatrix} 1 & 3 & 3 & 1 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 4 & 5 & 3 & 3 \\ 3 & 4 & 4 & 3 & 3 \\ 1 & 3 & \textcircled{3} & 1 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 1 & 3 & 3 & 1 & 1 \end{pmatrix}. \tag{2.58}$$

con aplicabilidad $\mathcal{A}$ definida sobre una vecindad de 3×3 con los siguientes valores

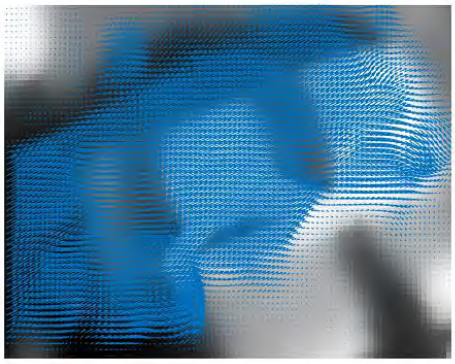
$$\mathcal{A} = \begin{pmatrix} 0.07 & 0.13 & 0.07 \\ 0.13 & 0.2 & 0.13 \\ 0.07 & 0.13 & 0.07 \end{pmatrix}, \tag{2.59}$$

6. Implementa el estimador de flujo óptico basado en el Algoritmo 2.

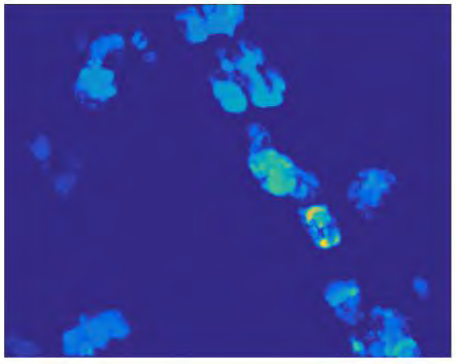
7. Implementa el estimador de flujo óptico basado en el Algoritmo 4.



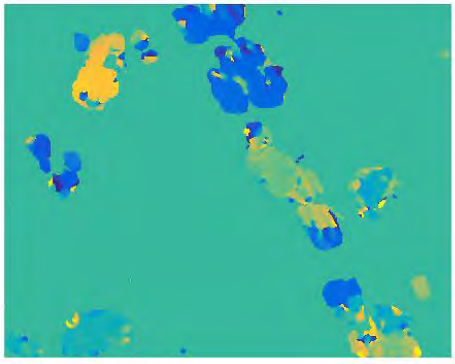
(a) Flujo Optico



(b) Detalle de Flujo Optico



(c) Magnitud de Flujo Optico



(d) Dirección del Flujo Optico

Figura 2.4: Flujo óptico usando el algoritmo de Farnebäck implementado por MathWorks. Farnebäck calcula el flujo óptico aproximando pequeñas regiones de la imagen por superficies cuadráticas y analizando la deformación de una imagen a la siguiente. Característico en su método, se hace una distinción entre las mediciones y la certidumbre que se tiene sobre las mediciones. El análisis se realiza sobre una pirámide de imágenes para describir desplazamientos largos. En las imágenes (c) y (d) no se muestran resultados para desplazamientos menores a un pixel.

Bibliografía

[1] Bradski, G.: *The OpenCV Library*. Dr. Dobb's Journal of Software Tools, 2000.

[2] Farnebäck, Gunnar: *Polynomial Expansion for Orientation and Motion Estimation*. Tesis de Doctorado, Linköping University, 2002.

[3] Farnebäck, Gunnar: *Two-Frame Motion Estimation based on Polynomial Expansion*. En *Image Analysis*, páginas 363–370. Springer, 2003.

[4] Gelfand, Izrail y Sergei Fomin: *Calculus of Variations*. Dover Publications, 2000.

[5] Horn, Berthold y Brian Schunck: *Determining Optical Flow*. En *Technical Symposium East*, páginas 319–331. International Society for Optics and Photonics, 1981.

[6] Knutsson, Hans y C. Westin: *Normalized and Differential Convolution*. En *IEEE Conference on Computer Vision and Pattern Recognition*, páginas 515–523, 1993.

CAPÍTULO 3

Caracterización Local

Una herramienta muy utilizada en Visión por Computadora para caracterizar localmente los objetos en las imágenes son los llamados *descriptores SIFT* (*Scale Invariant Feature Transform*), los cuales fueron introducidos por Lowe[10]. Los SIFT son descriptores invariantes a cambio de escala, orientación, y parcialmente invariantes a distorsiones afines (ver Figura 3.1). Además son bastante robustos a cambios de iluminación y algo tolerantes al ruido en las imágenes.

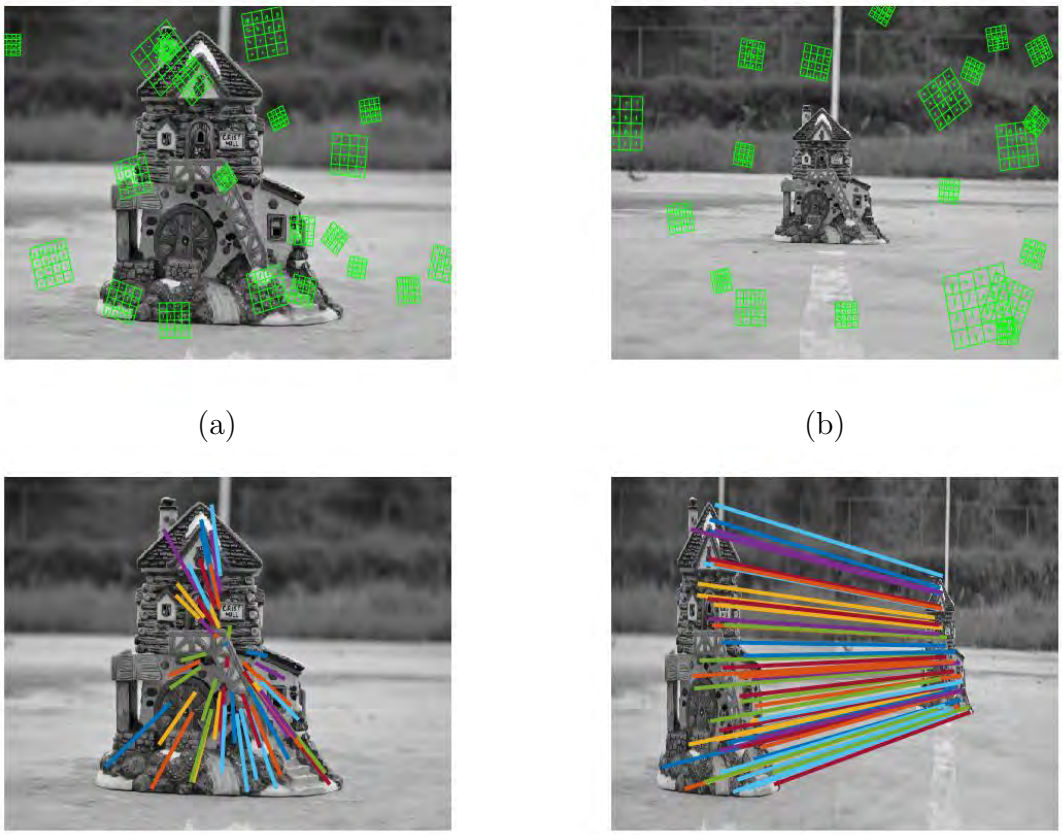
En este capítulo, realizamos una introducción a las características SIFT y las utilizamos para una aplicación de reconocimiento individual de jaguares. Es decir, dada una base de datos de individuos, nos interesa determinar si estamos viendo una foto del jaguar *zutanito* o *perenganita*. En este tipo de aplicaciones, la presencia de algunas características SIFT puede tomarse como evidencia convincente de la presencia de un individuo en particular, posibilitando su identificación aún cuando hay oclusiones parciales. Una limitación de las características SIFT para algunas aplicaciones es que aunque pueden usarse para describir objetos flexibles, su mejor desempeño se obtiene al trabajar con objetos rígidos. Otra limitación es que aún cuando son algo tolerantes a transformaciones afines que pudieran modelar algún cambio en un objeto tridimensional, realmente han sido diseñadas para trabajar con objetos bidimensionales. Durante el reconocimiento, se busca la correspondencia de una característica con aquellas almacenadas en una base de datos y que se sabe existen en el objeto de interés. Luego, un esquema de votación permite identificar acumulaciones relacionadas con un objeto en particular. La verificación normalmente consiste en encontrar parámetros geométricos que expresen transformaciones coherentes.

3.1. Extracción de Características

Las etapa de extracción de características incluye la detección de puntos estables a cambios de escala, el refinamiento de la localización de la característica, la asignación de la orientación, y finalmente la obtención del descriptor de la característica. A continuación detallamos cada uno de estos procesos.

3.1.1. Regiones Estables en Cambios de Escala

En esta etapa se buscan características que permanezcan estables sobre un amplio rango de escalas. En relación con ello, Koenderink [7] plantea que siempre y cuando una restricción de imágenes de menor resolución sea que las características que aparecen en ellas tengan su



(a) (b)

(c) (d)

Figura 3.1: Caracterización Local. El objetivo de la caracterización local es obtener un conjunto de descriptores a partir de la imagen. En el caso de las características SIFT ofrecen invarianza a cambio de escala, traslación, orientación, y parcialmente a transformaciones afines. Además son bastante robustos a cambios de iluminación y algo tolerantes al ruido en las imágenes. Aquí ilustramos una muestra de 20 las características extraídas de imágenes tomadas a diferentes escalas (a) y (b) (el total es de 977 y 1350 respectivamente). En (c) se muestra el punto de fuga mientras que en (d) la correspondiente entre características de ambas imágenes. Los resultados se obtuvieron usando `vlfeat`[13]. Imágenes proporcionadas por Rubén García.

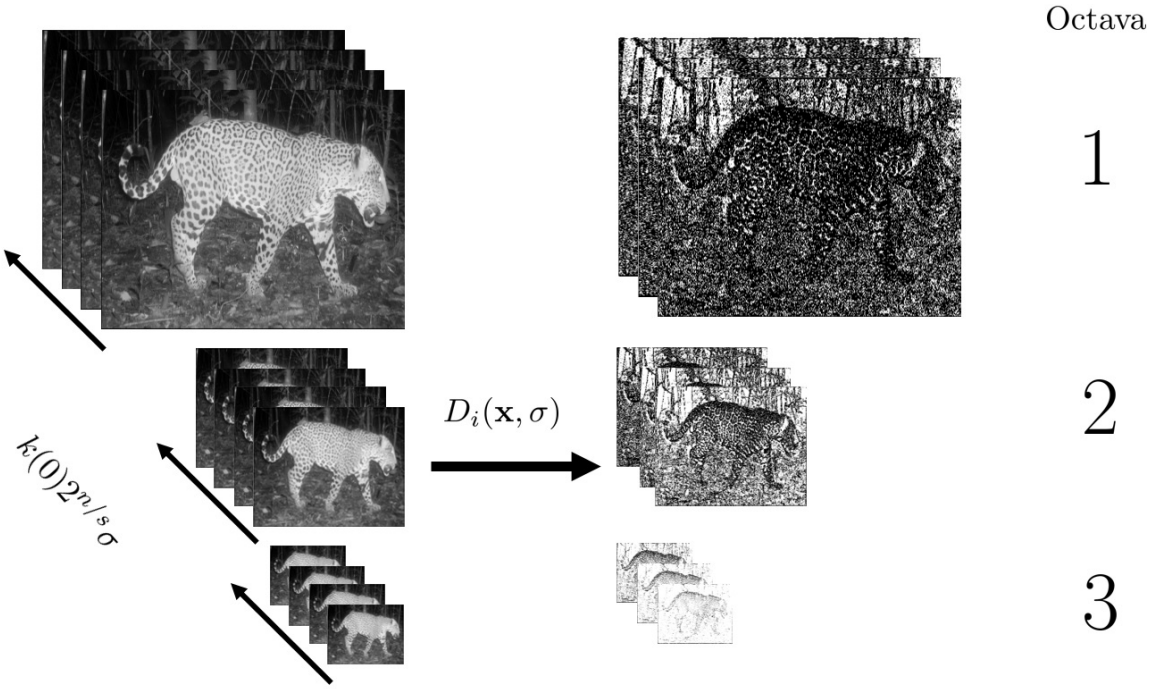


Figura 3.2: Búsqueda de los puntos estables mediante análisis piramidal. Los pixeles candidatos a ser puntos característicos son aquellos que son máximos o mínimos en las diferencias $D_i(\mathbf{x}, \sigma)$ de las convoluciones de las imágenes con Gaussianos. En cada octava, el valor de la desviación estándar de la convolución, $k(n)$, cambia de acuerdo a la expresión $k(0)2^{n/s}\sigma$, para un valor inicial $k(0)$. Las figuras muestran las convoluciones con la Gaussiana y luego las diferencias entre ellas.

origen en imágenes de más alta resolución, el único *kernel* que permite estabilidad a través de cambios en espacio y escala es la función Gaussiana. Esto permite justificar la representación de una imagen por medio de la operación

$$L(\mathbf{x}, \sigma) = G(\mathbf{x}, \sigma) * I(\mathbf{x}), \tag{3.1}$$

donde $*$ representa la convolución en el punto $\mathbf{x} = (x, y)$ y

$$G(\mathbf{x}, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}, \tag{3.2}$$

es una Gaussiana centrada en el origen cuya desviación estándar está dada por σ. En [8], Lowe muestra que la diferencia entre imágenes en escalas cercanas

$$D_i(\mathbf{x}, \sigma) = L(\mathbf{x}, k(i+1)\sigma) - L(\mathbf{x}, k(i)\sigma), \tag{3.3}$$

podría proporcionar localización de características que permanecen estables. En (3.3), $k(n) = k(0)2^{n/s}$, y s es el número de intervalos por octava (cada vez que σ se incrementa al doble). Por ejemplo, si $s = 4$, entonces $k(j)$, para $j = 0, \ldots, s$, con $k(0) = 1$, corresponde a la serie $\{1, 2^{1/4}, \ldots, 2^{j/4}, \ldots\}$.

Los extremos locales de $D(\mathbf{x}, \sigma)$ se encuentran al comparar cada pixel con sus 8 vecinos espaciales, y sus 18 vecinos en escala, 9 en la escala superior y 9 en la escala inferior. El punto es seleccionado como estable si es mayor o menor que todos sus vecinos. Los experimentos de Lowe[10] indican que la mejor frecuencia de muestreo es $s = 3$ por octava. Por otro lado, sus experimentos confirman que valores más grandes de σ aumentan la repetitividad de la localización de características. Sin embargo, con la finalidad de reducir el cálculo numérico, Lowe usa el valor de $\sigma = 1.6$. La imagen base del procesamiento piramidal se obtiene incrementando al doble la resolución de la imagen usando interpolación lineal y suavizando con un filtro Gaussiano con $\sigma_0 = 1$.

3.1.2. Localización Precisa de Características

La posición proporcionada mediante el procedimiento descrito en la sección anterior puede ser mejorada, pues el procesamiento a diferentes escalas puede generar imprecisiones en su localización. Para ello, la función de diferencias, $D(\mathbf{x})$, se aproxima con los primeros términos de la serie de Taylor[1] tal que

$$D(\mathbf{x}) = D_0 + D_{\mathbf{x}}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T D_{\mathbf{x}^2} \mathbf{x}, \tag{3.4}$$

donde $D_0 = D(0)$ es la evaluación en el lugar de la detección, $D_{\mathbf{x}} = \dfrac{\partial D(\mathbf{x})}{\partial \mathbf{x}}$ es el gradiente de D en el origen [2], $D_{\mathbf{x}^2} = \dfrac{\partial^2 D(0)}{\partial \mathbf{x}^2}$ es el Hessiano evaluado en el origen[3], y $\mathbf{x} = (x, y, \sigma)^T$. El punto extremo de $\hat{\mathbf{x}}$ se encuentra derivando (3.4) e igualando a cero. Es decir

$$\frac{\partial D(\mathbf{x})}{\partial \mathbf{x}} = D_{\mathbf{x}} + D_{\mathbf{x}^2} \mathbf{x} = 0. \tag{3.5}$$

[1] La expansión en serie de Taylor de una función $f(x)$ es igual a

$$f(x) = \sum_{n=0}^{\infty} f^{(n)}(a)(x-a)^n/n! = f(a) + f'(a)(x-a) + f''(a)(x-a)^2/2! + \cdots + f^{(n)}(a)(x-a)^n/n! + \ldots$$

[2] El gradiente de $D(\mathbf{x})$ con respecto a $\mathbf{x}^T = (x, y, \sigma)$ se define como

$$\frac{\partial D(\mathbf{x})}{\partial \mathbf{x}} = \left(\frac{\partial D(\mathbf{x})}{\partial x}, \frac{\partial D(\mathbf{x})}{\partial y}, \frac{\partial D(\mathbf{x})}{\partial \sigma} \right)^T$$

[3] El Hessiano de $D(\mathbf{x})$ con respecto a $\mathbf{x} = (x, y, \sigma)$ se define como

$$\frac{\partial^2 D(\mathbf{x})}{\partial \mathbf{x}^2} = \begin{pmatrix} \dfrac{\partial^2 D(\mathbf{x})}{\partial x^2} & \dfrac{\partial^2 D(\mathbf{x})}{\partial y \partial x} & \dfrac{\partial^2 D(\mathbf{x})}{\partial \sigma \partial x} \\ \dfrac{\partial^2 D(\mathbf{x})}{\partial x \partial y} & \dfrac{\partial^2 D(\mathbf{x})}{\partial y^2} & \dfrac{\partial^2 D(\mathbf{x})}{\partial \sigma \partial y} \\ \dfrac{\partial^2 D(\mathbf{x})}{\partial x \partial \sigma} & \dfrac{\partial^2 D(\mathbf{x})}{\partial y \partial \sigma} & \dfrac{\partial^2 D(\mathbf{x})}{\partial \sigma^2} \end{pmatrix},$$

donde se aprecia que el Hessiano es igual a su transpuesta.

Despejando términos conteniendo la incógnita, $\mathbf{x}$, de que contienen el gradiente y el Hessiano, se obtiene

$$D_{\mathbf{x}} = -D_{\mathbf{x}^2}\mathbf{x}. \tag{3.6}$$

Resolviendo para $\mathbf{x}$ se obtiene la expresión

$$\hat{\mathbf{x}} = -D_{\mathbf{x}^2}^{-1} D_{\mathbf{x}}. \tag{3.7}$$

Lowe utiliza el valor de $D(\hat{\mathbf{x}})$ para eliminar puntos con contraste bajo. Este valor se encuentra reemplazando (3.7) en (3.5). El resultado es

$$D(\hat{\mathbf{x}}) = D_0 - D_{\mathbf{x}}^T D_{\mathbf{x}^2}^{-1} D_{\mathbf{x}} + \frac{1}{2}\left(D_{\mathbf{x}^2}^{-1} D_{\mathbf{x}}\right)^T D_{\mathbf{x}^2} D_{\mathbf{x}^2}^{-1} D_{\mathbf{x}}, \tag{3.8}$$

que puede ser simplificado en

$$D(\hat{\mathbf{x}}) = D_0 - D_{\mathbf{x}}^T D_{\mathbf{x}^2}^{-1} D_{\mathbf{x}} + \frac{1}{2} D_{\mathbf{x}}^T D_{\mathbf{x}^2}^{-T} D_{\mathbf{x}}, \tag{3.9}$$

cuya suma de los dos últimos términos resulta en

$$D(\hat{\mathbf{x}}) = D_0 - \frac{1}{2} D_{\mathbf{x}}^T D_{\mathbf{x}^2}^{-T} D_{\mathbf{x}} = D_0 - \frac{1}{2} D_{\mathbf{x}}^T \hat{\mathbf{x}}. \tag{3.10}$$

Y Lowe elimina todos aquellos puntos para los cuales $D(\mathbf{x})$ es menor que 0.03.

Finalmente, Lowe propone eliminar aquellas características que se encuentran definidas a lo largo de contornos, evitando descripciones locales donde sea altamente probable que se tenga el problema de la apertura y, por tanto, tengan dificultades para estar localizados espacialmente. Con este fin, utiliza la matriz del tensor estructural definida por

$$H = \begin{pmatrix} \dfrac{\partial^2 D(\mathbf{x})}{\partial x^2} & \dfrac{\partial^2 D(\mathbf{x})}{\partial y \partial x} \\ \\ \dfrac{\partial^2 D(\mathbf{x})}{\partial x \partial y} & \dfrac{\partial^2 D(\mathbf{x})}{\partial y^2} \end{pmatrix}. \tag{3.11}$$

Lowe evita el cálculo de los eigenvalores usando el esquema propuesto por Harris y Stephen[6] donde se hace notar la equivalencia con la traza y el determinante, tal que

$$\begin{aligned} \texttt{trace}(H) &= \frac{\partial^2 D(\mathbf{x})}{\partial x^2} + \frac{\partial^2 D(\mathbf{x})}{\partial y^2} = \alpha + \beta, \\ \texttt{det}(H) &= \frac{\partial^2 D(\mathbf{x})}{\partial x^2}\frac{\partial^2 D(\mathbf{x})}{\partial y^2} + \left(\frac{\partial^2 D(\mathbf{x})}{\partial x \partial y}\right)^2 = \alpha\beta, \end{aligned} \tag{3.12}$$

donde α y β son los eigenvalores. Si el determinante es negativo el punto es inmediatamente descartado. Extendiendo estos criterios, Lowe propone analizar el cociente r entre el eigenvalor mayor α y el eigenvalor menor β. En lugar de calcular los eigenvalores, Lowe utiliza el cociente entre la traza y el determinante. Esto resulta, haciendo $\alpha = r\beta$, en la expresión

$$\frac{\texttt{trace}(H)^2}{\texttt{det}(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}. \tag{3.13}$$

La cual es conveniente de evaluar pues no incluye el cálculo de eigenvalores. Lowe propone usar un valor de $r = 10$ para eliminar puntos que se encuentran en contornos.

3.1.3. Determinación de Orientación

La orientación del descriptor SIFT se basa en el análisis de los niveles de intensidad en la vecindad del punto que fue seleccionado como característica. Para ello se evalúa la vecindad para obtener la orientación y magnitud del gradiente para la imagen correspondiente a la escala donde se encontró al descriptor. Es decir, sea la magnitud, $m(\mathbf{x})$, y dirección, $\theta(\mathbf{x})$, definidas como

$$m(\mathbf{x}) = \sqrt{L_x^2 + L_y^2}, \text{ y } \theta(\mathbf{x}) = \arctan\left(L_y, L_x\right), \tag{3.14}$$

donde $\mathbf{x} = (x, y)$ y $L_x = L(x+1, y) - L(x-1, y)$, y $L_y = L(x, y+1) - L(x, y-1)$ y la función arctan maneja el caso cuando $L_x = 0$. Luego se construye un histograma de la orientación usando 36 divisiones uniformes. Cada contribución al histograma es ponderada por el respectivo valor de la magnitud en esa posición. Como resultado se crea una característica para el pico más alto en el histograma, así como para cualquier otro pico local que sea mayor al 80 % del pico más alto. Finalmente, la orientación es definida ajustando una parábola con los tres valores de histograma más cercanos al pico y escogiendo el máximo de la parábola.

3.2. Descriptor Local

El descriptor SIFT se basa en la sumarización de las orientaciones del gradiente sobre pequeñas regiones de la imagen. La orientación de la característica es la obtenida anteriormente, así que todos los vectores del gradiente son expresados relativos a esta dirección. Supóngase que se tiene la imagen de 16× 16 ilustrada en la Figura 3.3(a). Sobrepuesto a esa imagen se ha colocado la ilustración de los vectores del gradiente. La magnitud del gradiente pesa la contribución de cada orientación. Para ello se usa una Gaussiana cuya dispersión σ es igual a la mitad de la anchura de la ventana de análisis. En la Figura 3.3(b) se ilustra la magnitud y en 3.3(c) la Gaussiana usada. Para la obtención de los datos se usa la escala donde se observó la característica. La región de 16 × 16 se divide en 4 × 4 celdas, cada celda conteniendo 4 × 4 pixeles. Con base en la orientación de los pixeles, se construyen en las vecindades histogramas utilizando ocho direcciones. Cada elemento en el histograma es pesado por la magnitud del gradiente. La Figura 3.3(d) muestra los histogramas ponderados por la magnitud del gradiente. El descriptor se forma construyendo un vector con los valores de todos los histogramas de los 4 × 4 bloques de 8 orientaciones, formando un vector de $4 \times 4 \times 8 = 128$ elementos. El vector así obtenido es normalizado por su magnitud, haciendo al descriptor invariante a cambios de iluminación donde cada entrada sea o multiplicada por un factor o se le sume una constante. Para otros efectos no lineales, no se permite que el valor máximo de las entradas de este vector sean mayores a 0.2, tras lo cual el vector es normalizado.

3.3. Aplicación: Identificación Individual de Jaguares

Las características SIFT pueden ser utilizadas como base en una variedad de aplicaciones que incluye el flujo óptico, la reconstrucción tridimensional, o la identificación de objetos. En particular, para esta última aplicación, una posible estrategia podría ser la sugerida por

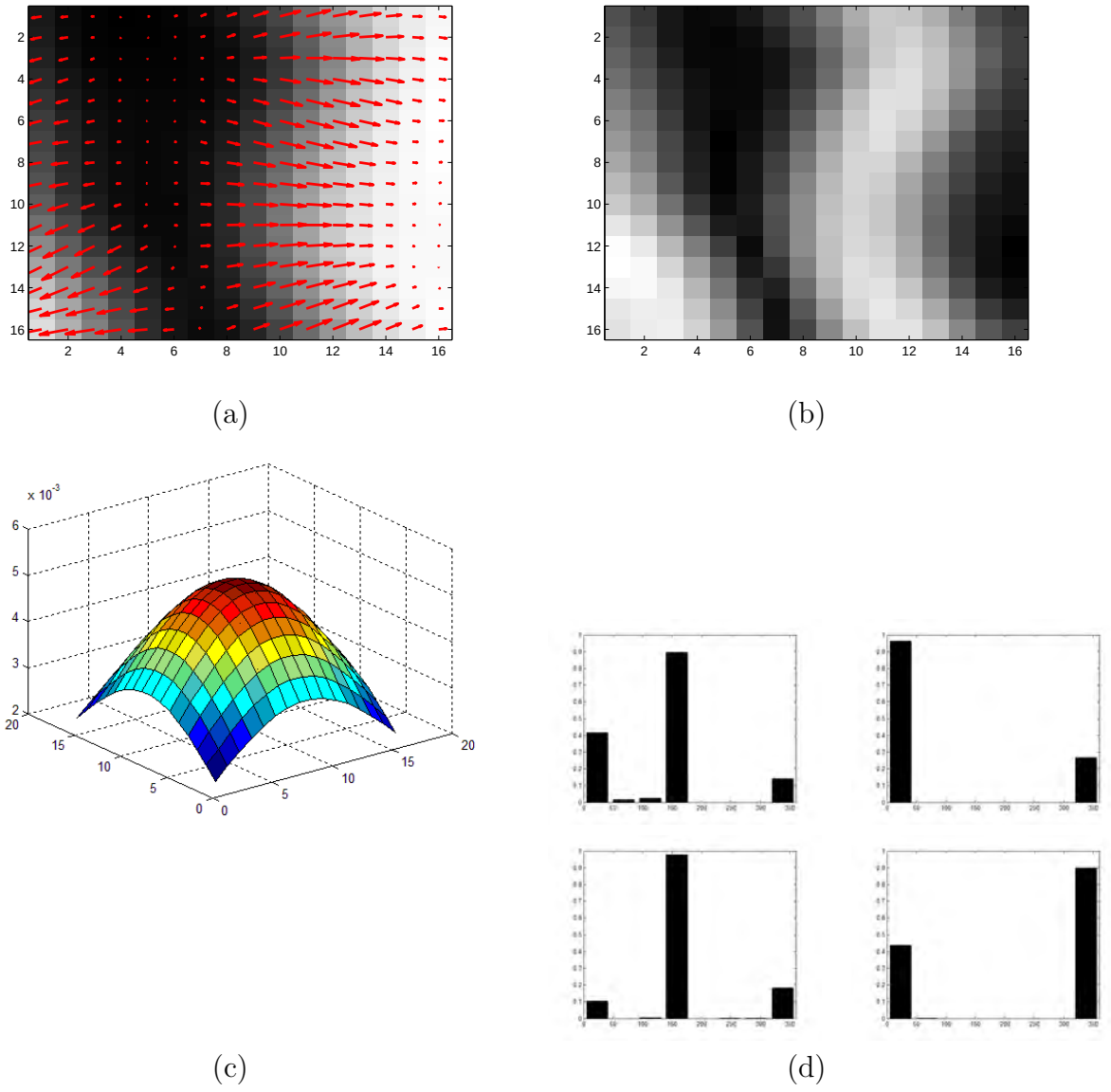


Figura 3.3: Construcción del descriptor SIFT. Pequeñas porciones de 16×16 son consideradas para el análisis. El gradiente(a) es utilizado para construir histogramas(d), sobre cuatro ventanas no traslapadas de 4×4, pesados por la magnitud gradiente(b) ponderado a su vez por una Gaussiana(c) cuya dispersión σ es la mitad de la anchura de la ventana.

Llamada : $\langle \mathbf{X}, \Theta, \mathbf{H} \rangle \leftarrow$ `SIFT` $(\mathbf{I})$
Entradas: Una imagen de intensidad $\mathbf{I}$
Salidas : El conjunto de posiciones $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ donde se encuentran las características SIFT, los descriptores $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_n\}$, las orientaciones dominantes $\Theta = \{\theta_1, \ldots, \theta_n\}$

```
// Definir el número de octavas
O ← ...;
// Incrementar el tamaño de la imagen al doble
J ← Incrementar_Tamaño_al_Doble (I);
// Inicializa las diferencias
D ← {};
// Para cada octava
for o ← 1 : O do
    // La imagen para esta octava se convoluciona con un conjunto de
       Gaussianas, cada una con diferente dispersión σ
    G ←Convolucionar_con_Gaussianas (J, o);
    // Calcular las diferencias de las Gaussianas para esta octava
    D ← D ⋃ Diferencias_de_Gaussianas (G);
end
// Buscar los puntos extremos para cada octava
X' ← Puntos_Extremos (D);
// Crear puntos característicos en la orientación dominante del
   punto extremo
⟨X_{2×n}, Θ_{1×n}⟩ ← Crear_Punto_Característico (X');
// Obtener descripción para cada punto característico
for i ← 1 : n do
    // Rotar la región de acuerdo a la orientación y obtener la
       información del gradiente de I, I_x y I_y
    ⟨I_x, I_y⟩ ← Rotar_Región (I, x_i, θ_i);
    // Obtener descriptor en la forma de histogramas de orientaciones
       pesadas por la magnitud del gradiente
    h_i ← Obtener_Descriptor (I_x, I_y);
end
```

Algorithm 5: Algoritmo para obtener las características SIFT a partir de una imagen $\mathbf{I}$. Las características SIFT se obtienen de regiones de la imagen que presentan valores extremos en las diferencias de imágenes convolucionadas con Gaussianas.

Cuadro 3.1: Número de fotografías por animal. Nuestra base de datos contiene 64 fotos. El número acumulado de las funciones depende del tamaño del jaguar en el conjunto de fotos en las que aparece.

Individuo	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Fotos	8	7	3	3	4	4	1	4	6	7	3	3	1	5
SIFT	5,916	12,696	3,857	8,165	13,119	6,931	1,832	9,998	6,837	11,746	5,149	4,497	514	12,174

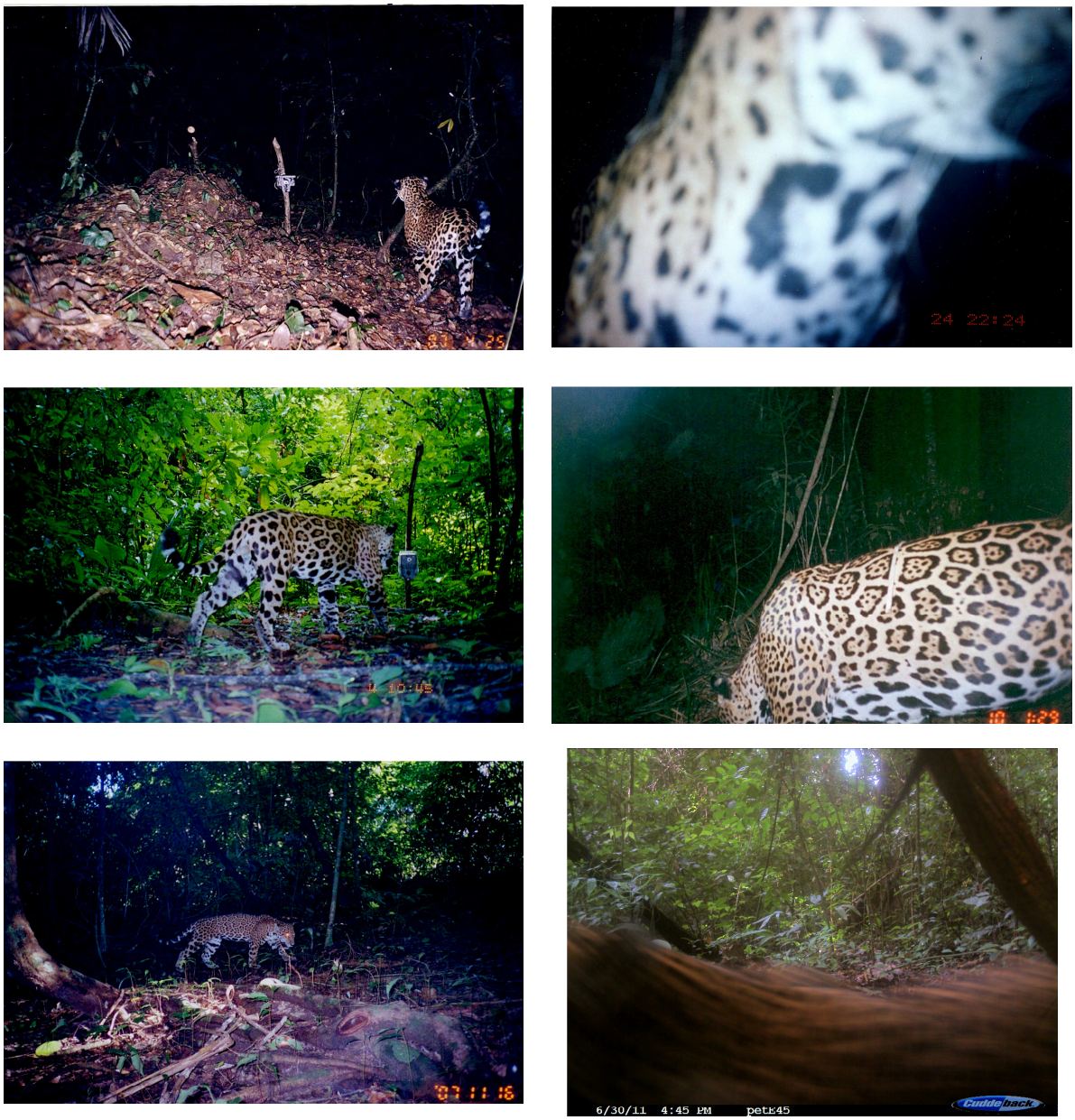


Figura 3.4: Algunos ejemplos de fotos de jaguar. La iluminación, el enfoque, la resolución, postura y oclusión son factores que varían en la base de imágenes (Gracias a Antonio de la Torre Lara por permitirme el uso de estas imágenes).

Figura 3.5: Segmentación. En (a) se ilustra la segmentación basada en *GrabCut*[11]. Por lo general, el cuadro de límite y unas cuantas pinceladas bastan para proporcionar información suficiente para obtener un buen resultado de la segmentación. En (b) se obtiene la máscara binaria donde se encuentra el objeto de interés. Después de la aplicación de *GrabCut* necesitamos llenar agujeros y aplicar un filtro de sal y pimienta para eliminar el ruido.

Lowe[10] y la cual pudiera ser sumarizada como sigue. Primero, cada punto característico es apareado a la base de datos de características extraídas de imágenes de entrenamiento. Luego, grupos de al menos tres correspondencias son detectadas. Finalmente, se ajusta el modelo con las características y el resultado del ajuste es usado para aceptar o rechazar la interpretación. Para poder realizar el apareamiento entre características, la mejor correspondencia para cada punto característico puede ser encontrada usando la mínima distancia Euclidiana. Para descartar correspondencias incorrectas se evalúa la segunda mejor correspondencia. Si la distancia entre la mejor correspondencia y la segunda mejor correspondencia es pequeña, entonces el punto se descarta pues se considera que esta correspondencia es ambigua. Es importante que la segunda mejor correspondencia provenga de imágenes que efectivamente provengan de diferentes objetos.

Se ha mostrado que para más de 10 dimensiones no hay mejor algoritmo de búsqueda de correspondencia que la exhaustiva[5]. Así que Lowe usa el algoritmo de *Best-Bin-First* (BBF)[1], el cual regresa el vecino más próximo con una probabilidad muy alta. El BBF usa un ordenamiento similar a los árboles $k-d$[4] tal que los bins en el espacio de características se buscan en el orden de su distancia más corta a la localidad de búsqueda, más que el bin más cercano en el árbol. Lowe detiene la búsqueda después de analizar los 200 candidatos más cercanos, lo cual representa una diferencia considerable cuando se tienen bases de datos con muchos posibles candidatos. Para asignar una correspondencia sólo se consideran candidatos donde el vecino más próximo está a menos de 0.8 veces la distancia al segundo vecino más cercano, eliminando los casos más difíciles donde muchos vecinos están muy cerca.

[4]Un árbol $k-d$ es un árbol binario que tiene en cada nodo la representación de un punto en k dimensiones. Cada nodo usa una de las dimensiones para dividir el espacio en dos mitades. Una mitad corresponde a la rama izquierda y la otra mitad a la rama derecha. En cada nivel del árbol el mismo eje es usado para hacer las divisiones. Una vez completadas todas las coordenadas, se vuelve a repetir el orden de ellas en los subsiguientes niveles del árbol.

Lowe encontró que podía hacer reconocimiento de objetos efectivo únicamente con tres características en correspondencia. Para ello, Lowe usa la transformada de Hough[2], la cual identifica grupos de características que votan por objetos que son consistentes con ellas. En casi todas las implementaciones de la transformada de Hough se usa un arreglo con tantas dimensiones como parámetros. Lowe usa un arreglo unidimensional indexamiento que mapea características con identidades. Algunas de las correspondencias son eliminadas mediante una verificación geométrica en la que se calcula la transformación afín entre el modelo y las características encontradas. Una transformación afín puede expresarse como

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}, \tag{3.15}$$

donde $(t_x, t_y)^T$ corresponde a la traslación, y la rotación, escalamiento y estiramiento están representados por los valores m_{ij}. Realizando las multiplicaciones correspondientes, agrupando las incógnitas y resolviendo, se tiene la expresión

$$\begin{pmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \cdots & & & \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{21} \\ m_{22} \\ t_x \\ t_y \end{pmatrix} = \begin{pmatrix} u \\ v \\ \vdots \end{pmatrix}, \text{ ó} \tag{3.16}$$

$$\mathbf{Ax} = \mathbf{b},$$

con solución de mínimos cuadrados expresada por

$$\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}. \tag{3.17}$$

La proyección mediante la transformada afín sirve para eliminar correspondencias cuando éstas no están dentro de cierto rango de error. Si menos de tres puntos sobreviven a este filtraje, la correspondencia es descartada. Una vez rechazados puntos, la transformación afín es recalculada y el proceso repetido.

La decisión final de aceptación o rechazo se basa en un modelo probabilístico. En este método se calcula el número esperado de correspondencias erróneas al modelo, dado el tamaño proyectado del modelo, el número de características dentro de la región, y la exactitud del ajuste (ver Lowe [9]). Luego se evalúa la probabilidad de que el objeto esté presente dado el número de correspondencias encontradas. Para Lowe, el modelo se acepta cuando hay más de un 0.98 de probabilidad de una interpretación correcta.

3.4. Implementación

En esta sección presentamos un ejemplo de identificación de animales basado en el uso de características locales SIFT. En el problema tenemos un conjunto de jaguares donde se pudo haber obtenido varias imágenes de cada uno de ellos (ver la Tabla 3.1 y los ejemplos

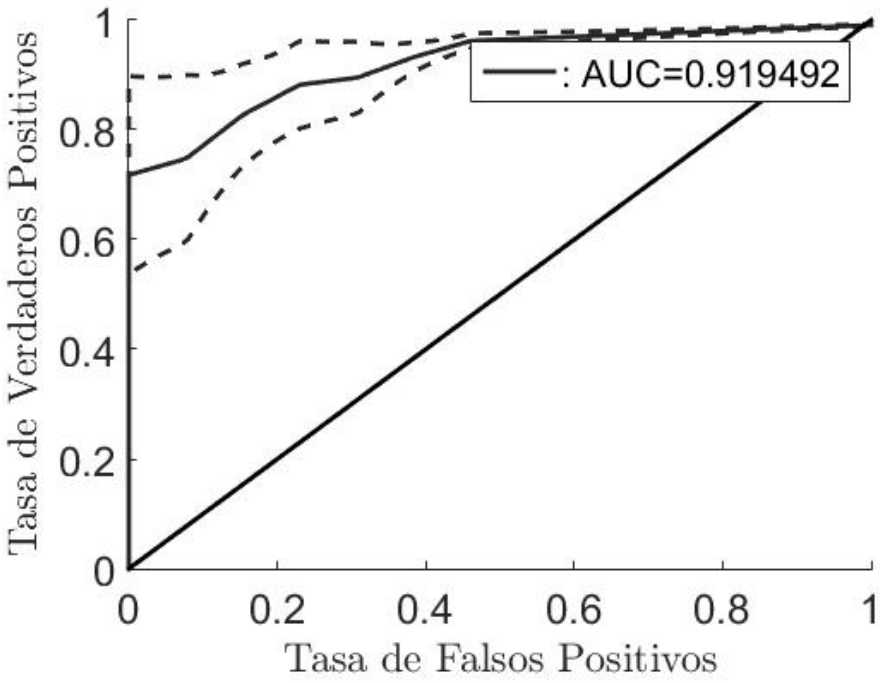


Figura 3.6: Análisis ROC. Una curva ROC ilustra el nivel de desempeõ de un algoritmo de clasificación. Una medida que sumariza la gráfica es el área bajo la curva (AUC). Medidas sobre la diagonal indican un comportamiento arriba de una decisión azarosa.

de imágenes en la Figura 3.4). El problema consiste en determinar el individuo al que se le tomó una imagen que se presenta.

Este ejemplo ilustra una aproximación en la cual la solución proviene tanto del proceso automático como de la intervención humana. Por ejemplo, las imágenes son segmentadas en forma semiautomática mediante el algoritmo de *GrabCut* [11]. Con esta herramienta, el usuario define un cuadro delimitador del objeto de interés en la imagen. Entonces, el algoritmo *GrabCut* construye una mezcla de Gaussianas de la distribución del color tanto para los pixeles al interior del cuadro límite como para los colocados al exterior. Enseguida, un pixel dentro del cuadro delimitador se clasifica en primer o segundo plano en función de la distribución que le representa mejor. Por lo general, este proceso es muy cercano al resultado deseado. Sin embargo, uno tiene la posibilidad de ofrecer información sobre el objeto o el fondo mediante algunas pinceladas en las regiones que fueron mal clasificadas. Por lo general, unas pocas iteraciones de este proceso son suficientes para lograr muy buenos resultados de segmentación. La Figura 3.5 (a) ilustra el proceso mediante una interfaz con el programa *GrabCut*. El resultado de *GrabCut* se procesa para llenar agujeros y eliminar el ruido de sal y pimienta. La Figura 3.5(b) muestra los resultados de la segmentación binaria que separa entre objeto y fondo.

En [3] se hace la caracterización de animales usando descriptores locales SIFT[13]. Dado que para la mayoría de los animales hay varias imágenes del mismo individuo, la idea es extraer características SIFT de la parte de la imagen donde se encuentra el animal. Luego, para cada individuo, se agregan tantas características como fuera posible obtener en una

especie de enfoque de la bolsa-de-características. Luego, durante la identificación, se calculan las características de la imagen y se comparan con las características correspondientes a un individuo en particular. Huelga decir que, cuando se compara la imagen de un individuo con la bolsa-de-características[4] que le corresponden, no se consideran las que vienen de la misma imagen que se compara. Este proceso resulta en algunas correspondencias de las características y una puntuación que es igual a la distancia Euclidiana entre los vectores SIFT. Aquí se asume que la puntuación global de un individuo es la distancia media entre correspondencias. Esto resulta en un ordenamiento en el que tenemos tanto la puntuación total y la información coincidente, que sabemos verdadera. Hemos encontrado que este proceso proporciona resultados bastante buenos. Para notarlo, ilustramos el análisis ROC (*Receiver Operating Characteristic*)[12]. En este análisis (ver Figura 3.6) graficamos la razón de falsos positivos en comparación con la tasa de verdaderos positivos. Una forma estándar de expresar este resultado es mediante la medición del área bajo la curva (AUC), que para el experimento realizado corresponde a un promedio de 0.93.

Sumario

Las características SIFT son una herramienta muy poderosa para la descripción local de las imágenes, pues son invariantes a cambio de escala, rotaciones, traslaciones, y parcialmente invariantes a transformaciones afines y cambios de iluminación. Una amplia evidencia experimental ha permitido la aplicación de parámetros muy robustos para una amplia gama de situaciones. Las características SIFT son localizadas como extremos en restas de convoluciones de Gaussianos de las imágenes. Una localización más precisa se logra mediante el análisis local del gradiente. El descriptor corresponde a un vector de 128 posiciones, donde se expresan las variaciones locales de orientación y magnitud del gradiente.

Las características SIFT tienen una gran aplicación en Visión por Computadora; por ejemplo, en el reconocimiento de objetos. Existen muchas implementaciones del algoritmo de Lowe [10]. Aquí utilizamos el `VlFeat` para presentar una aproximación de identificación individual de jaguares.

Ejercicios

1. Implementa el extractor de características SIFT basado en el Algoritmo 5.

2. Lowe asigna diversos valores a los parámetros del algoritmo de extracción de características. Por ejemplo:

   - Número de intervalos por octava, $s = 4$,
   - Valor de dispersión base, $\sigma = 1.6$,
   - Eliminación de aquellos candidatos para los cuales $D(\mathbf{x}) < 0.03$,
   - Eliminación de puntos sobre contornos, $r = 10$,
   - Umbral para la selección de orientaciones dominantes secundarias, 80 %,

- Valor mayor en el histograma normalizado no debe ser mayor a 0.2.

Modifica estos umbrales en tu implementación del algoritmo y comenta sobre los resultados.

Bibliografía

[1] Beis, Jeffrey y David Lowe: *Shape Indexing using Approximate Nearest-Neighbour Search in High-Dimensional Spaces*. En *IEEE Conference on Computer Vision and Pattern Recognition*, páginas 1000–1006, 1997.

[2] Duda, Richard y Peter Hart: *Use of the Hough Transformation to Detect Lines and Curves in Pictures*. Communications of the ACM, 15(1):11–15, 1972.

[3] Duyck, James, Chelsea Finn, Andy Hutcheon, Pablo Vera, Joaquin Salas y Sai Ravela: *Sloop: A Pattern Retrieval Engine for Individual Animal Identification*. Pattern Recognition, 48(4):1059–1073, 2015.

[4] Fei-Fei, Li y Pietro Perona: *A Bayesian Hierarchical Model for Learning Natural Scene Categories*. En *IEEE Conference on Computer Vision and Pattern Recognition*, volumen 2, páginas 524–531, 2005.

[5] Friedman, Jerome, Jon Bentley y Raphael Finkel: *An Algorithm for Finding Best Matches in Logarithmic Expected Time*. ACM Transactions on Mathematical Software, 3(3):209–226, 1977.

[6] Harris, Chris y Mike Stephens: *A Combined Corner and Edge Detector*. En *Alvey Vision Conference*, volumen 15, página 50, 1988.

[7] Koenderink, Jan: *The Structure of Images*. Biological Cybernetics, 50(5):363–370, 1984.

[8] Lowe, David: *Object Recognition from Local Scale-Invariant Features*. En *IEEE International Conference on Computer Vision*, volumen 2, páginas 1150–1157, 1999.

[9] Lowe, David: *Local Feature View Clustering for 3D Object Recognition*. En *IEEE Conference on Computer Vision and Pattern Recognition*, volumen 1, páginas I–682, 2001.

[10] Lowe, David: *Distinctive Image Features from Scale-Invariant Keypoints*. International Journal of Computer Vision, 60(2):91–110, 2004.

[11] Rother, Carsten, Vladimir Kolmogorov y Andrew Blake: *Grabcut: Interactive Foreground Extraction using Iterated Graph Cuts*. ACM Transactions on Graphics, 23(3):309–314, 2004.

[12] Swets, John, Robyn Dawes y John Monahan: *Better Decisions through Science*. Scientific American, página 83, 2000.

[13] Vedaldi, A. y B. Fulkerson: *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. `http://www.vlfeat.org/`, 2008.

Parte II

Reconstrucción Tridimensional

CAPÍTULO 4

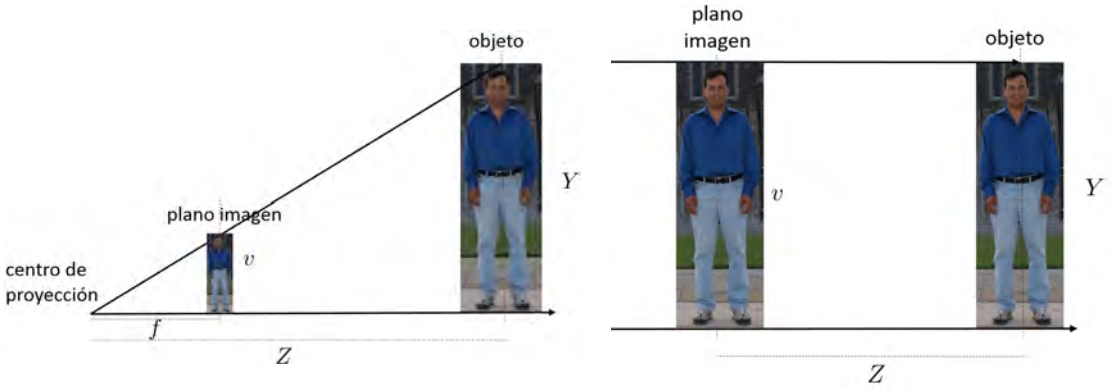
Estructura bajo Proyección Ortográfica

El problema de la estructura a partir del movimiento consiste en extraer las formas tridimensionales de los objetos y el movimiento de la cámara. En este documento estudiamos la solución a este problema mediante una familia de técnicas conocidas como *factorización*, en la cual la secuencia de imágenes es transformada en sus componentes de movimiento y forma. El movimiento se refiere a los parámetros de rotación y traslación del sistema de referencia de la cámara relativa al objeto. La forma se refiere a las características geométricas tridimensionales de los objetos. Tomasi y Kanade[11] presentaron una solución a este problema para el caso particular de la proyección ortográfica (ver Figura 4.1). En este capítulo se asumirá que los objetos sobre los cuales se trabaja permanecen estáticos y que la reconstrucción se expresa en un sistema de referencia fijo centrado en el objeto.

4.1. Formulación de la Solución

Sea $\mathbf{W}_{2F\times P}$ la matriz de medición que aglutina las observaciones realizadas sobre P puntos a lo largo de F imágenes. Los P puntos son aquellos para los cuales se ha encontrado correspondencia a lo largo de las F imágenes de la secuencia. Tomasi y Kanade[11] notaron que el rango, o número de hileras o columnas linealmente independientes, de esta matriz, es tres. Como consecuencia, la matriz $\mathbf{W}$ puede ser descompuesta en las matrices $\mathbf{R}_{2F\times 3}$, que representa la rotación de la cámara sobre cada lugar donde se toma una imagen, y $\mathbf{S}_{3\times P}$, que representa la forma tridimensional del objeto en un sistema coordenado centrado en él. En proyección ortográfica, los objetos permanecen del mismo tamaño, independientemente que tan alejados se encuentren. Por ello, los componentes de la traslación, en la dirección horizontal y vertical, pueden ser obtenidos promediando sobre las hileras de $\mathbf{W}$. Por supuesto, el componente de profundidad de la traslación no existe bajo proyección ortográfica.

Refraseando, dos de las características del método de Tomasi y Kanade es que asumen un sistema de referencia centrado en el objeto y proyección ortográfica. La primera condición robustece la solución numérica del sistema de ecuaciones en situaciones que podrían confundirse, *e.g.*, una rotación pequeña sobre el eje vertical y una traslación pequeña sobre el eje horizontal. La segunda condición también robustece la solución ante aspectos numéricos. Por ejemplo, supóngase que se quiere obtener la profundidad de un objeto por la diferencia de distancia de los elementos que lo conforman. Bajo proyección perspectiva se tiene gran sensibilidad a la posición de la imagen donde se estima que ocurre la proyección de los puntos del mundo. Esto es así porque en proyección perspectiva la profundidad está inversamente



(a) Proyección Perspectiva (b) Proyección Ortográfica

Figura 4.1: Proyecciones. En proyección perspectiva(a), el tamaño de la imagen, v, de un objeto cuyo tamaño real es Y está en relación inversa a su distancia Z a una cámara de longitud focal f. Por otro lado, en proyección ortográfica(b) el tamaño del objeto en la imagen es el mismo, independientemente de su distancia a la cámara, *i.e.*, $v = Y$.

relacionada con la distancia entre las proyecciones.

Esta sección se basa en el desarrollo que Tomasi y Kanade presentan en [11]. Dada un conjunto de trayectorias $\{(u_{fp}, v_{fp}) | f = 1, \ldots, F, p = 1, \ldots, P\}$ de P puntos sobre F imágenes, la matriz $\mathbf{W}$ de observaciones se forma con la combinación de las matrices $\mathbf{U}_{F\times P}$ y $\mathbf{V}_{F\times P}$ conteniendo respectivamente las coordenadas horizontales u_{fp} y verticales v_{fp} de las características sobresalientes de la imagen, tal que

$$\mathbf{W} = \begin{pmatrix} \mathbf{U} \\ \mathbf{V} \end{pmatrix}. \tag{4.1}$$

El desplazamiento horizontal y vertical de la cámara puede ser obtenido, bajo proyección ortográfica, al obtener el centroide de las características, tal que

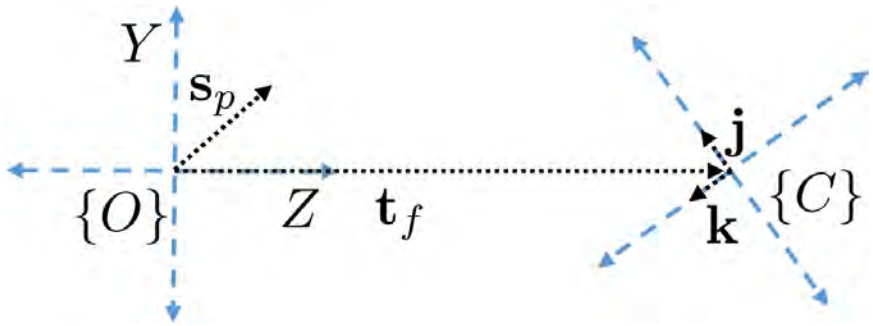
$$\overline{u}_f = \frac{1}{P}\sum_{p=1}^{P} u_{fp}, \text{ y } \overline{v}_f = \frac{1}{P}\sum_{p=1}^{P} v_{fp}. \tag{4.2}$$

Restando el centroide a cada coordenada de los puntos característicos, tal que $\tilde{u}_{fp} = u_{fp} - \overline{u}_f$ y $\tilde{v}_{fp} = v_{fp} - \overline{v}_f$, se obtienen las matrices $\tilde{\mathbf{U}} = [\tilde{u}_{fp}]$ y $\tilde{\mathbf{V}} = [\tilde{v}_{fp}]$ que contienen las matrices registradas de mediciones

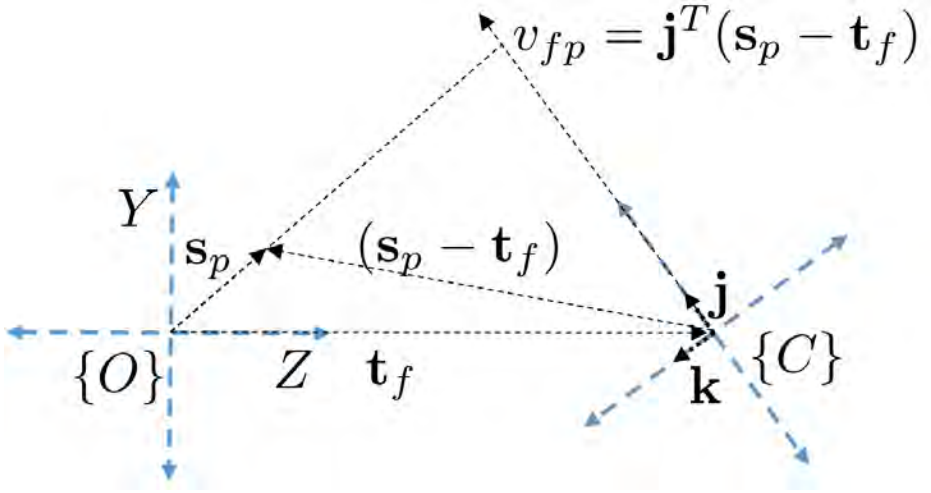
$$\tilde{\mathbf{W}} = \begin{pmatrix} \tilde{\mathbf{U}} \\ \tilde{\mathbf{V}} \end{pmatrix}. \tag{4.3}$$

La proyección ortográfica difiere de la más cotidiana proyección de perspectiva. En proyección ortográfica, las coordenadas horizontal y vertical de los objetos se traslada directamente a la imagen, obviando la información de profundidad. Esta simplificación aproxima casos donde los objetos se encuentran alejados de la cámara (o son pequeños) y pueden ser aproximados con el uso de una lente telecéntrica.

En la Figura 4.2 se pretende ilustrar el problema en forma simplificada, utilizando sólo una de las coordenadas de la proyección ortográfica. Ahí se observa que una de las coordenadas



(a) Sistemas de Referencia



(b) Proyección en el plano imagen

Figura 4.2: Estructura a partir del movimiento usando factorización. El sistema de referencia se centra en el objeto, en $\{O\}$, y la cámara tiene su sistema de referencia en $\{F\}$(a). El eje focal de la cámara se encuentra en la dirección de $\mathbf{k}$. Ambos $\{O\}$ y $\{F\}$ están separados por una traslación $\mathbf{t}_f$. Un punto del objeto se encuentra en la posición $\mathbf{s}_p$. El problema de la estructura a partir del movimiento consiste en obtener los valores de la estructura $\mathbf{s}_p$ y el movimiento $\mathbf{i}$ (no mostrado) y $\mathbf{j}$, con respecto a $\{O\}$, partiendo de los puntos de la imagen (u_{fp}, v_{fp}) seguidos a través de una secuencia de imágenes (b). La traslación horizontal y vertical se encuentran obteniendo los puntos del centroide del conjunto de características observadas. Debido a que se tiene proyección ortográfica, las dimensiones de los objetos en el mundo se preservan en la imagen. En el diagrama no se ilustra la tercera dimensión, X en $\{O\}$ e $\mathbf{i}$ en $\{F\}$.

de la proyección v_{fp} es producto de unos de los ejes de referencia de la cámara $\mathbf{j}$ y la diferencia entre la posición del punto en el mundo $\mathbf{s}_p$ y la posición de la cámara con respecto al objeto $\mathbf{t}_f$, tal que

$$v_{fp} = \mathbf{j}_f^T(\mathbf{s}_p - \mathbf{t}_f), \tag{4.4}$$

donde $\mathbf{s}_p$ es un punto del objeto, $\mathbf{t}_f$ es la posición del centro del sistema de referencia de la cámara $\{F\}$ con respecto al objeto $\{O\}$, y $\mathbf{i}$ y $\mathbf{j}$ son vectores unitarios, ortogonales entre si, con las orientaciones del plano imagen en el sistema de referencia $\{O\}$. Partiendo de la ecuación de las coordenadas centradas en el centroide se tiene que

$$\tilde{v}_{fp} = v_{fp} - \overline{v}_f. \tag{4.5}$$

Usando (4.2) y (4.4), (4.5) puede expresarse como

$$\tilde{v}_{fp} = \mathbf{j}_f^T(\mathbf{s}_p - \mathbf{t}_f) - \frac{1}{P}\sum_{q=1}^{P}\mathbf{j}_f^T(\mathbf{s}_q - \mathbf{t}_f), \tag{4.6}$$

donde el término $\mathbf{j}^T$ puede ser factorizado, y el término $\mathbf{t}_f$ eliminado, obteniendo

$$\tilde{v}_{fp} = \mathbf{j}_f^T\left(\mathbf{s}_p - \frac{1}{P}\sum_{q=1}^{P}\mathbf{s}_q\right). \tag{4.7}$$

Dado que por construcción $\dfrac{1}{P}\displaystyle\sum_{q=1}^{P}\mathbf{s}_q = 0$, (4.7) se reduce a

$$\tilde{v}_{fp} = \mathbf{j}_f^T\mathbf{s}_p. \tag{4.8}$$

Un desarrollo similar se puede realizar para mostrar que $\tilde{u}_{fp} = \mathbf{i}_f^T\mathbf{s}_p$. En suma, la matriz de mediciones centralizadas se puede expresar como

$$\tilde{\mathbf{W}} = \mathbf{RS}, \tag{4.9}$$

donde $\mathbf{R}_{2F\times 3}$ es la matriz que corresponde al movimiento y está dada por

$$\mathbf{R} = \begin{pmatrix} \mathbf{i}_1 & \dots & \mathbf{i}_F & \mathbf{j}_1 & \dots & \mathbf{j}_F \end{pmatrix}^T, \tag{4.10}$$

y $\mathbf{S}_{3\times P}$ es la matriz que corresponde a la forma del objeto, y está dada por

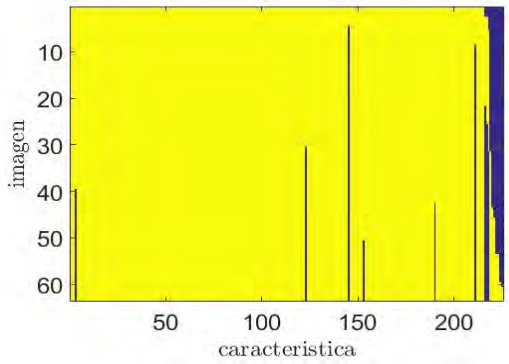
$$\mathbf{S} = \begin{pmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_P \end{pmatrix}. \tag{4.11}$$

4.2. Método de Factorización

La matriz $\mathbf{W}$ tiene rango tres cuando se encuentra libre de error. En casos prácticos hay que obtener el valor óptimo. Un criterio que puede ser seleccionado está dado por

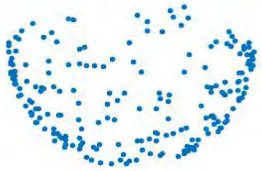
$$e = \arg\min_{R,S} \|\mathbf{W} - \mathbf{RS}\|_F^2, \tag{4.12}$$

(a) Imagen



(b) Matriz de llenado

(c) Trayectorias



(d) Reconstrucción 3D

(e) Movimiento de la cámara

Figura 4.3: Implementación del algoritmo de estructura y movimiento a partir de una secuencia de imágenes. En (a) se presenta una imagen de la secuencia. El seguimiento de las características resulta en las trayectorias descritas en (c) y correspondiente matriz de llenado en (b). La estructura resutante se muestra en (d). La trayectoria seguida por la cámara se muestra en (e).

donde $\|\cdot\|_F$ es la norma de Frobenius [1]. Una manera de resolver este problema de optimización es mediante la descomposición en valores singulares[2] de $\mathbf{W}$, tal que

$$\mathbf{W} = \mathcal{U}\mathbf{\Sigma}\mathcal{V}^T. \tag{4.13}$$

Dado que $\mathbf{W}$ contiene ruido, ya que sabemos que es rango tres, estamos interesados en las primeras tres columnas de $\mathcal{U}$, la matriz de 3×3 superior izquierda de $\mathbf{\Sigma}$, y las primeras tres hileras de $\mathcal{V}$, como sigue:

$$\mathcal{U} = \left(\mathcal{U}'_{2F\times 3} | \mathcal{U}''_{2F\times(P-3)}\right), \mathbf{\Sigma} = \left(\begin{array}{c|c} \mathbf{\Sigma}'_{3\times 3} & 0 \\ \hline 0 & \mathbf{\Sigma}''_{(P-3)\times(P-3)} \end{array}\right), \mathcal{V} = \left(\begin{array}{c} \mathcal{V}'_{3\times P} \\ \hline \mathcal{V}''_{(P-3)\times P} \end{array}\right). \tag{4.14}$$

Así, la mejor aproximación a la matriz de medición es

$$\hat{\mathbf{W}} = \mathcal{U}'\mathbf{\Sigma}'\mathcal{V}'^T. \tag{4.15}$$

Supóngase que se llega a una solución, arbitraria, para $\mathbf{R}$ y $\mathbf{S}$ dada por la descomposición

$$\hat{\mathbf{R}} = \mathcal{U}' \text{ y } \hat{\mathbf{S}} = \mathbf{\Sigma}'\mathcal{V}'^T. \tag{4.16}$$

Sin embargo, esta solución no es única, pues cualquier matriz invertible $\mathbf{Q}_{3\times 3}$ satisface

$$(\hat{\mathbf{R}}\mathbf{Q})(\mathbf{Q}^{-1}\hat{\mathbf{S}}) = \hat{\mathbf{R}}\hat{\mathbf{S}}. \tag{4.17}$$

Por tanto, la solución general de $\mathbf{R}$ y $\mathbf{S}$ son las ecuaciones

$$\mathbf{R} = \hat{\mathbf{R}}\mathbf{Q}, \text{ y } \mathbf{S} = \mathbf{Q}^{-1}\hat{\mathbf{S}}. \tag{4.18}$$

donde la matriz invertible $\mathbf{Q}$. Recordando que para matrices ortonormales la inversa es igual a la traspuesta, $\mathbf{Q}^T = \mathbf{Q}^{-1}$, la idea es asegurar que se cumplan las siguientes restricciones, llamadas métricas,

$$\hat{\mathbf{i}}_f^T\mathbf{Q}\mathbf{Q}^T\hat{\mathbf{i}}_f = 1, \hat{\mathbf{j}}_f^T\mathbf{Q}\mathbf{Q}^T\hat{\mathbf{j}}_f = 1, \text{ y } \hat{\mathbf{i}}_f^T\mathbf{Q}\mathbf{Q}^T\hat{\mathbf{j}}_f = 0, \tag{4.19}$$

donde $\hat{\mathbf{i}}_f^T = (i_1, i_2, i_3)$, y $\hat{\mathbf{j}}_f^T = (j_1, j_2, j_3)$.

4.3. Restricciones Métricas

Morita y Kanade[7] presentan un método para la solución de las restricciones métricas. Ellos parten del sistema

$$\mathbf{L} = \mathbf{Q}\mathbf{Q}^T = \begin{pmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \mathbf{q}_3^T \end{pmatrix} \begin{pmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 \end{pmatrix} = \begin{pmatrix} \mathbf{q}_1^T\mathbf{q}_1 & \mathbf{q}_1^T\mathbf{q}_2 & \mathbf{q}_1^T\mathbf{q}_3 \\ \mathbf{q}_2^T\mathbf{q}_1 & \mathbf{q}_2^T\mathbf{q}_2 & \mathbf{q}_2^T\mathbf{q}_3 \\ \mathbf{q}_3^T\mathbf{q}_1 & \mathbf{q}_3^T\mathbf{q}_2 & \mathbf{q}_3^T\mathbf{q}_3 \end{pmatrix} \tag{4.20}$$

[1] La norma de Frobenius se define como

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{trace(\mathbf{A}\mathbf{A}^H)},$$

donde A^H es la transpuesta conjugada o matriz Hermitiana.

Llamada : $< \mathbf{R}, \mathbf{S} > \leftarrow$ `SfM_Factorizacion` $(\mathbf{W})$

Entradas: La matriz de mediciones $\mathbf{W}_{2F\times P} = \begin{pmatrix} \mathbf{U}_{F\times P} \\ \mathbf{V}_{F\times P} \end{pmatrix}$

Salidas : La rotación de la cámara a través de la secuencia $\mathbf{R}_{2F\times 3}$ y la matriz de la forma del objecto $\mathbf{S}_{3\times P}$

`// Calcular la traslación de la cámara con respecto al objeto`
$< \overline{\mathbf{u}}_{F\times 1}, \overline{\mathbf{v}}_{F\times 1} > \leftarrow$ `Centroide` $(\mathbf{U}, \mathbf{V})$;
`// Obtener las medidas centralizadas`
$\tilde{\mathbf{U}} \leftarrow \mathbf{U} - \overline{\mathbf{u}} \cdot \mathbf{1}_{1\times P}; \tilde{\mathbf{V}} \leftarrow \mathbf{V} - \overline{\mathbf{v}} \cdot \mathbf{1}_{1\times P}; \tilde{\mathbf{W}} = \begin{pmatrix} \tilde{\mathbf{U}} \\ \tilde{\mathbf{V}} \end{pmatrix}$;
`// Aplicar factorización`
$< \mathcal{U}, \mathcal{S}, \mathcal{V}^T > \leftarrow$ `Descomposición_Valores_Singulares` $(\tilde{\mathbf{W}})$;
`// Estimar el movimiento`
$\mathbf{R}' \leftarrow \mathcal{U}(:, 1:3)$;
`// Estimar la forma`
$\mathbf{S}' \leftarrow \mathcal{S}\mathcal{V}(1:3, :)$;
`// Aplicar restricciones métricas`
$\mathbf{Q} \leftarrow$ `Restricciones_Métricas` $(\mathbf{R}', \mathbf{S}')$;
`// Calcular el movimiento`
$\mathbf{R} \leftarrow \mathbf{R}'\mathbf{Q}$;
`// Calcular la forma`
$\mathbf{S} \leftarrow \mathbf{Q}^{-1}\mathbf{S}'$;

Algorithm 6: Algoritmo de Reconstrucción usando Factorización. El algoritmo recibe como entrada una matriz de mediciones $\mathbf{W}$ y calcula como salida las matrices de movimiento $\mathbf{R}$, forma $\mathbf{S}$ y posición $\mathbf{t}$ de la cámara con respecto al objeto.

Notando que la matriz simétrica $\mathbf{L}_{3\times 3}$ puede ser representada, elemento por elemento, como

$$\mathbf{L} = \begin{pmatrix} l_1 & l_2 & l_3 \\ l_2 & l_4 & l_5 \\ l_3 & l_5 & l_6 \end{pmatrix}, \tag{4.21}$$

donde las incógnitas l_i deben ser encontradas resolviendo los sistemas lineales

$$\mathbf{i}_f^T \mathbf{L} \mathbf{i}_f = 1, \mathbf{j}_f^T \mathbf{L} \mathbf{j}_f = 1, \text{ y } \mathbf{i}_f^T \mathbf{L} \mathbf{j}_f = 0. \tag{4.22}$$

Expandiendo los productos, considerando sus componentes, es posible llegar a la construcción de un sistema lineal que puede ser expresado en forma compacta por

$$\mathbf{G}\mathbf{l} = \mathbf{c}, \tag{4.23}$$

donde $\mathbf{G}_{3F\times 6}$, $\mathbf{l}_{6\times 1}$, y $\mathbf{c}_{3F\times 1}$ están definidas como

$$\mathbf{G} = \begin{pmatrix} \mathbf{g}^T(\mathbf{i}_1, \mathbf{i}_1) \\ \vdots \\ \mathbf{g}^T(\mathbf{i}_F, \mathbf{i}_F) \\ \mathbf{g}^T(\mathbf{j}_1, \mathbf{j}_1) \\ \vdots \\ \mathbf{g}^T(\mathbf{j}_F, \mathbf{j}_F) \\ \mathbf{g}^T(\mathbf{i}_1, \mathbf{j}_1) \\ \vdots \\ \mathbf{g}^T(\mathbf{i}_F, \mathbf{j}_F) \end{pmatrix}, \ \mathbf{l} = \begin{pmatrix} \mathbf{l}_1 \\ \vdots \\ \mathbf{l}_6 \end{pmatrix}, \text{ y } \mathbf{c} = \begin{pmatrix} \mathbf{1}_{2F\times 1} \\ \mathbf{0}_{F\times 1} \end{pmatrix}, \tag{4.24}$$

y a su vez $\mathbf{g}^T(\mathbf{a}_f, \mathbf{b}_f)$ es la transpuesta de la función $\mathbf{g}(\mathbf{a}_f, \mathbf{b}_f)$, la cual se define como

$$\mathbf{g}(\mathbf{a}_f, \mathbf{b}_f) = \begin{pmatrix} a_1 b_1 \\ a_1 b_2 + a_2 b_1 \\ a_1 b_3 + a_3 b_1 \\ a_2 b_2 \\ a_2 b_3 + a_3 b_2 \\ a_3 b_3 \end{pmatrix}, \tag{4.25}$$

para vectores $\mathbf{a}_f = (a_1, a_2, a_3)^T$ y $\mathbf{b}_f = (b_1, b_2, b_3)^T$. Finalmente, $\mathbf{l}$ puede ser calculada usando el producto

$$\mathbf{l} = \mathbf{G}^+ \mathbf{c}, \tag{4.26}$$

donde $\mathbf{G}^+$ es la pseudoinversa de Moore-Penrose de $\mathbf{G}$[2]. Una vez obtenida $\mathbf{l}$ se puede reconstruir $\mathbf{L}$. Enseguida, la matriz $\mathbf{Q}$ puede ser obtenida de una variedad de formas, incluyendo la aplicación de factorización de eigenvalores, tal que

$$\mathbf{L} = \mathbf{P}\Lambda\mathbf{P}^T = \mathbf{P}\Lambda^{1/2}\Lambda^{1/2}\mathbf{P}^T = \mathbf{Q}\mathbf{Q}^T. \tag{4.27}$$

Sin embargo, puede resultar que, debido a inexactitudes numéricas, la matriz $\mathbf{L}$ no sea positiva semidefinida, y por consecuencia alguno de sus eigenvalores resulte negativo.

Higham[4] da una solución a este problema y propone un método para calcular la matriz simétrica positiva definida más cercana a una matriz arbitraria. En su aproximación, Highman prueba que, bajo la norma de Frobenius, esta matriz puede obtenerse de la siguiente manera. Primero se calcula la matriz simétrica $\mathbf{L}_{sym}$ como

$$\mathbf{L}_{sim} = \frac{\mathbf{L} + \mathbf{L}^T}{2}. \tag{4.28}$$

Luego, se obtiene la descomposición del eigensistema de $\mathbf{L}_{sim}$ como

$$\mathbf{L}_{sim} = \mathcal{U}\Lambda\mathcal{U}^T. \tag{4.29}$$

[2]Una solución a la Moore-Penrose pseudoinversa está dada por la descomposición en valores singulares. Así, si $\mathbf{A} = \mathcal{U}\mathbf{S}\mathcal{V}^T$, la pseudoinversa está dada por $\mathbf{A}^+ = \mathcal{V}\mathbf{S}^+\mathcal{U}^T$, donde $\mathbf{S}^+$ se construye dejando los elementos con valor cero de $\mathbf{S}$ iguales e invirtiendo los elementos de la diagonal. En la práctica, elementos de la diagonal cercanos a cero con convertidos a cero.

Finalmente, la matriz positiva semidefinida más cercana a $\mathbf{L}$, en la norma de Frobenius, es

$$\mathbf{L}_{psd} = \mathcal{U}\Lambda_{+}\mathcal{U}^{T}, \tag{4.30}$$

donde en Λ_{+} corresponde a la matriz Λ con todos sus valores negativos puestos a cero.

4.4. Implementación

Seo[9] ofrece una implementación del algoritmo para recuperar la estructura y el movimiento a partir de una secuencia de imágenes usando el método de factorización para los casos de proyección ortográfica[11], paraperspectiva[8], y proyectiva[12][3]. De manera interesante Seo muestra experimentalmente que la aproximación, usando proyección ortográfica, resulta ser la más robusta bajo ciertas condiciones. Sin embargo, se debe tener cuidado de que las restricciones del modelo de proyección ortográfico se cumplan. Esto puede lograrse colocando los objetos alejados de la cámara y haciendo uso de una lente telecéntrica. El algoritmo para el cálculo de la estructura a partir del movimiento mediante el método de factorización, puede ser sumarizado por las líneas de pseudocódigo en el Algoritmo 6.

En la Figura 4.3 se muestra el resultado de la implementación del algoritmo de estructura y movimiento a partir de una secuencia de imágenes usando factorización. Algunos cuadros de la secuencia son desplegados en (a)-(c). Luego en (d) se muestran las trayectorias obtenidas mediante el algoritmo de Lucas-Kanade[10]. Los resultados para la estructura del objeto y el movimiento de la cámara se muestran en (d) y (e). Para el cálculo de la solución por factorización solo se tomaron las observaciones que pudieron ser seguidas a lo largo de la secuencia.

Sumario

El algoritmo para la obtención de la estructura y el movimiento a partir de una secuencia de imágenes constituyó un gran adelanto en Visión por Computadora. Su invención fue seminal en el desarrollo de una gran cantidad de variantes. En este capítulo hemos presentado el planteamiento del problema en términos de una factorización. Hemos visto cómo la factorización puede ser resuelta en forma muy robusta mediante la descomposición en valores singulares. Aún así, dado que la solución no es única, se requiere aplicar algunas restricciones geométricas. Sin embargo, aún cuando el modelo matemático es completo, hay algunos detalles técnicos que observar para lograr una satisfactoria solución numérica. Aún cuando el algoritmo que asume proyección ortográfica tiene un espacio de aplicación restringido, en ocasiones da soluciones más robustas que modelos más completos, como el que asume proyección paraperspectiva, o proyectiva. La reconstrucción tridimensional puede ser complementada mediante el mapeo de la imagen en las superficies de la reconstrucción. Uno de los problemas abiertos es la construcción de trayectorias que guarden la historia de cada

[3]En proyección paraperspectiva las proyecciones ocurren en la dirección de la línea de proyección del centro de los objetos. Basri[1] ha mostrado que la proyección paraperspectiva es exactamente lo mismo que una proyección afin.

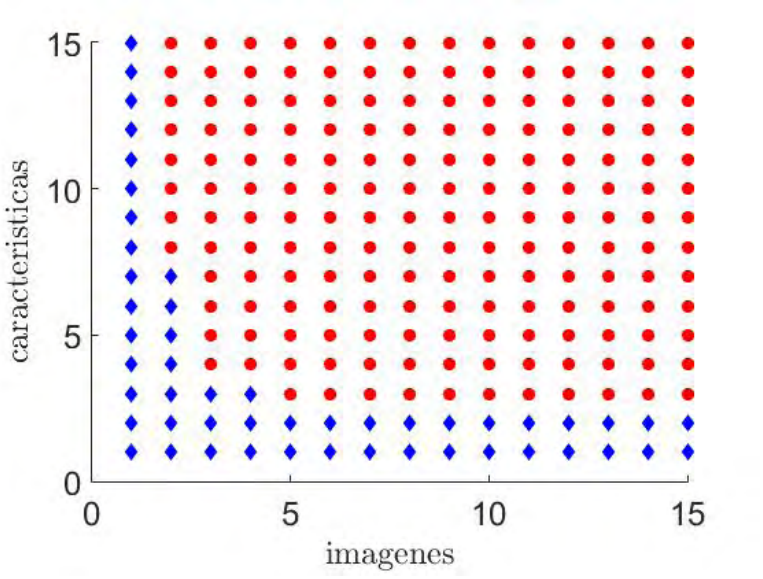


Figura 4.4: El número de imágenes F y características para poder hacer la reconstrucción por factorización sigue la relación $2FP - 4F - 3P \geq 0$. Los círculos rojos muestran combinaciones válidas, mientras que los rombos azules muestran combinaciones insuficientes de imágenes o características.

punto en el mundo. Esto ayudaría en la solución de problemas visuales, marcadamente la reconstrucción tridimensional a partir de movimiento.

El método de factorización ha despertado un gran interés entre los investigadores, de tal forma que diversas extensiones han surgido. Por ejemplo, Morita y Kanade[7] presentaron una formulación para la obtención de la estructura y el movimiento bajo proyección de perspectiva. Alrededor del mismo tiempo, Triggs[12] presentó una formulación para el caso de la proyección de perspectiva. Un resumen de las diversas técnicas que surgieron hasta ese momento fue presentado por Kanade y Morris[5]. Algunos investigadores han propuesto soluciones a algunas limitaciones del método original. Por ejemplo, Hartley y Schaffalitzky[3] proponen una solución al problema de la pérdida de características durante su seguimiento. Así también, Kennedy *et al.*[6] estudian la reconstrucción bajo la suposición de que las imágenes son procesadas conforme se van capturando.

Ejercicios

1. La Figura 4.4 ilustra que el mínimo número de características P e imágenes F requeridas para solucionar el problema de la reconstrucción vía factorización. Muestra que esta relación está descrita por

$$2FP - 4F - 3P \geq 0. \tag{4.31}$$

2. Deducir los términos de (4.25).

3. Implementa el Algoritmo 6 para la reconstrucción basada en la factorización de la matriz de mediciones.

Llamada : $\mathbf{Q} \leftarrow$ `Restricciones_Métricas` $(\mathbf{R}, \mathbf{S})$
Entradas: Aproximaciones a la matriz de rotaciones $\mathbf{R}_{2F\times 3}$ y la matriz de estructura $\mathbf{P}_{3\times P}$
Salidas : Una matriz ortonormal $\mathbf{Q}_{3\times 3}$ que incluye las restricciones métricas que eliminan la ambigüedad en rotación y estructura

`// Construir la matriz` $\mathbf{G}$
for i = *1:F* **do**
  `// Obtener los componentes de la rotación`
  $\mathbf{i} \leftarrow \mathbf{R}(2(i-1)+1,:)$; $\mathbf{j} \leftarrow \mathbf{R}(2(i-1)+2,:)$;
end
`// Completar las hileras de` $\mathbf{G}$ `utilizando (4.25)`
$\mathbf{G}(i,:) = \mathbf{g}^T(\mathbf{i},\mathbf{i})$; $\mathbf{G}(i+F,:) = \mathbf{g}^T(\mathbf{j},\mathbf{j})$;$\mathbf{G}(i+2F,:) = \mathbf{g}^T(\mathbf{i},\mathbf{j})$;
`// Construir el vector de solución` $\mathbf{c}$
$\mathbf{c} \leftarrow \begin{pmatrix} \mathbf{1}_{2F\times 1} \\ \mathbf{0}_{F\times 1} \end{pmatrix}$;
`// Resolver para` $\mathbf{l}$ `utilizando la pseudo-inversa de` $\mathbf{G}$
$\mathbf{G} \leftarrow \mathcal{U}\mathcal{S}\mathcal{V}^T$; `// descomposición SVD de` $\mathbf{G}$
$\mathbf{G}^+ \leftarrow \mathcal{U}\mathcal{S}^+\mathcal{V}^T$; `// calcular la pseudo-inversa de` $\mathbf{G}$
$\mathbf{l} \leftarrow \mathbf{G}^+\mathbf{c}$;
`// Representar` $\mathbf{l}$ `en forma matricial`
$\mathbf{L} \leftarrow$ `Representación_Matricial`$(\mathbf{l})$;
`// Obtener la mejor aproximación a la matriz positiva definida de` $\mathbf{L}$
$\mathbf{L}_{sim} \leftarrow \dfrac{\mathbf{L}+\mathbf{L}^T}{2}$;
$\mathbf{L}_{sim} \leftarrow \mathcal{U}\Lambda\mathcal{U}^T$; `// descomposición SVD de` $\mathbf{L}_{sim}$
$\mathbf{L}_{psd} \leftarrow \mathcal{U}\Lambda_+\mathcal{U}^T$; `// Eliminación de entradas negativas de` Λ
`// Descomposición de` $\mathbf{L}_{psd}$ `en su eigensistema`
$\mathbf{L}_{psd} \leftarrow \mathbf{P}\Lambda\mathbf{P}^T$;
`// Solución para` $\mathbf{Q}$
$\mathbf{Q} \leftarrow \mathbf{P}\Lambda^{1/2}$;

Algorithm 7: Algoritmo para el Cálculo de las Restricciones Métricas. Después de formar las matrices $\mathbf{G}$ y $\mathbf{c}$ se resuelve para $\mathbf{l}$. Representando $\mathbf{l}$ en forma matricial como $\mathbf{L}$, se obtiene la mejor matriz positiva definida que la representa antes de resolver para $\mathbf{Q}$.

Bibliografía

[1] Basri, Ronen: *Paraperspective≡ Affine*. International Journal of Computer Vision, 19(2):169–179, 1996.

[2] Golub, Gene y Charles Van Loan: *Matrix Computations*, volumen 3. JHU Press, 2012.

[3] Hartley, Richard y Frederik Schaffalitzky: *PowerFactorization: 3D Reconstruction with Missing or Uncertain Data*. En *Australia-Japan Advanced Workshop on Computer Vision*, volumen 74, páginas 76–85, 2003.

[4] Higham, Nicholas: *Computing a Nearest Symmetric Positive Semidefinite Matrix*. Linear Algebra and its Applications, 103:103–118, 1988.

[5] Kanade, Takeo y Daniel Morris: *Factorization Methods for Structure from Motion*. Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 356(1740):1153–1173, 1998.

[6] Kennedy, Ryan, Laura Balzano, Stephen Wright y Camillo Taylor: *Online Algorithms for Factorization-Based Structure from Motion*. arXiv:1309.6964, 2013.

[7] Morita, Toshihiko y Takeo Kanade: *A Sequential Factorization Method for Recovering Shape and Motion from Image Streams*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(8):858–867, 1997.

[8] Poelman, Conrad y Takeo Kanade: *A Paraperspective Factorization Method for Shape and Motion Recovery*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(3):206–218, 1997.

[9] Seo, Naotoshi: *Shape and Motion from Image Streams: A Factorization Method*. `http://note.sonots.com/SciSoftware/Factorization.html`, 2006.

[10] Shi, Jianbo y Carlo Tomasi: *Good Features to Track*. En *IEEE Conference on Computer Vision and Pattern Recognition*, páginas 593–600, 1994.

[11] Tomasi, Carlo y Takeo Kanade: *Shape and Motion from Image Streams under Orthography: A Factorization Method*. International Journal of Computer Vision, 9(2):137–154, 1992.

[12] Triggs, Bill: *Factorization Methods for Projective Structure and Motion*. En *IEEE Conference on Computer Vision and Pattern Recognition*, páginas 845–851, 1996.

CAPÍTULO 5

Reconstrucción bajo Proyección Perspectiva

El problema de la estructura a partir de movimiento se puede resolver mediante algoritmos que encuentren la correspondencia entre puntos en una secuencia de imágenes y propiedades geométricas en las cámaras. Si bien geométricamente el problema está resuelto, en la práctica hace falta tratar el aspecto numérico con la finalidad de obtener soluciones robustas. En este capítulo hacemos una revisión de algunas definiciones geométricas que ayudan a expresar la relación entre los sistemas de referencias de las cámaras, los puntos que se observan, y sus proyecciones en imágenes. Estudiamos el algoritmo de los ocho puntos y verificamos cómo con él se puede recuperar la estructura de los objetos a partir de las coordenadas de puntos correspondientes en dos imágenes. Posteriormente, vemos como los valores obtenidos entre dos imágenes pueden ser utilizados como semilla para obtener la reconstrucción de objetos utilizando un conjunto de imágenes.

5.1. Geometría Epipolar

Cuando dos cámaras comparten campo visual, es posible utilizar las correspondencias que se establecen entre puntos en las imágenes para recuperar la información tridimensional de la escena, aprovechando la construcción de lo que se conoce como *geometría epipolar*. En la Figura 5.1 se ilustra tal situación. En ella, dado el arreglo de dos cámaras centradas en los sistemas de referencia $\{C_1\}$ y $\{C_2\}$, un punto común en el espacio $\mathbf{q}$ es proyectado en un punto $\mathbf{p}$ en la cámara centrada en $\{C_1\}$. Desde el punto de vista de la cámara en $\{C_2\}$ la proyección pudiera haber sido generada por, entre una infinidad de otros, cualquiera de los puntos $\mathbf{q}, \mathbf{q}_1, \mathbf{q}_2$. Una manera de resolver tal ambigüedad sería encontrar cuál de los puntos, en la cámara centrada en $\{C_2\}$, $\mathbf{p}', \mathbf{p}'_1, \mathbf{p}'_2$ es el correspondiente a $\mathbf{p}$. La geometría epipolar permite establecer que la correspondencia se encuentra en la llamada línea epipolar $\mathbf{l}'$ en $\{C_2\}$. Por otro lado, la línea que va del centro del sistema de referencia en $\{C_1\}$ al centro de $\{C_2\}$ cruza los planos de las imágenes en los epipolos $\mathbf{e}$ y $\mathbf{e}'$. Una propiedad interesante es que todas las líneas epipolares cruzan por los epipolos. Esta geometría genera planos entre los orígenes de $\{C_1\}$, $\{C_2\}$, y $\mathbf{p}$, conocidos como planos epipolares. Por supuesto, esta geometría genera la correspondiente línea epipolar $\mathbf{l}$ en $\{C_1\}$.

La alineación entre los sistemas de referencia $\{C_1\}$ y $\{C_2\}$ puede lograrse encontrando la rotación $\mathbf{R}$ y traslación $\mathbf{t}$ apropiadas. Una vez establecidas las correspondencias entre los puntos $\mathbf{p}$ y $\mathbf{p}'$, pertenecientes respectivamente a las cámaras centrados en $\{C_1\}$ y $\{C_2\}$, una relación que puede notarse es que el vector que va del origen de $\{C_1\}$ a $\mathbf{p}$ es perpendicular

al vector normal al plano formado por la línea que va del origen de $\{C_1\}$ al origen de $\{C_2\}$ y la línea que va del origen de $\{C_2\}$ a $\mathbf{p}'$. Esta condición puede plantearse de manera más concreta como

$$\mathbf{p}^T \left(\mathbf{t} \times [\mathbf{R}\mathbf{p}']\right) = 0, \tag{5.1}$$

donde $\mathbf{p}$ y $\mathbf{p}'$ se expresan en coordenadas homogéneas[1] y $\times$ es el producto cruz entre dos vectores. Esta operación puede ser condensada como

$$\mathbf{p}^T \mathbf{E} \mathbf{p}' = 0, \tag{5.2}$$

donde $\mathbf{E}$ es conocida como la matriz esencial y está dada por

$$\mathbf{E} = \lfloor \mathbf{t}_\times \rfloor \mathbf{R}, \tag{5.3}$$

y $\lfloor \mathbf{t}_\times \rfloor$ es una matriz *skew symmetric* que sirve para representar un producto cruz. Es decir, para el vector $\mathbf{t}^T = (t_1, t_2, t_3)$ la matriz $\lfloor \mathbf{t}_\times \rfloor$ está dada por

$$\lfloor \mathbf{t}_\times \rfloor = \begin{pmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{pmatrix}. \tag{5.4}$$

La matriz esencial captura la geometría epipolar de una estructura de toma de imágenes. Longuet-Higgins[5] mostró que para resolver la matriz esencial se requería encontrar la correspondencia de ocho puntos en el espacio. Sin embargo, las correspondencias son establecidas entre puntos de las imágenes que forman parte del arreglo estéreo. La relación entre los puntos $\mathbf{p}$ y $\mathbf{p}'$, expresados en los sistemas de referencia de las imágenes centradas en $\{C_1\}$ y $\{C_2\}$, y sus observaciones $\hat{\mathbf{p}}$ y $\hat{\mathbf{p}}'$, respectivamente, está dada por

$$\hat{\mathbf{p}} = \mathbf{K}\mathbf{p}, \text{ y } \hat{\mathbf{p}}' = \mathbf{K}'\mathbf{p}', \tag{5.5}$$

donde $\mathbf{K}$ es la matriz de la cámara, y está dada por

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & -c_x \\ 0 & f_y & -c_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{5.6}$$

en donde (f_x, f_y) son los ejes focales con respecto a las direcciones horizontal y vertical y (c_x, c_y) es el centro óptico de las cámaras. Es decir, la geometría epipolar, en términos de los pixeles correspondientes en la imagen $\hat{\mathbf{p}}$ y $\hat{\mathbf{p}}'$ puede ser establecida como

$$\hat{\mathbf{p}}^T \mathbf{K}^{-T} \mathbf{E} \mathbf{K}'^{-1} \hat{\mathbf{p}}' = \hat{\mathbf{p}}^T \mathbf{F} \hat{\mathbf{p}}' = 0. \tag{5.7}$$

Dado un punto $\hat{\mathbf{u}}_i$ y la matriz fundamental $\mathbf{F}$, la línea epipolar en la segunda imagen corresponde a $\mathbf{F}^T \hat{\mathbf{u}}_i$. Asimismo, $\mathbf{F}\hat{\mathbf{u}}'_i$ es la línea epipolar correspondiente a $\hat{\mathbf{u}'}_i$ en la primera imagen. La matriz fundamental establece la propiedad de que un punto $\hat{\mathbf{u}'}_i$ está en la línea epipolar $\mathbf{F}\hat{\mathbf{u}'}_i$ sí y sólo sí $\hat{\mathbf{u}'}_i \mathbf{F} \hat{\mathbf{u}}_i = 0$. Las matrices esencial y fundamental pueden ser usadas para reconstruir una escena, rectificar las imágenes, calcular invariantes proyectivos, detectar falsas correspondencias, y resolver el problema de la estereoscopía.

[1] La geometría proyectiva es inherentemente no lineal. Una forma de incorporar el herramental de Álgebra Lineal en la solución de problemas de perspectiva consiste en ampliar la representación agregando un escalar en una nueva dimensión y luego escalando las demás coordenadas por ese elemento agregado. Así, un vector $(x, y)^T$ se expande a $(x/w, y/w, w)$.

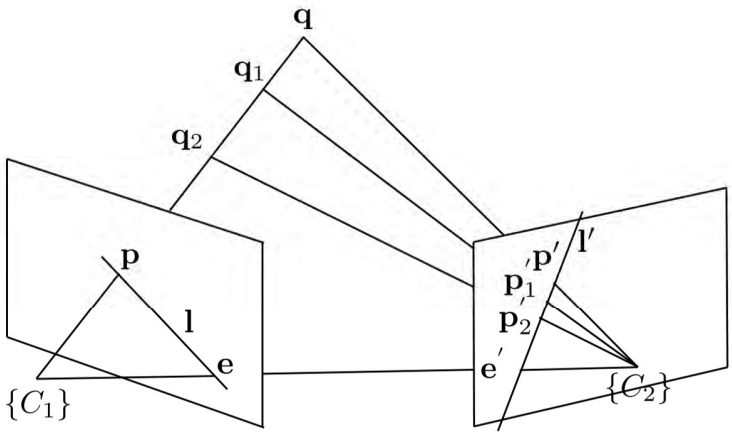


Figura 5.1: Geometría Epipolar. Dado un arreglo de dos cámaras, centradas en $\{C_1\}$ y $\{C_2\}$, que comparten un campo visual, un punto en el espacio $\mathbf{q}$ es proyectado en un punto $\mathbf{p}$ en la cámara $\{C_1\}$. Desde el punto de vista de la cámara en $\{C_2\}$ el punto que generó tal proyección pudiera ser, entre una infinidad de otros, cualquiera de los puntos $\mathbf{q}, \mathbf{q}_1, \mathbf{q}_2$. La manera de resolver tal ambigüedad sería encontrar cuál de los puntos, $\mathbf{p}', \mathbf{p}'_1, \mathbf{p}'_2$ es el correspondiente a $\mathbf{p}$. La geometría epipolar permite establecer que la correspondencia se encuentra en la línea epipolar $\mathbf{l}'$. La línea que va del centro de $\{C_1\}$ al centro de $\{C_2\}$ cruza los planos de las imágenes en los epipolos $\mathbf{e}$ y $\mathbf{e}'$. Una propiedad interesante es que las líneas epipolares cruzan por los epipolos. Esta geometría genera planos entre los vértices $\{C_1\}$, $\{C_2\}$, y $\mathbf{p}$, conocidos como planos epipolares. Esta geometría genera las correspondientes líneas epipolares $\mathbf{l}$ en $\{C_1\}$ y $\mathbf{l}'$ en $\{C_2\}$.

5.2. Algoritmo de los Ocho Puntos

Para dos puntos $\mathbf{u} = (u, v, 1)^T$ y $\mathbf{u}' = (u', v', 1)^T$ correspondientes en la imagen, la matriz fundamental $\mathbf{F}$ de 3×3 se define como

$$\mathbf{u}^T \mathbf{F} \mathbf{u}' = 0. \tag{5.8}$$

Expandiendo tenemos

$$(u, v, 1) \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0, \tag{5.9}$$

donde al hacer las multiplicaciones resulta en el sistema

$$uu'f_{11} + uv'f_{21} + uf_{31} + vu'f_{12} + vv'f_{22} + vf_{32} + u'f_{13} + v'f_{23} + f_{33} = 0. \tag{5.10}$$

Dado un conjunto de correspondencias entre puntos $\mathbf{u}_k$ y $\mathbf{u'}_k$, para $k = 1, \ldots, m$, podemos formar un sistema lineal con la siguiente estructura

$$\begin{pmatrix} u_1u'_1 & u_1v'_1 & u_1 & v_1u'_1 & v_1v'_1 & v_1 & u'_1 & v'_1 & 1 \\ \vdots & \ddots & & & & & & & \vdots \\ u_mu'_m & u_mv'_m & u_m & v_mu'_m & v_mv'_m & v_m & u'_m & v'_m & 1 \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{pmatrix} = 0. \tag{5.11}$$

En notación matricial, la expresión anterior puede escribirse en forma más compacta como

$$\mathbf{Af} = 0. \tag{5.12}$$

Dado que la matriz fundamental se define hasta un factor de escala, se usa la restricción de que la magnitud de $\mathbf{f}$ es igual a uno. Así, la matriz fundamental $\mathbf{F}$ requiere la definición de nueve entradas. Sin embargo, se tiene la restricción adicional sobre la magnitud de $\mathbf{f}$. Por tanto, la matriz $\mathbf{A}$ tiene sólo ocho columnas linealmente independientes y por ello sólo se requiere el establecimiento de la correspondencia entre ocho puntos, excepción hecha de estructuras como planos, loa cuales veremos más adelante. De hecho, en muchas ocasiones tenemos muchas más de ocho correspondencias, lo que da pie a buscar la solución de mínimos cuadrados del sistema. Alternativamente, buscamos el eigenvector de $\mathbf{V}$ asociado al valor singular, σ_9, más pequeño de la descomposición en valores singulares (SVD) de $\mathbf{A}$.

Dado que la proyección de los puntos en la imagen por la matriz $\mathbf{F}$ resultan en las líneas epipolares sobre el plano de la segunda imagen, se puede deducir que la matriz $\mathbf{F}$ es rango dos. Todavía más, específicamente su espacio nulo izquierdo, $\mathbf{F}^T\mathbf{u} = 0$, y derecho, $\mathbf{Fu'} = 0$, corresponden con los epipolos en las imágenes en los sistemas de referencia $\{2\}$ y $\{1\}$ respectivamente. Una forma de forzar el rango dos de $\mathbf{F}$ es calculando la descomposición en valores singulares, tal que

$$\begin{aligned} \mathbf{F} &= \mathbf{U\Sigma V}^T \\ &= \mathbf{U} \begin{pmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{pmatrix} \mathbf{V}^T, \end{aligned} \tag{5.13}$$

y haciendo el valor singular más pequeño, igual a cero, tal que

$$\mathbf{F'} = \mathbf{U} \begin{pmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & 0 \end{pmatrix} \mathbf{V}^T. \tag{5.14}$$

Llamada : $\mathbf{F} \leftarrow$ `Calcular_Matriz_Fundamental` $(\mathbf{P}, \mathbf{Q})$
Entradas: Los puntos correspondientes $\mathbf{P}_{2\times n} = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$, con $\mathbf{p}_i = (u_i, v_i)^T$ y $\mathbf{Q}_{2\times n} = (\mathbf{q}_1, \ldots, \mathbf{q}_n)$, con $\mathbf{q}_i = (u'_i, v'_i)^T$, donde el punto $\mathbf{p}_i$ corresponde con el punto $\mathbf{q}_i$
Salidas : La matriz fundamental $\mathbf{F}$ entre las imágenes pertenecientes a las cámaras centradas en los sistemas de referencia $\{i\}$ y $\{j\}$

`// Normalizar coordenadas, usando §5.3, obteniendo las nuevas coordenadas y la transformación.`
$\langle \mathbf{P}_n, \mathbf{T}_p \rangle \leftarrow$ `Normalizar` $(\mathbf{P})$; $\langle \mathbf{Q}_n, \mathbf{T}_q \rangle \leftarrow$ `Normalizar` $(\mathbf{Q})$;
`// Formar la matriz` $\mathbf{A}$

$$\mathbf{A} = \begin{pmatrix} u_1u'_1 & u_1v'_1 & u_1 & v_1u'_1 & v_1v'_1 & v_1 & u'_1 & v'_1 & 1 \\ \vdots & \ddots & & & & & & & \vdots \\ u_mu'_m & u_mv'_m & u_m & v_mu'_m & v_mv'_m & v_m & u'_m & v'_m & 1 \end{pmatrix};$$

`// Obtener el espacio nulo de` $\mathbf{A}$
$\mathbf{f} \leftarrow$ `Espacio_Nulo` $(\mathbf{A})$;
`// Reordenar el vector como una matriz de` 3×3`. Si la reconstrucción es por columnas, habrá que trasponer la matriz resultante`
$\mathbf{F}' \leftarrow$`Reconstruir` $(\mathbf{f})$;
`// Asegurar que` $\mathbf{F}'$ `tenga` *rank*`-2`
$\langle \mathbf{U}_F, \mathbf{S}_F, \mathbf{V}_F \rangle \leftarrow$ `Descomposición_Valores_Singulares` $(\mathbf{F}')$;
$\mathbf{S}_F(3,3) \leftarrow 0$;
$\mathbf{F}' \leftarrow \mathbf{U}_F\mathbf{S}_F\mathbf{V}_F^T$;
`// Revertir los efectos de la normalización`
$\mathbf{F} \leftarrow \mathbf{T}_q^T\mathbf{F}'\mathbf{T}_p$;
`// Normalizar para obtener la matriz fundamental`
$\mathbf{F} \leftarrow \mathbf{F}/\|\mathbf{F}\|$;

Algorithm 8: Cálculo de la Matriz Fundamental. El algoritmo recibe como entrada los puntos correspondientes entre imágenes y regresa la matriz fundamental.

5.3. Limitaciones del Algoritmo de los Ocho Puntos

Si bien sencillo, el método de los ocho puntos es muy sensible a transformaciones de las coordenadas. Hartley[3] hace la observación de que una transformación $\mathbf{T}$ aplicada a los puntos de la imagen en correspondencia, hará que la solución de la matriz fundamental $\mathbf{F}$ cambie. Es decir, supóngase que las coordenadas de puntos correspondientes pasan a través de las transformaciones $\mathbf{T}$ y $\mathbf{T}'$, tal que

$$\hat{\mathbf{u}} = \mathbf{T}\mathbf{u}, \text{ y } \hat{\mathbf{u}}' = \mathbf{T}'\mathbf{u}'. \tag{5.15}$$

Sustituyendo en (5.8), tenemos que la nueva relación entre las coordenadas está dada por

$$\hat{\mathbf{u}}^T\mathbf{T}'^{-T}\mathbf{F}\mathbf{T}^{-1}\hat{\mathbf{u}}' = 0, \tag{5.16}$$

donde $\hat{\mathbf{F}} = \mathbf{T}'^{-T}\mathbf{F}\mathbf{T}^{-1}$ es la matriz fundamental de la correspondencia entre $\hat{\mathbf{u}}'$ y $\hat{\mathbf{u}}$.

Llamada : $< \mathbf{R}, \mathbf{t} > \leftarrow$ `Movimiento_entre_Imágenes` $(\mathbf{P}, \mathbf{Q})$
Entradas: Las correspondencias entre los puntos $\mathbf{P}_{2\times n} = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$, con $\mathbf{p}_i = (u_i, v_i)$ y $\mathbf{Q}_{2\times n} = (\mathbf{q}_1, \ldots, \mathbf{q}_n)$, con $\mathbf{q}_i = (u'_i, v'_i)$, y matriz de parámetros intrínsecos de la cámara $\mathbf{K}$.
Salidas : La rotación $\mathbf{R}$ y traslación $\mathbf{t}$ entre los sistemas de referencia centrados en la cámara $\{i\}$ y $\{j\}$

`// Calcular la matriz fundamental`
$\mathbf{F} \leftarrow$ `Calcular_Matriz_Fundamental` $(\mathbf{P}, \mathbf{Q})$
`// Obtener la matriz esencial`
$\mathbf{E} \leftarrow \mathbf{K}^T\mathbf{F}\mathbf{K}$;
`// Obtener la descomposición en valores singulares de` $\mathbf{E}$
$< \mathbf{U}_E, \mathbf{S}_E, \mathbf{V}_E > \leftarrow$ `Descomposición_Valores_Singulares` $(\mathbf{E})$;
`// Dada la matriz` $\mathbf{W} \leftarrow \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ `Obtener la rotación` $\mathbf{R}$ `y traslación` $\mathbf{t}$
$\mathbf{R}'_1 \leftarrow \mathbf{U}_E\mathbf{W}\mathbf{V}_E^T$; $\mathbf{R}'_2 \leftarrow \mathbf{U}_E\mathbf{W}^{-1}\mathbf{V}_E^T$; $\mathbf{T} \leftarrow \mathbf{U}_E\mathbf{S}_E\mathbf{U}_E^T$;
`// Reducir las soluciones espejo`
$\mathbf{R}'_1 \leftarrow \mathbf{R}'_1\|\mathbf{R}'_1\|$;$\mathbf{R}'_2 \leftarrow \mathbf{R}'_2\|\mathbf{R}'_2\|$;
`// Extraer el vector de traslación`
$\mathbf{t}' \leftarrow \begin{pmatrix} \mathbf{T}_{3,2} \\ \mathbf{T}_{1,3} \\ \mathbf{T}_{2,1} \end{pmatrix}$;
`// Resolver la ambigüedad`
$< \mathbf{R}, \mathbf{t} > \leftarrow$ `Resolver_Ambigüedad` $(\mathbf{R}'_1, \mathbf{R}'_2, \mathbf{t}', \mathbf{P}, \mathbf{Q})$;

Algorithm 9: Algoritmo de Reconstrucción entre Imágenes. El algoritmo recibe como entrada los puntos correspondientes entre imágenes y regresa la rotación y traslación entre cuadros.

Con el cambio de variable en (5.15), la ecuación $\mathbf{A}\mathbf{f} = 0$ tomaría la forma $\hat{\mathbf{A}}\hat{\mathbf{f}} = 0$. Intrínsico en el cambio de variable se tendría una relación $\hat{\mathbf{A}} = \mathbf{A}\mathbf{S}$, donde $\mathbf{S}$ es una matriz de 9×9 que se expresa en términos de $\mathbf{T}$. Para que la solución, con y sin la transformación, fuera la misma, $\hat{\mathbf{f}}$ debería corresponder al eigenvector más pequeño de $\hat{\mathbf{A}}^T\hat{\mathbf{A}}$, tanto como $\mathbf{f}$ corresponde al eigenvector más pequeño de $\mathbf{A}^T\mathbf{A}$. Es decir, mientras que la solución a $\mathbf{A}\mathbf{f} = 0$ corresponde al menor eigenvector de $\mathbf{A}^T\mathbf{A}$, expresado por $\mathbf{A}^T\mathbf{A}\mathbf{f} = \lambda\mathbf{f}$, para $\hat{\mathbf{A}}\hat{\mathbf{f}}$ se debería cumplir la siguiente relación para su eigensistema

$$\mathbf{S}^T\mathbf{A}^T(\mathbf{A}\mathbf{S})\hat{\mathbf{f}} = \lambda\hat{\mathbf{f}}. \tag{5.17}$$

La relación $\hat{\mathbf{A}}\hat{\mathbf{f}} = \mathbf{A}\mathbf{S}\hat{\mathbf{f}} = 0$ también implica que la relación entre las soluciones $\hat{\mathbf{f}}$ y $\mathbf{f}$ debería

Llamada : $< \mathbf{R}, \mathbf{t} > \leftarrow$ `Resolver_Ambigüedad` $(\mathbf{R}'_1, \mathbf{R}'_2, \mathbf{t}', \mathbf{P}, \mathbf{P}')$

Entradas: Dos posibles rotaciones $\mathbf{R}'_1$ y $\mathbf{R}'_2$ y traslación $\mathbf{t}$ entre los sistemas de referencia centrados en la cámara $\{i\}$ y $\{j\}$, y puntos correspondientes $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ y $\mathbf{P}' = \{\mathbf{p}'_1, \ldots, \mathbf{p}'_n\}$

Salidas : La rotación $\mathbf{R}$ y traslación $\mathbf{t}$ entre los sistemas de referencia centrados en la cámara $\{i\}$ y $\{j\}$

`// Considerar las combinaciones` $< \mathbf{R}'_1, \mathbf{t}' >, < \mathbf{R}'_2, \mathbf{t}' >, < \mathbf{R}'_1, -\mathbf{t}' >, < \mathbf{R}'_2, -\mathbf{t}' >$`. Para cada una de ellas hacer lo siguiente`

`// Supóngase que cada una de ellas se asigna sucesivamente a` $\mathbf{R}$ `y` $\mathbf{t}$`, donde`

$$\mathbf{R} \leftarrow \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}; \mathbf{t} \leftarrow \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix};$$

`// Resolver el sistema siguiente, donde implícitamente se supone que la rotación de la primera posición se asignan a la identidad y su traslación es cero. En adición, se supone que` $\mathbf{p}_j = (x_j, y_j)$ `y` $\mathbf{p}'_j = (x'_j, y'_j)$ `corresponden al punto` (u_j, v_j, w_j)

$$\begin{pmatrix} -1 & 0 & x_j \\ 0 & -1 & y_j \\ r_{31}x'_j - r_{11} & r_{32}x'_j - r_{12} & r_{33}x'_j - r_{13} \\ r_{31}y'_j - r_{21} & r_{32}y'_j - r_{22} & r_{33}y'_j - r_{23} \end{pmatrix} \begin{pmatrix} u_j \\ v_j \\ w_j \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ t_x - t_z x'_j \\ t_y - t_z y'_j \end{pmatrix}. ;$$

`// Escoger la combinación de rotación y traslación que no tenga valores negativos en los puntos reconstruidos.`

Algorithm 10: Algoritmo para Solución de Ambigüedad. Las cuatro posibles combinaciones son tratadas. La combinación que se escoge es aquella sin puntos del mundo, detrás de las cámaras.

ser $\hat{\mathbf{f}} = \mathbf{S}^{-1}\mathbf{f}$. Por tanto, de (5.17) uno puede derivar las siguientes equivalencias

$$\begin{aligned} \mathbf{S}^T\mathbf{A}^T(\mathbf{AS})\hat{\mathbf{f}} &= \mathbf{S}^T\mathbf{A}^T(\mathbf{AS})\mathbf{S}^{-1}\mathbf{f}, \\ &= \mathbf{S}^T\mathbf{A}^T\mathbf{Af}, \\ &= \mathbf{S}^T\lambda\mathbf{f}, \\ &= \lambda\mathbf{S}^T\mathbf{S}\hat{\mathbf{f}}, \end{aligned} \tag{5.18}$$

y (5.17) y (5.18) en general difieren. En otras palabras, si $\mathbf{S}$ es diferente a la matriz identidad, $\hat{\mathbf{f}}$ dejará de ser un eigenvector de la matriz $(\mathbf{AS})^T(\mathbf{AS})$. La solución, en palabras de Hartley[3], requiere normalizar las coordenadas expresándolas en un sistema de referencia canónico.

Otro de los problemas del algoritmo de los ocho puntos es el rango dinámico de la matriz $\mathbf{A}^T\mathbf{A}$. Hartley[3] ilustra el punto con el siguiente ejemplo. Considere una coordenada correspondiente a $(100, 100)$ para ambos $\mathbf{u}$ y $\mathbf{u}'$. Un renglón $\mathbf{a}^T$ de $\mathbf{A}$ tendría coordenadas

$$\mathbf{a}^T = (10^4, 10^4, 10^2, 10^4, 10^4, 10^2, 10^2, 10^2, 1). \tag{5.19}$$

Luego, las entradas de $\mathbf{A}^T\mathbf{A}$ tendrían alguna entrada de la forma $\mathbf{a}\mathbf{a}^T$. Por ejemplo, la diagonal de $\mathbf{A}^T\mathbf{A}$ tendrá entradas en el orden de magnitud de $(10^8, 10^8, 10^4, 10^8, 10^8, 10^4, 10^4, 10^4, 1)$. Usando la propiedad de entrelazado[2][2], Hartley muestra que en este caso la condición de la matriz[3] de $\mathbf{A}^T\mathbf{A}$ es mayor de 10^4+1. Un segundo problema fácil de observar es que el origen del sistema de coordenadas de la imagen se encuentra en la esquina superior izquierda. Ello resulta en que todas las coordenadas sean positivas.

Una forma fácil de mejorar la condición de una matriz es trasladar el sistema de referencia de tal manera que el centroide de los puntos sea el origen del sistema de referencia. Es decir, suponiendo que $\mathbf{x}_i = (x_i, y_i)$ describe las coordenadas en la imagen del i-ésimo punto, las coordenadas centralizadas se obtendrían con

$$\mathbf{y}_i = \mathbf{x}_i - \overline{\mathbf{x}}, \text{ y } \overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i, \tag{5.20}$$

y n es el número de puntos. Luego, Hartley[3] explora dos esquemas de escalamiento de coordenadas. En el primero, la idea es que en promedio la distancia al origen sea $\sqrt{2}$. En otras palabras que el punto promedio se localice en $(1,1,1)^T$ en coordenadas homogéneas para cada imagen. Es decir, necesitamos definir una constante k tal que

$$k \cdot \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{y}_i\| = \sqrt{2}, \tag{5.21}$$

de donde se deriva que las coordenadas normalizadas se obtienen aplicando

$$\mathbf{y}_i^n = k \cdot \mathbf{y}_i = \sqrt{2}n\frac{\mathbf{y}_i}{\sum_{i=1}^{n}\|\mathbf{y}_i\|}, \tag{5.22}$$

o en otras palabras, si $\mathbf{u}$ toma la forma $\mathbf{u} = \begin{pmatrix} \mathbf{y}_i^n \\ 1 \end{pmatrix}$ entonces la transformación canónica $\mathbf{T}$ toma correspondientemente la forma $\mathbf{T} = \text{diag}(k, k, 1)$.

Hartley[3] explora un segundo esquema de escalamiento tal que los momentos de segundo orden (momentos principales) de los datos sean unitarios. Supóngase que tenemos n puntos $\mathbf{u}_i = (u_i, v_i, 1)^T$. Ahora fórmece con ellos la matriz

$$\sum_{i=1}^{n}\mathbf{u}_i\mathbf{u}_i^T = n\mathbf{J}\mathbf{J}^T, \tag{5.23}$$

[2]La propiedad de entrelazado permite calcular relaciones entre los eigenvalores de una matriz simétrica. Así, para una submatriz principal de $r \times r$ correspondiente a una matriz simétrica $\mathbf{A}_n$ de $n \times n$ la siguiente propiedad se cumple

$$\lambda_{r+1}(\mathbf{A}_{r+1}) \leq \lambda_r(\mathbf{A}_r) \leq \lambda_r(\mathbf{A}_{r+1}) \cdots \leq \lambda_2(\mathbf{A}_{r+1}) \leq \lambda_1(\mathbf{A}_r) \leq \lambda_1(\mathbf{A}_{r+1}),$$

donde $\mathbf{A}_r$ es la matriz principal de $r \times r$ de $\mathbf{A}_n$ y λ_r su correspondiente r-eigenvalor más grande.

[3]La condición de una matriz $\mathbf{A}$ puede referirse aproximadamente a la magnitud del cambio de $\mathbf{x}$ con respecto a un cambio en $\mathbf{b}$ en el sistema $\mathbf{A}\mathbf{x} = \mathbf{b}$. En otras ocasiones la condición se define como el cociente entre el máximo eigenvalor y el mínimo eigenvalor de una matriz.

donde $\mathbf{J}$ es una matriz triangular inferior y corresponde a uno de los factores de la descomposición de Cholesky[2][4]. Se sigue que la transformación

$$\sum_{i=1}^{n} \mathbf{J}^{-1}\mathbf{u}_i\mathbf{u}_i^T\mathbf{J}^T = n\mathbf{I}, \tag{5.24}$$

puede ser usada para generar las variables $\hat{\mathbf{u}}_i = \mathbf{J}^{-1}\mathbf{u}_i$, tal que

$$\sum_{i=1}^{n} \hat{\mathbf{u}}_i\hat{\mathbf{u}}_i^T = n\mathbf{I}, \tag{5.25}$$

con lo cual los puntos $\hat{\mathbf{u}}_i$ tienen su centroide en el origen y sus momentos principales son unitarios. En este caso la transformación canónica toma la forma $\mathbf{T} = \mathbf{J}^{-1}$.

Con esta transformación, podemos ahora calcular la matriz $\mathbf{F}'$ con los datos transformados por el operador $\mathbf{T}$ y usando el algoritmo de los ocho puntos. Enseguida, se puede recuperar la matriz fundamental aplicando la transformación

$$\mathbf{F} = \mathbf{T}^T\mathbf{F}'\mathbf{T}. \tag{5.26}$$

5.4. Obtención de la Estructura

De (5.7) la relación entre la matriz esencial y la matriz fundamental está dada por la relación

$$\mathbf{E} = \mathbf{K}^T\mathbf{F}\mathbf{K}', \tag{5.27}$$

y la matriz esencial puede ser descompuesta en su rotación y traslación, mediante la solución de lo que es conocido como el *problema de la orientación relativa*. Una cosa importante es que mientras la rotación puede ser recuperada con exactitud, la traslación tiene una ambigüedad de un factor de escala. Esto es así por la inherente limitación de la proyección perspectiva en donde existe la dual interpretación sobre si las imágenes corresponden a un objeto distante y grande, visto desde cámaras que están separadas entre sí, o se está observando un objeto pequeño y cercano desde cámaras que están muy cercanas una de la otra.

Para obtener la rotación y translación, Hartley y Zisserman[4] proponen una estrategia que utiliza la descomposición en valores singulares de la matriz esencial, como $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$. En esta descomposición, por el rango y normalización de escala de $\mathbf{E}$, la matriz de valores singulares queda definida como $\mathbf{L} = \texttt{diag}(1, 1, 0)$. En su método, Hartley y Zisserman requieren la definición de la matriz $\mathbf{W}$ como

$$\mathbf{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.28}$$

[4]La decomposición de Cholesky de una matriz positiva definida $\mathbf{A}$ tiene la forma $\mathbf{A} = \mathbf{L}\mathbf{L}^*$, donde $\mathbf{L}$ es una matriz triangular inferior y $\mathbf{L}^*$ es la transpuesta conjugada de $\mathbf{L}$[2]. Por su lado, el complejo conjugado de un número tiene igual parte real y parte imaginaria opuesta.

Lo cual resulta en aplicación de la factorización

$$\begin{aligned} \lfloor \mathbf{t}_\times \rfloor &= \mathbf{ULWU}^T, \\ \mathbf{R} &= \mathbf{UW}^{-1}\mathbf{V}^T. \end{aligned} \tag{5.29}$$

La verificación de la matriz $\lfloor \mathbf{t}_\times \rfloor$ puede realizarse expandiendo la matriz $\mathbf{U}$ en sus componentes constitutivos y la realización de los cálculos matriciales. Para verificar $\mathbf{R}$, Hartely y Zisserman[4] proponen la siguiente derivación,

$$\mathbf{E} = \lfloor \mathbf{t}_\times \rfloor \cdot \mathbf{R} = \mathbf{UWLU}^T \cdot \mathbf{UXV}^T = \mathbf{UWL} \cdot \mathbf{XV}^T, \tag{5.30}$$

con lo cual $\mathbf{X}$ debe ser igual a $\mathbf{W}^{-1}$ o equivalentemente, pues $\mathbf{W}$ es una matriz ortonormal, $\mathbf{W}^T$.

Al final, el vector $\mathbf{t}$ recuperado se convierte en vector unitario. Hartley y Zisserman [4] muestran que hay un limitación, pues se tiene de recuperar el signo de la matriz $\mathbf{E}$. Es decir, hay cuatro soluciones posibles: dos por el signo de la traslación $\mathbf{t}$, y dos por el uso de $\mathbf{W}$ ó $\mathbf{W}^{-1}$. Las soluciones se manifiestan en la posición de los puntos tridimensionales con respecto al sistema de referencia de las cámaras. Una solución resulta en que los puntos recuperados se encuentren enfrente de ambas cámaras. Otras dos soluciones resultan en que los puntos se encuentren para una cámara adelante y para otra atrás de sus respectivos sistemas de referencia. En la última solución los puntos se encuentran atrás de los sistemas de referencia. Para escoger la solución forma de $\mathbf{t}$ y $\mathbf{W}$, hay que tratar un par de pixeles correspondientes y verificar cuál es la posición del punto en el mundo con respecto al sistema de referencia de las cámaras.

Una vez obtenidos los valores de la posición relativa de las cámaras, es posible recuperar la posición de los puntos en el mundo $\mathbf{p}$ usando su proyección en la imagen $\mathbf{q}$ mediante la solución de la relación[10]

$$\begin{aligned} \mathbf{q} &= \mathbf{KTp}, \\ \lambda_j \begin{pmatrix} x_j \\ y_j \\ 1 \end{pmatrix} &= \mathbf{K} \begin{pmatrix} r_{11j} & r_{12j} & r_{13j} & t_{xj} \\ r_{21j} & r_{22j} & r_{23j} & t_{yj} \\ r_{31j} & r_{32j} & r_{33j} & t_{zj} \end{pmatrix} \begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix}, \end{aligned} \tag{5.31}$$

donde λ_j refleja el factor de escala, y el índice j se refiere a la cámara j-ésima. Premultiplicando ambos lados de (5.31) por $\mathbf{K}^{-1}$ se tiene

$$\lambda_j \begin{pmatrix} x'_j \\ y'_j \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11j} & r_{12j} & r_{13j} & t_{xj} \\ r_{21j} & r_{22j} & r_{23j} & t_{yj} \\ r_{31j} & r_{32j} & r_{33j} & t_{zj} \end{pmatrix} \begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix}. \tag{5.32}$$

El último renglón puede ser usando para resolver λ_j como

$$\lambda_j = r_{31j}u + r_{32j}v + r_{33j}w + t_{zj}. \tag{5.33}$$

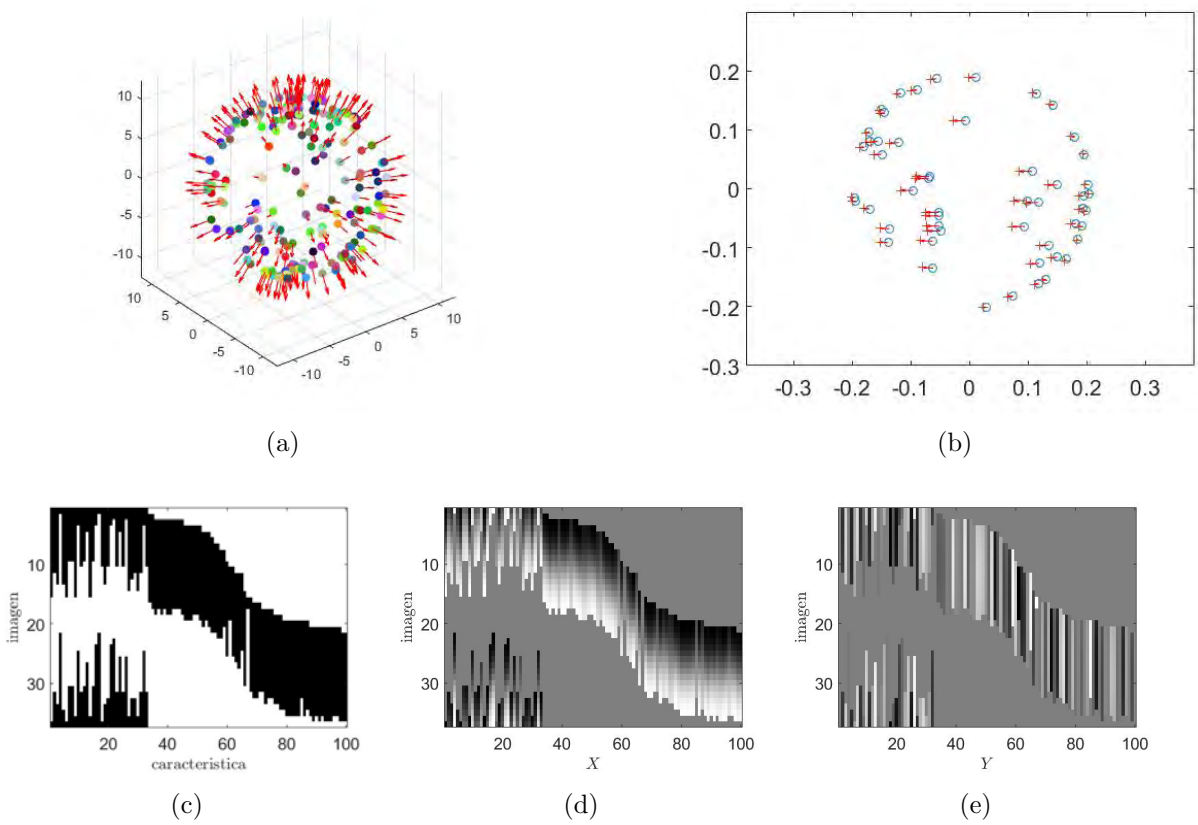


Figura 5.2: Obtención de Características. (a) Imaginemos una esfera que contiene algunos puntos característicos en su superficie, cuya normal se conoce. Luego, la cámara se mueve alrededor de la esfera al tiempo que toma imágenes. El resultado de la búsqueda de su correspondencia se encuentra en la matriz de llenado (c), y las medidas horizontales (d), y verticales (e). Finalmente, su correspondencia se ilustra en (b).

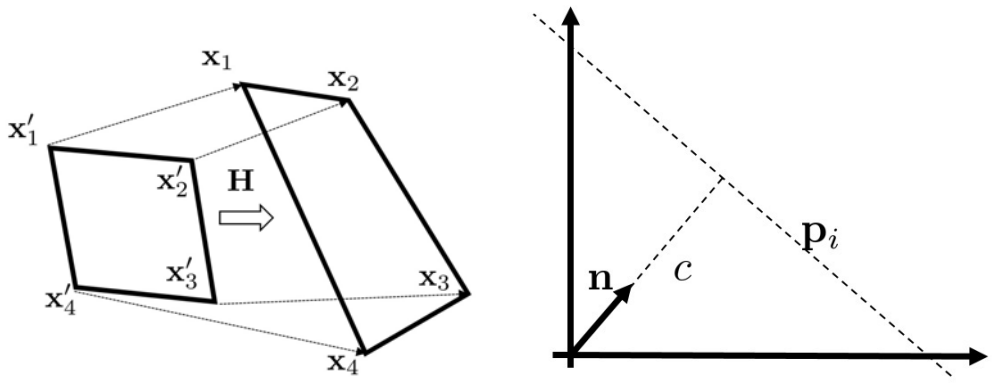
Reemplazando (5.33) en los dos primeros renglones de (5.32) se tiene

$$\begin{pmatrix} (r_{31j}u + r_{32j}v + r_{33j}w + t_{zj})x'_j \\ (r_{31j}u + r_{32j}v + r_{33j}w + t_{zj})y'_j \end{pmatrix} = \begin{pmatrix} r_{11j} & r_{12j} & r_{13j} & t_{xj} \\ r_{21j} & r_{22j} & r_{23j} & t_{yj} \end{pmatrix} \begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix}. \tag{5.34}$$

Realizando las multiplicaciones y resolviendo para $\mathbf{p} = (u, v, w)$, se tiene

$$\begin{pmatrix} r_{31j}x'_j - r_{11j} & r_{32j}x'_j - r_{12j} & r_{33j}x'_j - r_{13j} \\ r_{31j}y'_j - r_{21j} & r_{32j}y'_j - r_{22j} & r_{33j}y'_j - r_{23j} \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} t_{xj} - t_{zj}x'_j \\ t_{yj} - t_{zj}y'_j \end{pmatrix}. \tag{5.35}$$

Mediante la observación del mismo punto por varias cámaras se tienen las restricciones suficientes para resolver el sistema lineal por mínimos cuadrados.



(a) Homografía (b) Parámetros de un plano

Figura 5.3: Objetos planos. Una homografía (a) es una transformación lineal que mapea puntos de un plano a otro plano. Un plano (b) es la figura geométrica para cuyos puntos $\mathbf{p}_i$ su proyección a un vector normal $\mathbf{n}$ al plano resulta en una distancia c

5.5. Objetos Planos

Sorprendentemente, los objetos planos presentan cierta dificultad para ser reconstruidos mediante el esquema que presentamos anteriormente. En la práctica, la matriz $\mathbf{A}$ en (2.44) tiene un rango menor a ocho y las ecuaciones comienzan a producir resultados que no tienen sentido. En esos casos, la rotación y traslación pueden derivarse de la homografía que se forma entre las correspondencias de las imágenes del plano . Sin embargo, las formulaciones, conocidas como descomposición de la homografía, actualmente encontradas[1, 11, 8], resultan numéricamente muy inestables. Una homografía es una transformación lineal que permite proyectar los puntos de un plano en otro plano (ver Figura 5.3(a)). Para el caso de posiciones de pixeles representados en coordenadas homogeneas, donde los puntos se representan en un espacio de tres dimensiones, una homografía requerirá definirse como una matrix $\mathbf{H}$ de 3×3. Dado que que cada punto proporciona dos restricciones, correspondientes a su posición en la imagen, y que la homografía se define hasta un factor de escala, requerimos cuando menos cuatro correspondencias para definir la transformación.

La ecuación de un plano (ver Figura 5.3(b)) se define como

$$\mathbf{n}^T \mathbf{p} = c, \tag{5.36}$$

donde $\mathbf{p}$ representa un punto en el plano, $\mathbf{n}$ es un vector unitario normal al plano y c es la distancia entre el origen del sistema de referencia y el plano. Es decir, un plano es el lugar geométrico de los puntos $\mathbf{p}_i$ cuya proyección con respecto a un vector unitario $\mathbf{n}$ permanecen constantes. Por otro lado, puntos correspondientes, $\mathbf{p}$ y $\mathbf{q}$, pertenecientes a dos posiciones diferentes de la cámara se relacionan mediante la relación

$$\mathbf{q} = \mathbf{R}\mathbf{p} + \mathbf{t}, \tag{5.37}$$

donde $\mathbf{R}$ y $\mathbf{t}$ son respectivamente la rotación y traslación que relacionan los sistemas de referencia locales de las cámaras. Si multiplicamos y dividimos el término de traslación de

(5.37) la siguiente derivación resulta

$$\begin{aligned} \mathbf{q} &= \mathbf{R}\mathbf{p} + \mathbf{t}\frac{c}{c}, \\ &= \mathbf{R}\mathbf{p} + \mathbf{t}\frac{\mathbf{n}^T\mathbf{p}}{c}, \\ &= \left(\mathbf{R} + \frac{\mathbf{t}\mathbf{n}^T}{c}\right)\mathbf{p}, \\ &= \mathbf{H}\mathbf{p}, \end{aligned} \tag{5.38}$$

donde $\mathbf{H}$ es la homografía que modela hasta por un factor de escala, la transformación de los puntos correspondientes de una imagen a la siguiente. Para obtener la homografía se puede utilizar el Algoritmo 8, usado para obtener la matriz fundamental.

Para la extracción de la rotación y translación hay varias alternativas. Sin embargo, todas ellas son complejas y numéricamente sensibles. Por ejemplo, Faugeras[1] y Zhang[11] proponen soluciones numéricas donde la descomposición por valores singulares (de $\mathbf{H}$ en el caso de Faugeras y de $\mathbf{H}\mathbf{H}^T$ en el caso de Zhang) les permiten definir expresiones para la rotación y translación. En el caso de Malis y Vargas[8], ellos obtienen expresiones análiticas para la rotación y traslación partiendo del análisis de los cofactores de la expresión $\mathbf{H}^T\mathbf{H} - \mathbf{I}$. Todos los casos resultan múltiples posibilidades que hay que descartar mediante el uso de restricciones espaciales. Por ejemplo, algunas soluciones se eliminan pues el resultado debe expresar que ambas cámaras se encuentran en el mismo lado del plano. Otras soluciones se eliminan pues los puntos reconstruidos visibles deben estar frente a la cámara. Los lectores son invitados al estudio de las citas para obtener más detalles sobre la obtención de la rotación y la traslación. Con el conocimiento de estos parámetros se puede obtener la estructura del objeto aplicando (5.35).

5.6. *Bundle Adjustment*

Como resultado del proceso anterior, se tiene una aproximación a la estructura tridimensional de la escena, por medio de los puntos $\mathbf{p}_i$, para $i = 1, \ldots, P$. Igualmente, se tiene una aproximación al movimiento de la cámara, a partir de las rotaciones $\mathbf{R}_j^w$ y $\mathbf{t}_j^w$, para $j = 1, \ldots, F$ y $\{w\}$ el sistema de referencia del mundo. Estas aproximaciones han sido obtenidas mediante el análisis entre pares de imágenes. Sin embargo, la estructura puede aproximarse mejor utilizando las observaciones que pudiera haber para cada una de las características sobre todas las imágenes donde hubieran sido observadas. Igualmente, la nueva estimación de la estructura afecta la estimación de la rotación y traslación entre imágenes. Así, el *Bundle Adjustment* se refiere a la estimación de los parámetros de estructura y movimiento utilizando todas las características sobre todas las imágenes mediante alguna medida que permita identificar su valor óptimo. La forma estándar de hacer esto es midiendo la diferencia entre las mediciones hechas y las proyecciones que resultan del valor actual de los

parámetros. Es decir, partiendo de (5.31), la medida de error puede ser definida como

$$e = \sum_{j=1}^{F} \sum_{i=1}^{P} \| \mathbf{q}_{ij} - \hat{\mathbf{q}} \left(\mathbf{K}\mathbf{T}_j \mathbf{p}_i \right) \|^2, \tag{5.39}$$

donde $\mathbf{q}_{ij}$ es un punto en la imagen y $\hat{\mathbf{q}}$ es una función que proyecta al punto $\mathbf{p}_i$ en el plano imagen. Para entender su definición, comencemos con la expansión del término **KTp**, para un cierto punto de una imagen dada. Para ello, utilizamos la definición de **K** en (5.6) y **T** en (5.31). Esto resulta en la expansión

$$\begin{aligned} \mathbf{KTp} &= \begin{pmatrix} f_x & 0 & -c_x \\ 0 & f_y & -c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix}, \\ &= \begin{pmatrix} f_x(ur_{11} + vr_{12} + wr_{13} + t_x) - c_x(ur_{31} + vr_{32} + wr_{33} + t_z) \\ f_y(ur_{21} + vr_{22} + wr_{23} + t_y) - c_y(ur_{31} + vr_{32} + wr_{33} + t_z) \\ ur_{31} + vr_{32} + wr_{33} + t_z \end{pmatrix}. \end{aligned} \tag{5.40}$$

De esta forma, la función $\hat{\mathbf{q}}$ puede ser definida como

$$\hat{\mathbf{q}} \left(\mathbf{KTp} \right) = \begin{pmatrix} \dfrac{f_x(ur_{11} + vr_{12} + wr_{13} + t_x) - c_x}{ur_{31} + vr_{32} + wr_{33} + t_z} \\ \\ \dfrac{f_y(ur_{21} + vr_{22} + wr_{23} + t_y) - c_y}{ur_{31} + vr_{32} + wr_{33} + t_z} \end{pmatrix}. \tag{5.41}$$

La solución de (5.39) requiere la utilización de un algoritmo de optimización no lineal, tal como Levenberg-Marquardt (LM) o Nelder-Mead (NM)[9]. El método de LM es un algoritmo de optimización no-lineal popular. El método utiliza una mezcla de información del gradiente y de la curvatura para determinar la dirección y cantidad de desplazamiento sobre el espacio de solución. Por su parte, el método de NM utliza el concepto de Simplex, o figura geométrica más simple con $n + 1$ vértices en espacios de dimensión n, *i.e.*, un triángulo en dos dimensiones, un tetraedro en tres dimensiones. La idea es expandir o contraer, mediante el desplazamiento de uno o varios vértices durante cada paso, buscando los mínimos.

5.7. Implementación

Prince[10] propone que la reconstrucción tridimensional de las observaciones realizadas requiere la solución de los siguientes pasos:

1. Calcular la posición de puntos característicos, por ejemplo mediante la utilización del detector de Lucas-Kanade[7] o detector de SIFT [6] (ver Figura 5.2).

2. Encontrar la correspondencia entre los puntos característicos.

3. Calcular la matriz fundamental **F** utilizando la versión normalizada del algoritmo de los ocho puntos [3].

4. Refinamiento de correspondencias que no concuerdan con la matriz fundamental. Esto se hace regularmente eliminando correspondencias que se encuentran muy lejos de la línea epipolar o muy desviadas con respecto a la amplitud del campo visual.

5. Estimar la matriz esencial $\mathbf{E}$ mediante (5.27).

6. Descomponer la matriz esencial en la rotación $\mathbf{R}$ y traslación $\mathbf{t}$ relativa entre las posiciones de las cámaras usando (5.29).

7. Estimar los puntos en tres dimensiones usando la solución por mínimos cuadrados del sistema expresado en (5.39).

En nuestro caso, hemos desarrollado el ejemplo que se ilustra en la Figura 5.2. Supóngase que un conjunto de puntos son desplegados en la superficie de una esfera centrada en el origen, definida por las relaciones

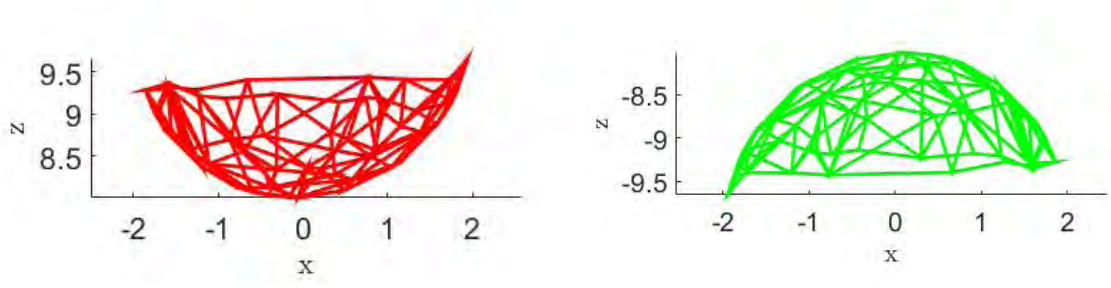
$$\begin{aligned} u &= r\cos\theta\cos\alpha, \\ v &= r\cos\theta\sin\alpha, \\ w &= r\sin\theta, \end{aligned} \tag{5.42}$$

donde $-\pi/2 \leq \theta \leq \pi/2$ corresponde a la elevación y $0 \leq \alpha \leq 2\pi$ corresponde al azimuth. Las normales son fáciles de definir pues corresponden a la línea que va del centro de la esfera a cada punto, normalizadas por su magnitud. Luego se simula una cámara donde se proyectan los puntos de la esfera. Enseguida, se encuentra la correspondencia entre los puntos para imágenes sucesivas. Esto proporciona la matriz de llenado $\mathcal{F}_{F\times P}$ y de medidas $\mathcal{W}_{2F\times P}$, donde F es el número de imágenes y P el número de características.

En el Algoritmo 9 se describen los pasos para realizar la recuperación de la rotación y traslación entre imágenes. En particular, esto requiere la obtención de la matriz fundamental $\mathbf{F}$. El conjunto de pasos que hay que seguir para ello se ilustra en el Algoritmo 8. En adición, la extracción de la rotación y traslación a partir de la matriz esencial genera ocho candidatos. La selección de la solución a partir de los candidatos se detalla en el Algoritmo 10. Algunos ejemplos de la reconstrucción tridimensional se ilustran en la Figura 5.4. La que corresponde a coordenadas de Z positivas es aquella que es correcta.

Sumario

La geometría epipolar establecida para un par de imágenes tomadas con cámaras con campo visual compartido permite el establecimiento de restricciones que facilitan tanto la búsqueda de la correspondencia como la recuperación de información tridimensional. La matriz fundamental y la matriz esencial son componentes primarios de esta aproximación. Sin embargo, su obtención requiere la normalización cuidadosa de los valores de las posiciones de las características en la escena, como lo muestra el algoritmo de los ocho puntos. Una vez obtenida la matriz fundamental, la matriz esencial puede ser calculada utilizando los valores intrínsecos de la cámara. Posteriormente, la rotación y traslación relativas entre observaciones pueden ser recuperadas. La estimación de la posición de puntos en el espacio puede resolverse mediante un sistema lineal. Los datos obtenidos proporcionan un buen valor inicial donde el uso de multitud de vistas permite soluciones más generales y potencialmente más robustas.



(a) Correcta

(b) Incorrecta

Figura 5.4: Reconstrucción usando el algoritmo de los 8 puntos para dos imágenes consecutivas. La solución corresponde a puntos en el mundo, enfrente de la cámara.

Ejercicios

1. ¿Cuál es la ventaja/desventaja de los esquemas de normalización presentados?
2. ¿Dónde se origina la matriz $\mathbf{W}$ en (5.28)?
3. ¿Cuál es la interpretación física de las soluciones posibles para $\mathbf{R} = \mathbf{R}\|\mathbf{R}\|$ en el algoritmo 10?
4. Programar el Algoritmo 9 de reconstrucción de los Ocho Puntos.

Bibliografía

[1] Faugeras, Olivier y Francis Lustman: *Structure from Motion in a Piecewise Planar Wnvironment*. International Journal of Pattern Recognition and Artificial Intelligence, 2(03):485–508, 1988.

[2] Golub, Gene y Charles Van Loan: *Matrix Computations*, volumen 3. JHU Press, 2012.

[3] Hartley, Richard: *In Defense of the Eight-Point Algorithm*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(6):580–593, 1997.

[4] Hartley, Richard y Andrew Zisserman: *Multiple View Geometry in Computer Vision*. Cambridge university press, 2003.

[5] Longuet-Higgins, Christopher: *A Computer Algorithm for Reconstructing a Scene from Two Projections*. Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, páginas 61–62, 1987.

[6] Lowe, David: *Distinctive Image Features from Scale-Invariant Keypoints*. International Journal of Computer Vision, 60(2):91–110, 2004.

[7] Lucas, Bruce y Takeo Kanade: *An Iterative Image Registration Technique with an Application to Stereo Vision*. En *International Joint Conferences on Artificial Intelligence*, volumen 81, páginas 674–679, 1981.

[8] Malis, Ezio y Manuel Vargas: *Deeper Understanding of the Homography Decomposition for Vision-based Control*. Tesis de Doctorado, INRIA, 2007.

[9] Press, William: *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007.

[10] Prince, Simon: *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, 2012.

[11] Zhang, Zhongfei y Allen Hanson: *3D Reconstruction based on Homography Mapping*. Proceedings of ARPA, páginas 1007–1012, 1996.

CAPÍTULO 6

Imágenes de Profundidad

Existen diferentes formas mediante las cuales puede realizarse una reconstrucción tridimensional virtual a partir de imágenes usando Visión por Computadora. Una de ellas es mediante estructura a partir de movimiento[8]. Por ejemplo, una vez obtenido un conjunto de observaciones, uno podría aplicar factorización matricial. Hemos visto que en este caso, se utilizan las suposiciones de proyección ortográfica de los objetos y que los objetos constituyen el centro del sistema de referencia. En otra aproximación, tenemos las técnicas basadas en el establecimiento de una geometría epipolar entre un conjunto de observaciones, la obtención de las matrices que relacionan geométricamente las cámaras y la búsqueda de la correspondencia entre los puntos característicos[4]. Sin embargo, aunque los desarrollos relacionados con estas tendencias han sido considerables, aún queda camino por avanzar para su aplicación general y en tiempo real. En esta parte del documento, tomamos nota del reciente uso de imágenes de profundidad de muy buena calidad que pueden obtenerse con las cámaras disponibles en el mercado (ver Figura 6.1).

En este capítulo, enfocamos nuestra atención al uso de técnicas para la reconstrucción tridimensional de escenarios mediante el denominado *mapeo y localización simultanea* (SLAM). En particular, estudiamos el método desarrollado por Newcombe *et al.*[6] con el cual se obtienen reconstrucciones volumétricas de escenarios del tamaño de una habitación en tiempo real usando imágenes de profundidad. Dos de las características más sobresalientes de este método es que la correspondencia entre la nueva observación y las anteriores se hace mediante el algoritmo llamado *Iterative Closest Point* (ICP)[1], y que esta correspondencia se hace entre la observación actual y el modelo completo y denso que hasta ese momento se ha elaborado.

6.1. Calcular Vértices

El método que proponen Newcombe *et al.*[6] tiene etapas para la medición de las superficies, la actualización de la reconstrucción, la predicción de la forma de la superficie y la estimación de la posición del sensor (ver Figura 6.2). En su método, la posición de la cámara en el tiempo k, que define un sistema de referencia denominado $\{k\}$, se representa en el sistema de referencia global $\{g\}$ con la matriz

$$\mathbf{T}_k^g = \begin{pmatrix} \mathbf{R}_k^g & \mathbf{t}_k^g \\ \mathbf{0}^T & 1 \end{pmatrix}, \tag{6.1}$$

(a) (b)

Figura 6.1: Imágenes de color y profundidad tomadas con la cámara Kinect 2 de Microsoft.

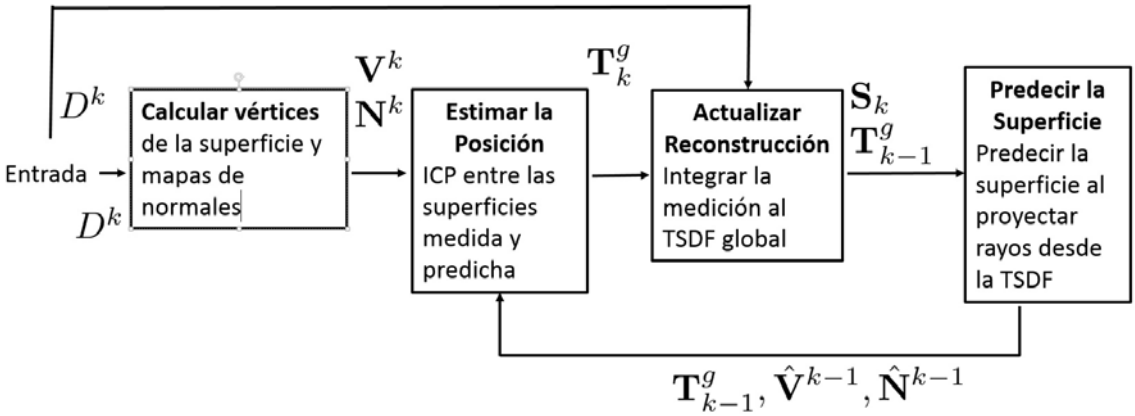


Figura 6.2: Diagrama de funcionamiento del algoritmo de KinectFusion basado en [6].

donde $\mathbf{R}_k^g \in \mathbb{SO}_3$[1], es la rotación del sistema de referencia de la cámara con respecto al sistema de referencia global, y $\mathbf{t}_k^g \in \mathbb{R}^3$ es la traslación que separa ambos sistemas de referencia expresados en el global. Bajo esta notación, un punto $\mathbf{p}^k \in \mathbb{R}^3$ en el sistema de referencia de la cámara en el cuadro $\{k\}$ puede ser representado en el sistema de referencia global mediante la operación $\mathbf{p}^g = \mathbf{T}_k^g \mathbf{p}^k$. Por su lado, la matriz de la cámara $\mathbf{K}$ transforma puntos en el plano de la cámara a pixeles de la imagen. Asimismo, la función $\mathbf{q} = \pi(\mathbf{p})$ proyecta $\mathbf{p} \in \mathbb{R}^3 = (x, y, z)^T$ a puntos $\mathbf{q} \in \mathbb{R}^2 = (x/z, y/z)^T$ en la imagen. En la notación de Newcombe *et al.*[6], la versión homogénea de un vector es $\dot{\mathbf{u}} = \begin{pmatrix} \mathbf{u} \\ 1 \end{pmatrix}$.

En el tiempo k, los puntos en el escenario $\mathbf{p}^k$ pueden ser representados a partir de las

[1]El subgrupo de matrices ortogonales con determinante uno es llamado grupo ortogonal especial $\mathbb{SO}_3$.

observaciones hechas en coordenadas $\mathbf{u}$ de la imagen mediante la transformación

$$\begin{aligned}\mathbf{p}^k &= R^k(\mathbf{u})\mathbf{K}^{-1}\dot{\mathbf{u}}, \\ &= R^k(\mathbf{u}) \begin{pmatrix} 1/f_x & 0 & -c_x/f_x \\ 0 & 1/f_y & -c_y/f_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_x \\ u_y \\ 1 \end{pmatrix}, \\ &= R^k(\mathbf{u}) \begin{pmatrix} (u_x - c_x)/f_x \\ (u_y - c_y)/f_y \\ 1 \end{pmatrix},\end{aligned} \tag{6.2}$$

donde $R_k(\mathbf{u}) \in \mathbb{R}$ es la distancia de profundidad para el pixel $\mathbf{u} = (u, v)^T$. Enseguida, con la finalidad de reducir el ruido presente en las observaciones, sin afectar los bordes entre superficies u objetos con diferente profundidad, se aplica un filtro bilateral, propuesto por Tomasi y Manduchi[9] (ver Figure6.3), al mapa de profundidad, obteniendo con ello

$$D^k(\mathbf{u}) = \frac{1}{W_\mathbf{p}} \sum_{\mathbf{q} \in \mathbb{R}^2} \mathcal{N}\left(\|\mathbf{u} - \mathbf{q}\|_2, \sigma_s\right) \mathcal{N}\left(\|R^k(\mathbf{u}) - R^k(\mathbf{q})\|_2, \sigma_r\right) R^k(\mathbf{q}), \tag{6.3}$$

donde $\mathcal{N}(t, \sigma) = \exp(-0.5t^2/\sigma^2)$ corresponde a una distribución normal y $W_\mathbf{p}$ es una constante de normalización. Estos valores filtrados son proyectados al sistema de referencia del sensor para obtener el mapa de vértices $\mathbf{V}^k$ utilizando la expresión

$$\mathbf{V}^k(\mathbf{u}) = D^k(\mathbf{u})\mathbf{K}^{-1}\dot{\mathbf{u}}. \tag{6.4}$$

Enseguida se calcula, mediante el producto cruz en una pequeña vecindad, la dirección de la normal en cada posición de la imagen de profundidad, tal que (ver Figura 6.4)

$$\mathbf{N}^k(\mathbf{u}) = v\left[\left(\mathbf{V}^k(u+1, v) - \mathbf{V}^k(u, v)\right) \times \left(\mathbf{V}^k(u, v+1) - \mathbf{V}^k(u, v)\right)\right], \tag{6.5}$$

donde $v[\mathbf{x}] = \mathbf{x}/\|\mathbf{x}\|_2$ es una expresión para darle magnitud uno al vector normal. Finalmente, la representación de la imagen de vértices y vectores normales en el sistema de referencia global $\{g\}$ está dado por la siguiente transformación de coordenadas

$$\mathbf{V}_k^g(\mathbf{u}) = \mathbf{T}_k^g \dot{\mathbf{V}}^k(\mathbf{u}), \tag{6.6}$$

y, de manera equivalente, las normales son mapeadas al sistema de referencia global mediante la expresión

$$\mathbf{N}_k^g(\mathbf{u}) = \mathbf{R}_k^g \mathbf{N}^k(\mathbf{u}). \tag{6.7}$$

6.2. Estimar la Posición

En principio se tiene la nube de puntos correspondiente a la forma que se está reconstruyendo $\{\mathbf{q}_i^g\}$ y la nube de puntos de la nueva imagen $\{\mathbf{q}_i^k\}$. Uno de los problemas que surge es encontrar la correspondencia entra ambas. En principio, una rotación $\mathbf{R}_k^g$ y una traslación

Llamada : $\langle \mathbf{R}, \mathbf{t} \rangle \leftarrow$ `ICP` $(\hat{\mathbf{V}}^{k-1}, \hat{\mathbf{N}}^{k-1}, \mathbf{V}^{k})$
Entradas: Los puntos $\hat{\mathbf{V}}^{k-1}_{3\times n}$ y $\mathbf{V}^{k}_{3\times n'}$ correspondientes a los vértices modelo y los datos, y $\hat{\mathbf{N}}^{k-1}_{3\times n}$ corresponde al estimado de las normales del modelo.
Salidas : Rotación $\mathbf{R}$ y traslación $\mathbf{t}$ que permiten la alineación entre los conjuntos de puntos.

`// Definir valores para el número de iteraciones,` κ`, el número de distancias medias` α`, y el número de puntos para seleccionar,` m

$\kappa \leftarrow \ldots; \alpha \leftarrow \ldots; m \leftarrow \ldots; \mathbf{U} \leftarrow \mathbf{V}^{k}$;

`// Iterar un número` κ `de veces`

for $k \leftarrow 1$ **to** κ **do**

`// Seleccionar` m `puntos al azar con índices` $\mathbf{i}$

$\left\langle \mathbf{V}'_{3\times m}, \mathbf{i}_{m\times 1} \right\rangle \leftarrow$ `Seleccionar_Puntos` $(\mathbf{U}, m)$;

`// Definir su correspondencia en términos del punto más cercano en el modelo. Aquí` $\mathbf{P} = \{\mathbf{p}_i\}, \mathbf{Q} = \{\mathbf{q}_i\}$

$\langle \mathbf{P}_{3\times m}, \mathbf{Q}_{3\times m} \rangle \leftarrow$ `Encontrar_Correspondencia` $(\hat{\mathbf{V}}^{k-1}, \mathbf{V}')$;

`// Rechazar correspondencias cuya distancia sea mayor a un cierto número de veces` α `la media de las distancias. Aquí` $\mathbf{P}' = \{\mathbf{p}'_i\}, \mathbf{Q}' = \{\mathbf{q}'_i\}$

$\left\langle \mathbf{P}'_{3\times m'}, \mathbf{Q}'_{3\times m'}, \mathbf{i}' \right\rangle \leftarrow$ `Refinar_Correspondencias` $(\mathbf{P}, \mathbf{Q}, \alpha, \mathbf{i})$;

`// Seleccionar las normales que participarán en la minimización. Aquí` $\mathbf{N}' = \{\mathbf{n}'_i\}$

$\mathbf{N}' \leftarrow$ `Seleccionar_Normales`$(\mathbf{N}^{k}, \mathbf{i}')$;

`// Encontrar la rotación` $\mathbf{R}$ `y traslación` $\mathbf{t}$ `minimizando`

$$E = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i} \left((\mathbf{R}\mathbf{p}'_i + \mathbf{t} - \mathbf{q}'_i)^T \mathbf{n}'_i \right)^2;$$

`// Modificar puntos de los datos`

$\mathbf{U} \leftarrow$`Transformar_Puntos` $(\mathbf{V}^{k}, \mathbf{R}, \mathbf{t})$;

end

Algorithm 11: ICP. El algoritmo de ICP busca calcular la rotación y traslación que alinean dos nubes de puntos.

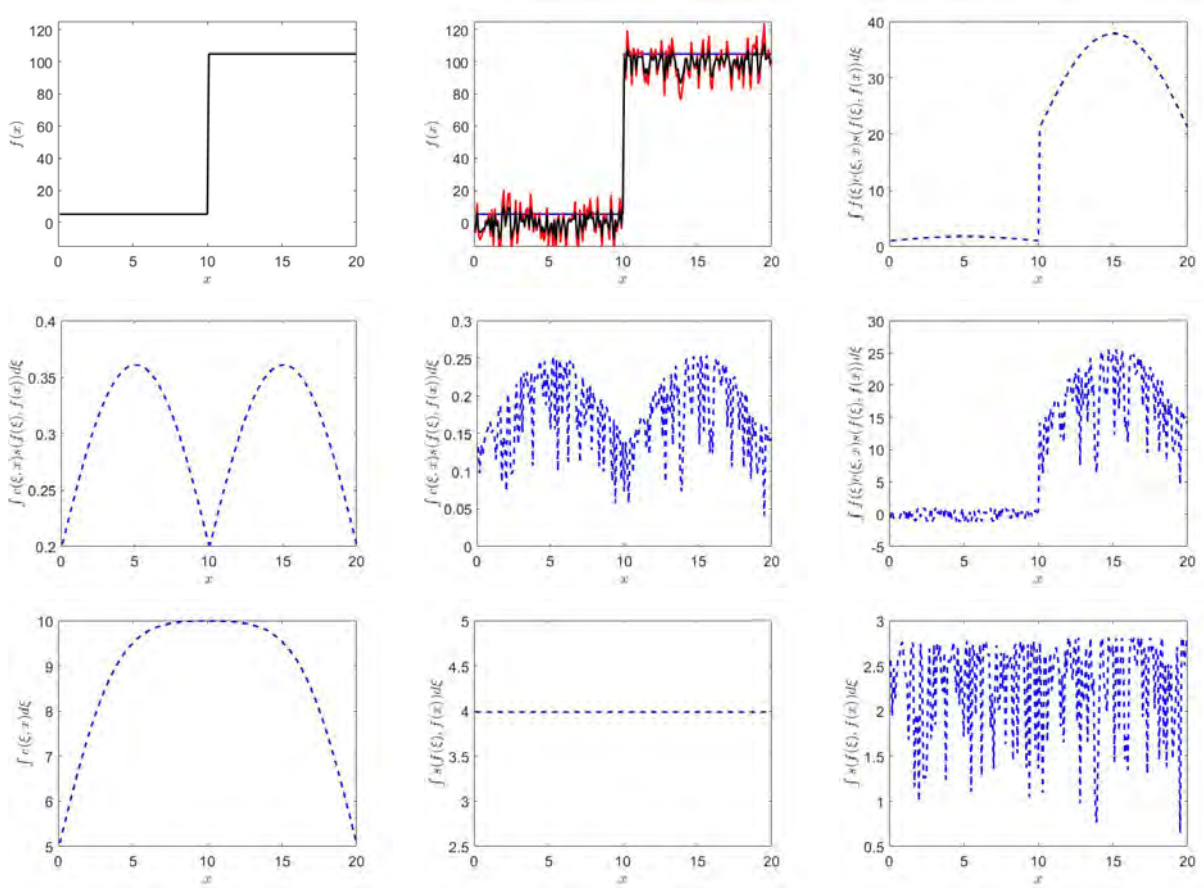


Figura 6.3: El filtro bilateral opera tanto en espacio como en rango permitiendo eliminar el ruido manteniendo los contornos. Aquí ilustramos su funcionamiento sobre un escalón sin (a) y con (b) ruido. En (b) se presenta la señal original, con ruido y filtrada. En (c) y (f) se presenta la señal sin normalizar, sin y con ruido. En (c) y (d) se presenta el factor de normalización. En (g) se presenta el término que establece la vecindad en el dominio. En (h) e (i) se presenta el factor del rango, sin y con ruido. El ruido es Gaussiano con $\sigma = 10$. La desviación estándar del filtro espacial es $\sigma_d = 3$ y del filtro de rango es $\sigma_r = 10$.

$\mathbf{t}_k^g$ son parte de esta transformación. En los pasados capítulos, los autores buscaban puntos sobresalientes y la transformación era calculada entre estos puntos. Aquí exploramos un tipo particular de solución en donde todos los puntos son utilizados, en una técnica conocida como ICP (*Iterative Closest Point*) (ver Algoritmo 11).

En ICP se supone que hay correspondencia entre los puntos más cercanos, rechazando aquellos cuya distancia sea superior a un umbral α. Con los puntos restantes se obtiene una solución cerrada, por ejemplo minimizando una función del tipo

$$E = \sum_i \left\| \mathbf{R}_k^g \mathbf{q}_i^k + \mathbf{t}^g - \mathbf{q}_i^g \right\|^2. \tag{6.8}$$

La variante de ICP que utiliza KinectFusion toma en cuenta la distancia de puntos a planos. Algunos autores enfatizan que esto permite el desplazamiento entre superficies planas. En

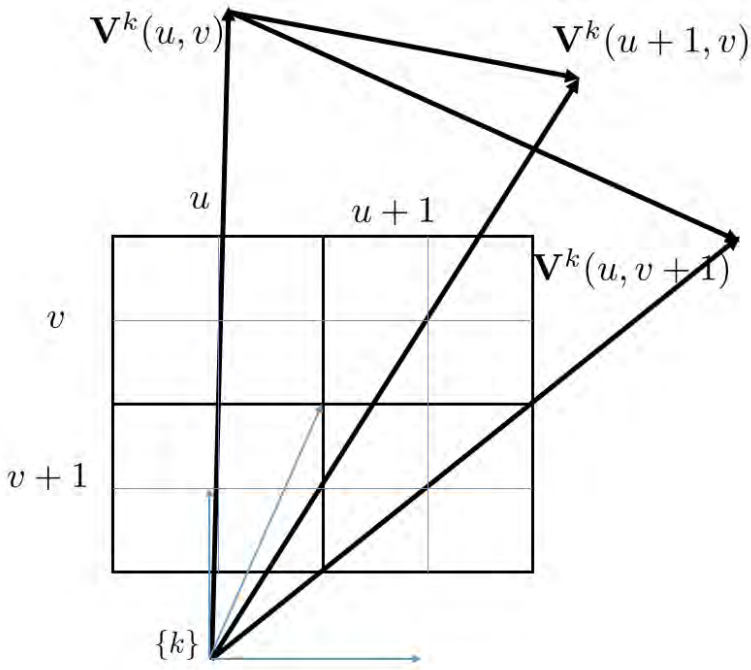


Figura 6.4: La normal local se obtiene con base en el análisis de una pequeña vecindad alrededor del punto de interés. Supóngase que se tiene una retícula entre las columnas u y $u+1$ y las columnas v y $v+1$. Las medidas de distancia que salen del centro de proyección de la cámara hasta las superficies de los objetos señalan los vértices $\mathbf{V}^k(\mathbf{x})$. La normal a la superficie en $\mathbf{x}^T = (u, v)$ se calcula usando (6.5).

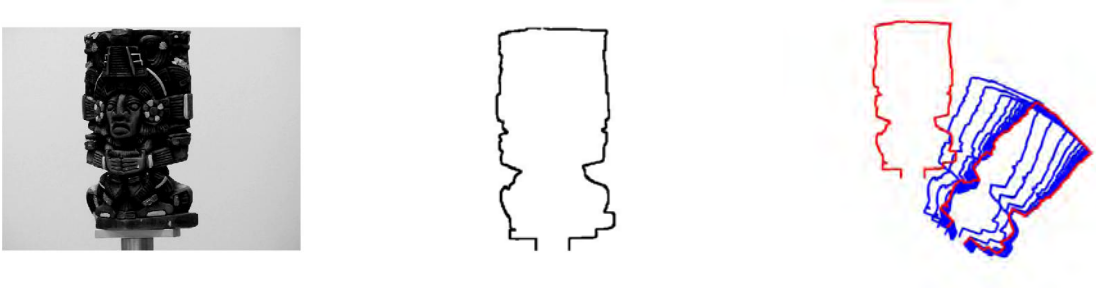
este caso, se busca minimizar la proyección del vector residual de la transformación con la normal de la superficie, es decir

$$E_{\mathbf{n}} = \sum_i \left\| (\mathbf{R}_k^g \mathbf{q}_i^k + \mathbf{t}^g - \mathbf{q}_i^g)^T \mathbf{n}^g \right\|^2 . \tag{6.9}$$

6.3. Actualizar Reconstrucción

En el trabajo de Newcombe *et al.*[6], cada nueva imagen se fusiona en la reconstrucción global que hasta ese momento se haya obtenido. Para lograrlo, ellos usan una versión para tres dimensiones de la función de distancia con signo (SDF) propuesta por Curless y Levoy [3], que Newcombe *et al.* denominan *función de distancia truncada con signo* (TSDF). En la SDF los valores de la función se asumen positivos para la superficie visible, creciendo hacia el espacio libre, cero para la superficie de interfase, y negativos para los lados no visibles. Para Newcombe *et al.*[6] los valores de la función se truncan en la superficie, tomando en cuenta la incertidumbre en la medición de la profundidad, $\pm\mu$.

La TSDF $\mathbf{S}_k(\mathbf{p}^g)$ representa la fusión de los cuadros hasta la k-ésima imagen de rango, donde $\mathbf{p}^g \in \mathbb{R}^3$. La TSDF $\mathbf{S}_k$ tiene dos componentes: el valor de la distancia truncada con



(a) (b) (c)

Figura 6.5: Ilustración del funcionamiento de ICP. En (b) se muestra el contorno externo de la figura en (a). Supóngase que este contorno es rotado y trasladado (líneas rojas), como se muestra en (c). ICP busca iterativamente la rotación y traslación que hace coincidir los puntos (ver Algoritmo 11).

signo $F_k(\mathbf{p}^g)$ y un peso $W_k(\mathbf{p}^g)$, tal que

$$\mathbf{S}_k(\mathbf{p}^g) \mapsto [F_k(\mathbf{p}^g), W_k(\mathbf{p}^g)]. \tag{6.10}$$

Los términos de la TSDF se construyen a partir de las observaciones que se hacen en cuadros individuales. Así, para un mapa de profundidad D^k con una transformación $\mathbf{T}_k^g$ con respecto al sistema de referencia global, F_{D^k} se calcula, en un punto $\mathbf{p}^g$ que se encuentra en el marco de referencia global $\{g\}$, como (ver Figura 6.6)

$$F_{D^k}(\mathbf{p}^g) = \Psi\left(\|\mathbf{t}_k^g - \mathbf{p}^g\|_2 - \lambda D^k(\mathbf{x})\right), \tag{6.11}$$

donde

$$\mathbf{x} = \left\lfloor \pi\left(\mathbf{K}\mathbf{T}_g^k \mathbf{p}^g\right) \right\rfloor, \tag{6.12}$$

corresponde a la posición, en valores enteros, del pixel donde se proyecta el punto $\mathbf{p}^g$, el cual se encuentra en el sistema de referencia global. Por su parte, y

$$\Psi(\eta) = \begin{cases} \text{mín}\left(1, \frac{\eta}{\mu}\right) \text{sgn}(\eta) & \text{ssi } \eta \geq -\mu, \\ \text{null} & \text{otro caso.} \end{cases} \tag{6.13}$$

es la función que permite truncar los valores de distancia con signo (ver Figura 6.7). Es decir, mientras que η sea menor que $\|\mu\|$, el valor de Ψ es η/μ. De otra forma, Ψ es ± 1, dependiendo del valor del signo de η. Por su lado, el peso asociado $W_{D^k}(\mathbf{p}^g)$ es directamente proporcional al ángulo θ entre la dirección del rayo del pixel y la normal a la superficie medida en el sistema de referencia local, e inversamente proporcional a la profundidad a la que se encuentra el objeto con respecto a la cámara. Es decir, está definida como

$$W_{D^k}(\mathbf{p}^g) = \cos\theta / D^k(\mathbf{x}). \tag{6.14}$$

La fusión de los mapas de profundidad se realiza con el promedio pesado de las TSDF individuales. Esto se logra obteniendo la TSDF F que minimiza la función Es decir, sea $F_{D^k}(\mathbf{p}^g)$ los valores de rango en cada imagen y $W_{D^k}(\mathbf{p}^g)$ el valor de peso asociado a cada

Llamada : $\langle F_k, W_k \rangle \leftarrow$ `Actualizar_Reconstrucción` $(\mathbf{T}_k^g, D^k, F_{k-1}, W_{k-1})$
Entradas: La posición del sistema de referencia de la cámara en el sistema de referencia global $\mathbf{T}_k^g$, las medidas de profundidad para la imagen actual R^k, el valor actual de la distancia truncada con signo (TSDF) F_{k-1} y su peso asociado W_{k-1}
Salidas : Los valores actualizados de F_k y su peso asociado W_k

`// Calcular la distancia truncada con signa para el actual punto de vista, usando (6.13)`
$F_{D^k} \leftarrow$ `Calcular_Distancia_Truncada_Local` $(\mathbf{T}_k^g, \mathbf{V}^k, D^k)$;
`// Calcular la ponderación para el actual punto de vista, usando (6.14)`
$W_{D^k} \leftarrow$ `Calcular_Ponderación_Local` $(\mathbf{T}_k^g, \mathbf{V}^k, D^k)$;
`// Obtener los puntos visibles desde la imagen actual`
$\mathbf{p}^k = D^k(\mathbf{u})\mathbf{K}^{-1}\dot{\mathbf{u}}$;
`// Obtener los puntos en el sistema de referencia global`
$\mathbf{p}^g \leftarrow \mathbf{T}_k^g \mathbf{p}^k$;
`// Itera para todos los puntos visibles desde la imagen actual`
for $\mathbf{p}^g \in \mathbf{P}$ **do**
  `// Actualizar los valores de la distancia truncada con signo y las ponderaciones`
  $F_k(\mathbf{p}^g) \leftarrow \dfrac{W_{k-1}(\mathbf{p}^g)F_{k-1}(\mathbf{p}^g) + W_{D^k}(\mathbf{p}^g)F_{D^k}(\mathbf{p}^g)}{W_{k-1}(\mathbf{p}^g) + W_{D^k}(\mathbf{p}^g)}$;
  $W_k(\mathbf{p}^g) \leftarrow W_{k-1}(\mathbf{p}^g) + W_{D^k}(\mathbf{p}^g)$;
end

Algorithm 12: Actualización de la Reconstrucción. La distancia con signo truncada y su ponderación son recalculados después de cada toma de imagen.

medición. Durante la operación, el siguiente esquema (detallado en [3]) permite calcular iterativamente los promedios ponderados como

$$\begin{aligned} F_k(\mathbf{p}) &= \frac{W_{k-1}(\mathbf{p})F_{k-1}(\mathbf{p}) + W_{D^k}(\mathbf{p})F_{D^k}(\mathbf{p})}{W_{k-1}(\mathbf{p}) + W_{D^k}(\mathbf{p})}, \\ W_k(\mathbf{p}) &= W_{k-1}(\mathbf{p}) + W_{D^k}(\mathbf{p}). \end{aligned} \tag{6.15}$$

6.4. Predecir la Superficie

La representación del algoritmo de Newcombe *et al.*[6] permite la creación de vistas virtuales. Esto se puede lograr utilizando la información de la superficie reconstruida usando los valores para los cuales $F_k = 0$ y el estimado actual de la posición de la cámara con respecto al mundo, $\mathbf{T}_k^g$, tal que

$$\mathbf{p}^k = \mathbf{T}_g^k \mathbf{K}^{-1}\dot{\mathbf{u}}. \tag{6.16}$$

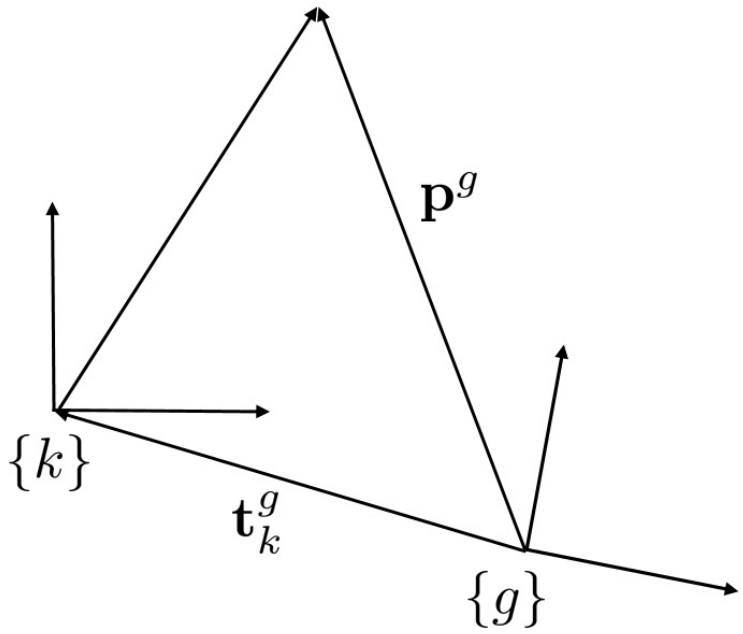


Figura 6.6: La función de distancia con signo truncada (TSDF) es expresada como la diferencia entre la medición hecha por el sensor en el cuadro $\{k\}$ y el estimado de esa distancia en el sistema de referencia del mundo $\|\mathbf{t}_k^g - \mathbf{p}^g\|$ en el mundo $\{g\}$. Esta distancia se compara contra $\lambda D^k(\mathbf{x})$.

Por su parte, el mapa de normales correspondiente, $\hat{\mathbf{N}}^k$, para un pixel $\mathbf{u}$, que representa la proyección de un punto $\mathbf{p}^g$, puede ser calculada a partir de la función de distancia truncada con signo usando

$$\mathbf{R}_k^g \hat{\mathbf{N}}^k = \hat{\mathbf{N}}_k^g(\mathbf{u}) = \nabla F(\mathbf{p}), \text{ donde } \nabla F = \left[\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right]^T. \tag{6.17}$$

Enseguida, Newcombe *et al.*[6] utilizan un esquema de *ray tracing*[7] para determinar las superficies visibles de la siguiente manera. Sea $\mathbf{p}^k$ el punto en la dirección de la visión y $t\mathbf{R}_k^g\mathbf{p}^k$ su representación en el sistema de referencia global. La idea es encontrar el valor t que genera una intersección con la superficie del objeto. Estos puntos son visibles siempre que su producto punto con la normal de la superficie, definido por ∇F, sea negativo y no haya una superficie que los oculte. Parker *et al.*[7] proponen un algoritmo para determinar eficientemente las voxeles que no contienen superficies, calcular la superficie cuando el rayo de visión intersecta con ella y proporcionar el sombreado que corresponda a la superficie. Como resultado, la superficie predicha se almacena como un mapa de vértices $\hat{\mathbf{V}}^k$.

6.5. Estimación de la Posición del Sensor

El problema de localizar la cámara consiste en estimar la transformación $\mathbf{T}_k^g \in \mathbb{SE}_3$[2], que incluye una rotación y una traslación. Newcombe *et al.*[6] usan extensiones del algoritmo de

[2] $\mathbb{SE}_3$ es el grupo de las transformaciones rígidas.

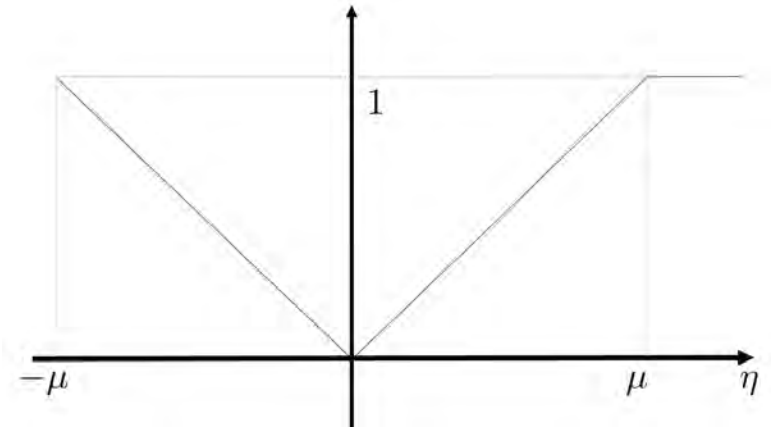


Figura 6.7: Ilustración de la función $\Psi(\eta)$. La función sólo está definida para valores de $\eta \geq -\mu$. Para valores mayores a μ la función vale 1. Entre $-\mu$ y μ la función vale η/μ.

asociación de datos de Blais y Levine[1] para obtener correspondencia y del algoritmo de métrica punto-plano de Chen y Medioni[2] para la optimización de la posición. Newcombe *et al.* alinean la medición de superficie en el cuadro actual $(\mathbf{V}^k, \mathbf{N}^k)$ contra la predicción del modelo que se ha hecho con la información hasta el cuadro anterior $(\hat{\mathbf{V}}^g_{k-1}, \hat{\mathbf{N}}^g_{k-1})$. Su formulación de energía para el estimador de la posición de la cámara $\mathbf{T}^g_k$ guarda similaridad con la función objetivo para obtener la rotación y translación en el algoritmo de ICP (ver (6.9)) y está definida como

$$E(\mathbf{T}^g_k) = \sum_{\substack{\mathbf{u} \in \mathcal{U} \\ \Omega_k(\mathbf{u}) \neq \text{null}}} \left\| \left(\mathbf{T}^g_k \dot{\mathbf{V}}^k(\mathbf{u}) - \hat{\mathbf{V}}^g_{k-1}(\mathbf{u}) \right)^T \hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}}) \right\|_2^2, \tag{6.18}$$

donde $\mathcal{U}$ es el conjunto de correspondencias y la condición $\Omega_k(\mathbf{u}) \neq \text{null}$ se utiliza para rechazar correspondencias muy malas

$$\Omega_k(\mathbf{u}) \neq \text{null ssi} \begin{cases} \mathbf{M}^k(\mathbf{u}) & = & 1, \text{y} \\ \| \tilde{\mathbf{T}}^{gz}_k \dot{\mathbf{V}}^k(\mathbf{u}) - \hat{\mathbf{V}}^g_{k-1}(\hat{\mathbf{u}}) \|_2 & \leq & \epsilon_d, \text{y} \\ \| \left(\mathbf{R}^{gz}_k \mathbf{N}^k(\mathbf{u}) \right)^T \hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}}) & \leq & 1 - \epsilon_\theta, \end{cases} \tag{6.19}$$

donde $\mathbf{M}^k(\mathbf{u}) = 1$ implica que hay una medición de profundidad válida en $\mathbf{u}$, ϵ_d y ϵ_θ son umbrales, y $\tilde{\mathbf{T}}^{gz}_k$ es la actual estimación de la transformación entre el sistema de referencia k y el global (en particular conteniendo una rotación $\mathbf{R}^{gz}_k$).

Al igual que el algoritmo de ICP, la solución se basa en un mecanismo iterativo donde la solución va obteniéndose poco a poco. Para ello, $\tilde{\mathbf{T}}^{gz=0}_k$ se inicializa con la posición anterior $\tilde{\mathbf{T}}^g_{k-1}$. Entonces, una solución iterativa para $\tilde{\mathbf{T}}^{gz}_k$, para $z > 0$ se obtiene minimizando una versión linealizada de (6.18) alrededor del valor previo $\tilde{\mathbf{T}}^{gz-1}_k$, de la siguiente manera.

Supongamos que el movimiento de cuadro a cuadro es tan rápido que permite considerar un ángulo pequeño entre las trasformaciones, tal que (ver Apéndice 6.A)

$$\tilde{\mathbf{T}}^z_{inc} = \left[\tilde{\mathbf{R}}^z|\tilde{\mathbf{t}}^z\right] = \begin{bmatrix} 1 & \alpha & -\gamma & t_x \\ -\alpha & 1 & \beta & t_y \\ \gamma & -\beta & 1 & t_z \end{bmatrix}, \tag{6.20}$$

entonces una actualización de la transformación pueda tener la siguiente forma

$$\tilde{\mathbf{T}}^{gz}_k = \tilde{\mathbf{T}}^z_{inc}\tilde{\mathbf{T}}^{gz-1}_k. \tag{6.21}$$

Si $\tilde{\mathbf{T}}^z_{inc}$ se escribe en base a un vector de parámetros con la forma

$$\mathbf{x} = (\beta, \gamma, \alpha, t_x, t_y, t_z) \in \mathbb{R}^6, \tag{6.22}$$

entonces uno puede formular una forma alternativa de representar (6.18) basado en la siguiente equivalencia

$$\tilde{\mathbf{T}}^{gz}_k\dot{\mathbf{V}}^k(\mathbf{u}) = \tilde{\mathbf{R}}^z\tilde{\mathbf{V}}^g_k(\mathbf{u}) + \tilde{\mathbf{t}}^{gz}_k = \mathbf{G}(\mathbf{u})\mathbf{x} + \tilde{\mathbf{V}}^g_k(\mathbf{u}), \tag{6.23}$$

donde la matriz $\mathbf{G}_{3\times 6}$ se construye con la forma en matriz simétrica-sesgada de $\tilde{\mathbf{V}}^g_k(\mathbf{u})$, tal que

$$\begin{aligned} \mathbf{G}(\mathbf{u}) &= \left[\left[\tilde{\mathbf{V}}^g_k(\mathbf{u})\right]_\times |\mathbf{I}_{3\times 3}\right], \\ &= \begin{pmatrix} 0 & -v_z & v_y & 1 & 0 & 0 \\ v_z & 0 & -v_x & 0 & 1 & 0 \\ -v_y & v_x & 0 & 0 & 0 & 1 \end{pmatrix}. \end{aligned} \tag{6.24}$$

Planteado de esa forma, tenemos que una expresión equivalente para minimizar (6.18) podría ser mediante la expresión

$$\min_{\mathbf{x}\in\mathbb{R}^6} \sum_{\Omega_k(\mathbf{u})\neq \text{null}} \|E\|_2^2, \tag{6.25}$$

donde con los cambios propuestos E puede expresarse como

$$\begin{aligned} E &= \hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}})^T\left(\mathbf{G}(\mathbf{u})\mathbf{x} + \hat{\mathbf{V}}^g_k(\mathbf{u}) - \hat{\mathbf{V}}^g_{k-1}(\mathbf{u})\right), \\ &= \hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}})^T\mathbf{G}(\mathbf{u})\mathbf{x} + \hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}})^T\left(\hat{\mathbf{V}}^g_k(\mathbf{u}) - \hat{\mathbf{V}}^g_{k-1}(\mathbf{u})\right). \end{aligned} \tag{6.26}$$

El mínimo de esta expresión se encuentra cuando se deriva con respecto a $\mathbf{x}$ e iguala a cero. Esto resulta en el sistema

$$\sum_{\Omega_k(\mathbf{u})\neq \text{null}} \left(\mathbf{A}^T\mathbf{A}\right)\mathbf{x} = \sum_{\Omega_k(\mathbf{u})\neq \text{null}} \mathbf{A}^T\mathbf{b}, \tag{6.27}$$

donde

$$\mathbf{A}^T = \mathbf{G}^T(\mathbf{u})\hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}}), \tag{6.28}$$

y

$$\mathbf{b} = \hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}})^T\left(\hat{\mathbf{V}}^g_{k-1}(\mathbf{u}) - \hat{\mathbf{V}}^g_k(\mathbf{u})\right). \tag{6.29}$$

Llamada : $\mathbf{T}_k^g \leftarrow$ `Estimar_Posición` $(\mathbf{T}_{k-1}^g, \hat{\mathbf{N}}_{k-1}^g, \hat{\mathbf{V}}_{k-1}^g, \mathbf{N}^k, \mathbf{V}^k)$

Entradas: La estimación de la posición en el cuadro anterior $\mathbf{T}_{k-1}^g$, las estimaciones de los vértices $\hat{\mathbf{V}}^{k-1}$ y las normales $\hat{\mathbf{N}}^{k-1}$ en el cuadro $k-1$ y las mediciones de los vértices y normales, $\mathbf{V}^k$ y las normales $\mathbf{N}^k$, en la imagen actual.

Salidas : La transformación rígida $\mathbf{T}_k^g$ que relaciona la posición actual de la cámara con el sistema de referencia global.

`// Inicializar el valor de la transformación rígida`

$\mathbf{T}_k^{gz=0} \leftarrow \mathbf{T}_{k-1}^g$;

`// Iterar hasta lograr convergencia`

repeat

  `// Inicializa la transformación`

  $\mathbf{A}' \leftarrow \mathbf{0}; \mathbf{b}' \leftarrow \mathbf{0}$;

  `// Itera para todos los puntos de la imagen válidos, ver (6.19)`

  for $\mathbf{u} \in \mathcal{U}, \Omega_k(\mathbf{u}) \neq null$ **do**

    `// Estimar el valor de los vértices en el sistema de referencia global`

    $\tilde{\mathbf{V}}_k^g(\mathbf{u}) \leftarrow \tilde{\mathbf{T}}_k^{gz-1} \dot{\mathbf{V}}^k(\mathbf{u})$;

    `// Calcular el valor de la función auxiliar` $\mathbf{G}(\mathbf{u})$

    $\mathbf{G}(\mathbf{u}) \leftarrow \left[\left[\tilde{\mathbf{V}}_k^g(\mathbf{u}) \right]_\times | \mathbf{I}_{3\times 3} \right]$;

    `// Calcular las matrices involcradas en la solución`

    $\mathbf{A}^T \leftarrow \mathbf{G}^T(\mathbf{u}) \hat{\mathbf{N}}_{k-1}^g(\hat{\mathbf{u}})$; $\mathbf{b} \leftarrow \hat{\mathbf{N}}_{k-1}^g(\hat{\mathbf{u}})^T \left(\hat{\mathbf{V}}_{k-1}^g(\mathbf{u}) - \hat{\mathbf{V}}_k^g(\mathbf{u}) \right)$;

    `// Acumular la contribución de cada punto`

    $\mathbf{A}' \leftarrow \mathbf{A}' + \mathbf{A}^T\mathbf{A}; \mathbf{b}' \leftarrow \mathbf{b}' + \mathbf{A}^T\mathbf{b}$;

  end

  `// Obtener la solución al sistema lineal`

  $\mathbf{x} \leftarrow \mathbf{A}'^{-1}\mathbf{b}'$;

  `// Obtener el incremento de la transformación rígida a partir de` $\mathbf{x}$

  $$\tilde{\mathbf{T}}_{inc}^z \leftarrow \begin{bmatrix} 1 & \alpha & -\gamma & t_x \\ -\alpha & 1 & \beta & t_y \\ \gamma & -\beta & 1 & t_z \end{bmatrix};$$

  `// Actualizar el valor de la transformación rígida`

  $\tilde{\mathbf{T}}_k^{gz} \leftarrow \tilde{\mathbf{T}}_{inc}^z \tilde{\mathbf{T}}_k^{gz-1}$;

until *convergencia*;

Algorithm 13: Estimación de la Posición de la Cámara. La posición de la cámara $\mathbf{T}_k^g$ se estima de forma iterativa, considerando todos los puntos válidos de la imagen de profundidad.

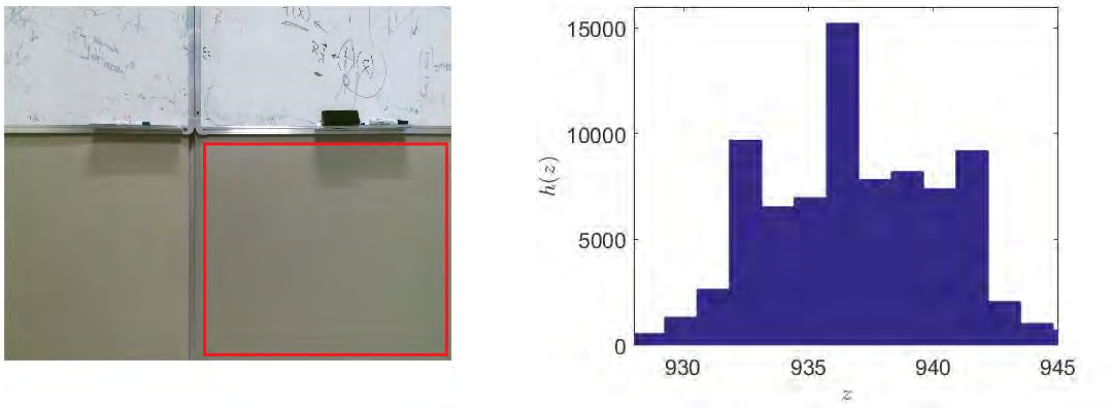
(a) (b)

Figura 6.8: Reconstrucción 3D. En (a) se presenta el escaneo usando la aplicación `3D builder` de Microsoft. En (b) se tiene el resultado de dejar únicamente a la persona sentada en (a), eliminando vértices usando el programa `MeshLab`.

6.6. Implementación

Como hemos visto en este capítulo, las imágenes de profundidad pueden facilitarnos la reconstrucción tridimensional de escenarios y la detección de personas. En particular, la llegada del Kinect ha permitido obtener imágenes de gran calidad a un costo muy bajo (ver Figura6.9). En la Figura 6.8 presentamos un ejemplo del uso de la utilería `3D builder` de Microsoft, que permite el escaneo de objetos y su reconstrucción 3D. En ella, una persona se sienta en una silla de oficina mientras, con el programa `3D Builder` corriendo, se escanea el entorno. El programa requiere Windows 8.1 y el Kinect v2 para funcionar. Una vez obtenida la reconstrucción, el resultado fue editado en `MeshLab` para dejar únicamente a la persona sentada.

Los algoritmos para la alineación de dos nubes de puntos (Algoritmo 11), para la actualización de la reconstrucción (Algoritmo 12) y para la estimación de la posición (Algoritmo 13) pueden ser la base para la implementación del algoritmo de Newcombe *et al.*[6].



(a) (b)

Figura 6.9: Sensores como el Kinect v2 ofrecen alta calidad de medición de profundidad a un precio accesible. Aquí tomamos una imagen de un pizarrón (a. El histograma de distribución de distancias en mm se muestra en (b). El Kinect v2 proporciona la distancia en la dirección de profundidad.

Sumario

En este capítulo hemos revisado los fundamentos de reconstrucción tridimensional usando imágenes de profundidad y el método introducido por Newcombe *et al.*[6]. El método incluye los pasos de medir, estimar la posición, actualizar la reconstrucción, y predecir la superficie. Durante la medición se obtienen los vértices y las normales con respecto a la posición de la cámara. Durante la estimación de la posición, se alinean las nubes de puntos del modelo con los de la imagen actual y se calcula la posición de la cámara con respecto al mundo. La absorción de la medición actual a los datos de reconstrucción se realiza mediante el algoritmo de *Iterative Closest Point*. Con el valor de la posición de la cámara en el sistema de referencia global se actualiza la función de distancia con signo truncada y su ponderación, lo cual permite el ahorro de espacio de almacenamiento. Con la reconstrucción es posible predecir cuáles serían los valores de los vértices y las normales desde algún punto de vista en particular.

Ejercicios

1. Utilizando la librería de Matlab desarrollada por Juan Ramón Terven y publicada en `http://tinyurl.com/tervenKinect`, obtén una secuencia de imágenes. Luego, aplica el algoritmo bilateral a una imagen de profundidad. Comenta los resultados.

2. Programar el algoritmo de ICP utilizando el pseudocódigo en el Algoritmo 11.

3. Demostrar que (6.20) corresponde a la expresión de una rotación por un ángulo pequeño.

4. Verificar la igualdad en

$$\tilde{\mathbf{T}}_k^{gz}\dot{\mathbf{V}}^k(\mathbf{u}) = \tilde{\mathbf{R}}^z\tilde{\mathbf{V}}_k^g(\mathbf{u}) + \tilde{\mathbf{t}}_k^{gz} = \mathbf{G}(\mathbf{u})\mathbf{x} + \tilde{\mathbf{V}}_k^g(\mathbf{u}). \tag{6.30}$$

6.A. Linealización de Rotaciones de Ángulos Pequeños

Las rotaciones pueden ser expresadas de una gran variedad de formas. Una muy popular es mediante los ángulos de Euler. Los ángulos de Euler pueden ser vistos como una secuencia de rotaciones elementales alrededor de los ejes de referencia. Sobre los ángulos de Euler, la literatura distingue entre los ángulos de Euler propios, y los de Tait-Bryan. La diferencia entre unos y otros es que mientras en para los de Euler el primer y el tercer eje de rotación son iguales, en los de Tait-Bryan los ejes son diferentes.

En [5], Nuchter muestra que para nuestros propósitos, nos interesa la combinación donde

$$\begin{aligned}
\mathbf{R} &= \mathbf{R}_x(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\gamma), \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}, \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \sin\gamma & \cos\gamma & 0 \\ -\sin\beta\cos\gamma & \sin\beta\sin\gamma & \cos\beta \end{pmatrix}, \\
&= \begin{pmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & -\sin\alpha\cos\beta \\ -\sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma & \sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma & \cos\alpha\cos\beta \end{pmatrix}.
\end{aligned} \tag{6.31}$$

Por otro lado, la expansión en términos de la serie de Taylor para $\sin\theta$ y $\cos\theta$ es respectivamente

$$\sin\theta = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!}\theta^{2n+1} = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \ldots, \tag{6.32}$$

y

$$\cos\theta \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!}\theta^{2n} = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \ldots. \tag{6.33}$$

En este caso, tratándose de ángulos muy pequeños, tomamos sólo el primer término de la expansión, y considerando que la multiplicación de términos pequeños por otros más pequeños resulta en productos que pueden despreciarse, tenemos que

$$\mathbf{R} \approx \begin{pmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{pmatrix}. \tag{6.34}$$

Bibliografía

[1] Blais, Gérard y Martin D. Levine: *Registering Multiview Range Data to Create 3D Computer Objects*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(8):820–824, 1995.

[2] Chen, Yang y Gérard Medioni: *Object Modelling by Registration of Multiple Range Images*. Image and Vision Computing, 10(3):145–155, 1992.

[3] Curless, Brian y Marc Levoy: *A Volumetric Method for Building Complex Models from Range Images*. En *Conference on Computer Graphics and Interactive Techniques*, páginas 303–312. ACM, 1996.

[4] Hartley, Richard: *In Defense of the Eight-Point Algorithm*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(6):580–593, 1997.

[5] Magnusson, Martin, Andreas Nuchter, Christopher Lorken, Achim Lilienthal y Joachim Hertzberg: *Evaluation of 3D Registration Reliability and Speed: A Comparison of ICP and NDT*. En *IEEE International Conference on Robotics and Automation*, páginas 3907–3912, 2009.

[6] Newcombe, Richard, Andrew Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim y Andrew Fitzgibbon: *KinectFusion: Real-Time Dense Surface Mapping and Tracking*. En *IEEE International Symposium on Mixed and Augmented Reality*, páginas 127–136, 2011.

[7] Parker, Steven, Peter Shirley, Yarden Livnat, Charles Hansen y Peter Pike Sloan: *Interactive Ray Tracing for Isosurface Rendering*. En *Proceedings of the Conference on Visualization*, páginas 233–238, 1998.

[8] Tomasi, Carlo y Takeo Kanade: *Shape and Motion from Image Streams under Orthography: A Factorization Method*. International Journal of Computer Vision, 9(2):137–154, 1992.

[9] Tomasi, Carlo y Roberto Manduchi: *Bilateral Filtering for Gray and Color Images*. En *IEEE International Conference on Computer Vision*, páginas 839–846. IEEE, 1998.

Parte III

Detección de Objetos

CAPÍTULO 7

Clasificadores basados en *Boosting*

Entre las características deseables de un detector está la efectividad y la eficiencia, *i.e.*, un gran nivel de desempeño, expresado como un mínimo número de falsos positivos y falsos negativos, y gran rapidez en su ejecución. El trabajo de Viola y Jones[14] presenta un detector de objetos con exactamente esas bondades. Para lograr eficiencia, su algoritmo introdujo algunas innovaciones, *e.g.*, el uso de la llamada *imagen integral*, lo cual permite el cálculo rápido de características. Para lograr efectividad, propusieron mejoras estructurales a los métodos de aprendizaje, *e.g.*, introdujeron un algoritmo de aprendizaje basado en Adaboost[6], el cual resulta en un esquema que combina clasificadores cada vez más sofisticados en *cascada*, descartando regiones poco interesantes rápidamente y concentrando recursos en regiones prometedoras. El trabajo de Viola y Jones ha sido seminal en el surgimiento de muchos otros detectores que se basan en el mismo esquema de desarrollo. Por ejemplo, tenemos el trabajo de Piotr Dóllar *et al.*[4], el cual generaliza el concepto de obtención rápida de características dentro del modelo de *boosting*[12]. Ahora bien, dicho lo anterior conviene remarcar que los algoritmos basados en *boosting* pueden ser muy susceptibles a errores en la asignación de etiquetas de clasificación de las muestras usadas para entrenamiento[8]. De tal suerte, dada la importancia de sus aplicaciones, en este capítulo estudiamos el algoritmo de clasificación de Viola-Jones y la generalización propuesta por Dóllar.

7.1. Características *Haar-like*

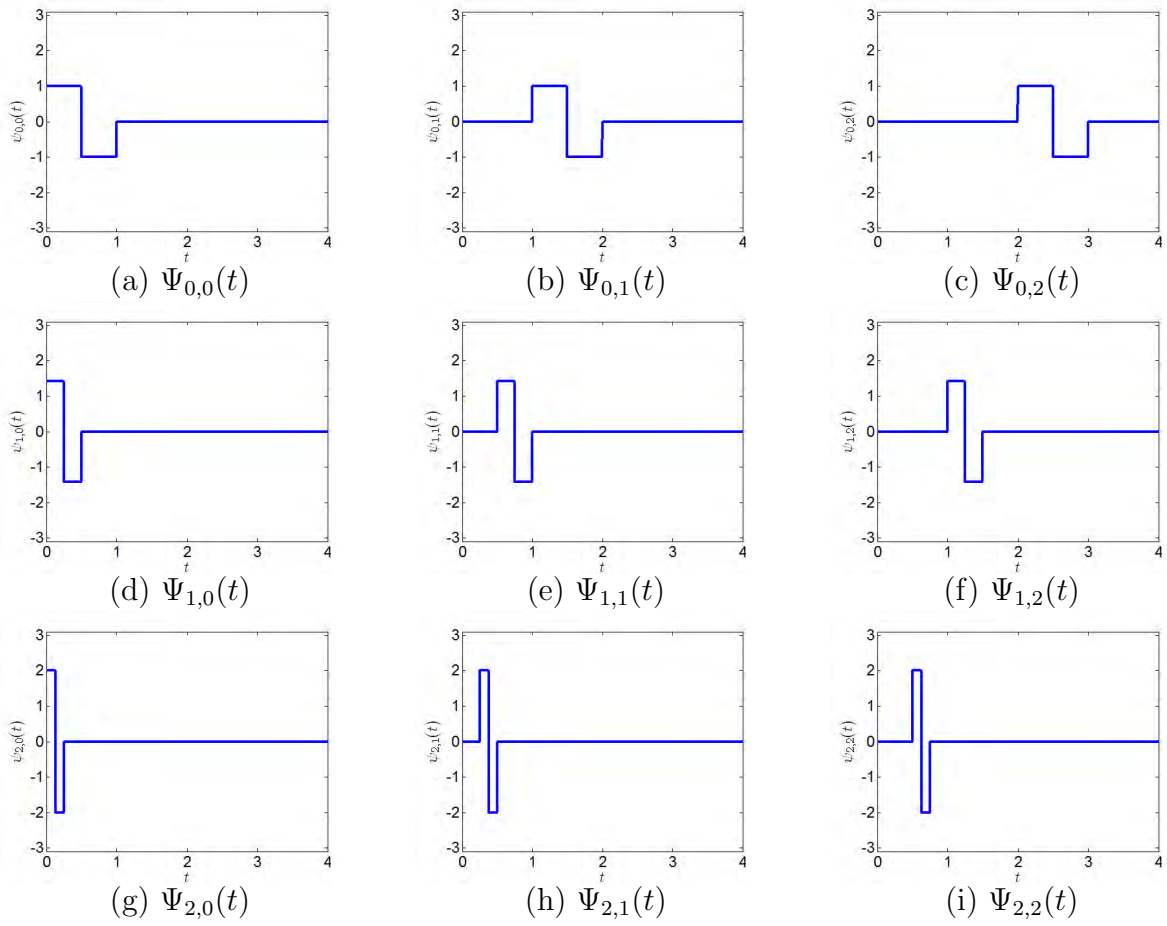
La familia de funciones ortonormales de Haar[11] es ahora reconocida como el primer ejemplo de análisis `wavelet`, o análisis espacio-frecuencia. La función `wavelet` madre de Haar $\psi(t)$ puede ser descrita como

$$\psi(t) = \begin{cases} 1 & 0 \leq t \leq 1/2, \\ -1 & 1/2 \leq t < 1, \\ 0 & \text{en otro caso,} \end{cases} \tag{7.1}$$

donde el resto de las funciones del sistema se definen como

$$\psi_{n,k}(t) = 2^{n/2}\psi(2^n t - k). \tag{7.2}$$

Un ejemplo de `wavelets` de Haar se muestra en la Figura 7.1 Las características usadas por Viola y Jones tienen mucha similitud con las bases de Haar introducidas en [9]. Es



(a) $\Psi_{0,0}(t)$ (b) $\Psi_{0,1}(t)$ (c) $\Psi_{0,2}(t)$

(d) $\Psi_{1,0}(t)$ (e) $\Psi_{1,1}(t)$ (f) $\Psi_{1,2}(t)$

(g) $\Psi_{2,0}(t)$ (h) $\Psi_{2,1}(t)$ (i) $\Psi_{2,2}(t)$

Figura 7.1: `Wavelets` de Haar para $n = 0, \ldots, 2$ y $k = 0, \ldots, 2$. Las funciones permiten la localización en espacio, a través de k (las columnas), y en frecuencia, a través de n (las hileras).

importante notar que las bases que actualmente se usan en librerías como la de OpenCV[1] son un conjunto extendido de las definidas por Viola y Jones[7].

Viola y Jones introducen una etapa de pro-procesamiento llamada *imagen integral*. En esta etapa, la imagen integral $ii(x, y)$ contiene en (x, y) la suma de la intensidad de los pixeles que se ubican arriba y a la izquierda, inclusive, como

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'), \tag{7.3}$$

donde $i(x, y)$ es la imagen original. Viola y Jones notan que la sumatoria puede ser resuelta usando las siguientes recurrencias

$$\begin{aligned} s(x, y) &= s(x, y-1) + i(x, y), \\ ii(x, y) &= ii(x-1, y) + s(x, y), \end{aligned} \tag{7.4}$$

donde suponiendo que la esquina superior izquierda de la imagen corresponde con el punto $(1, 1)$ y que las coordenadas positivas corren hacia la derecha y hacia abajo, se tienen las

11	10	9	11	2
13	13	11	9	1
8	11	10	7	0
3	7	9	8	3
6	7	10	11	7

(a)

0	0	0	0	0	0
0	11	21	30	41	43
0	24	47	67	87	90
0	32	66	96	123	126
0	35	76	115	150	156
0	41	89	138	184	197

(b)

0	0	0	0
0	11	21	30
0	24	47	67
0	32	66	96

0	0	0	0	0	0
0	11	21	30	41	43
0	24	47	67	87	90
0	32	66	96	123	126

0	0	0	0
0	11	21	30
0	24	47	67
0	32	66	96
0	35	76	115
0	41	89	138

0	0	0	0	0	0
0	11	21	30	41	43
0	24	47	67	87	90
0	32	66	96	123	126
0	35	76	115	150	156
0	41	89	138	184	197

(c)

Figura 7.2: Imagen integral. En (a) se presenta una matriz cuyos valores vienen de una imagen pequeña. En (b) se muestra la imagen integral. La imagen integral de un rectángulo se puede calcular utilizando los valores de la esquina inferior derecha de los rectángulos, como la operación + - - . En este ejemplo, la suma de los valores de los pixeles en la región de 2×2 en la esquina inferior derecha es 197 + 96 - 126 - 138 = 29.

relaciones $s(x, 0) = 0$ y $ii(0, y) = 0$. Así, dado el rectángulo de anchura w y altura h, cuya esquina inferior derecha está definida en (x, y), la suma de los valores dentro de él está definida por (ver Figura 7.2)

$$ii(x, y) + ii(x - w, y - h) - ii(x, y - h) - ii(x - w, y). \tag{7.5}$$

En el caso de las características a 45° del conjunto extendido definido por Lienhart y Maydt[7] (ver la Figura 7.3), el valor de la imagen integral también se define por cuatro evaluaciones de una imagen integral. Sin embargo, en este caso la imagen integral es construida en dos pasadas sobre la imagen de intensidad.

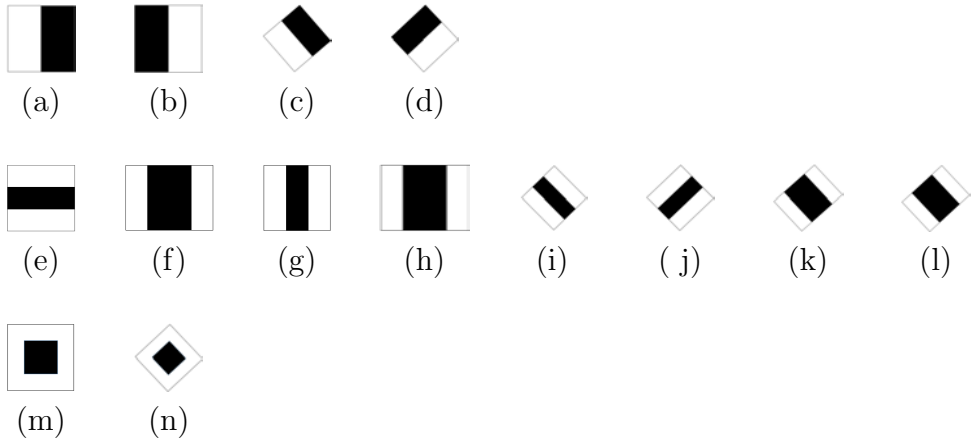


Figura 7.3: Conjunto de características usadas por OpenCV[1] para la producción de clasificadores.

7.2. Clasificación con *Adaboost*

Viola y Jones[14] proponen una variante de `Adaboost` tanto para seleccionar las características como para entrenar al clasificador. En su algoritmo, Viola y Jones seleccionan la característica rectangular que mejor separa ejemplos positivos de ejemplos negativos, medido esto como el mínimo número de clasificaciones erróneas. Así, un clasificador débil $h_j(\mathbf{x})$ consiste en una característica f_j, un umbral θ_j, y una polaridad p_j que indica la dirección de la desigualdad, tal que

$$h_j(\mathbf{x}) = \begin{cases} 1 & \text{si } p_j f_j(\mathbf{x}) < p_j \theta_j, \\ 0 & \text{en otro caso.} \end{cases} \tag{7.6}$$

A continuación desarrollamos un ejemplo siguiendo los pasos del Algoritmo 14. Supóngase que se tiene los siguientes datos:

x	1	2	3	4	5	6	7	8	9	10
y	1	1	1	0	0	0	1	1	1	0

y los pesos (redondeados) son inicializados como

índice	1	2	3	4	5	6	7	8	9	10
w_1	0.08	0.08	0.08	0.13	0.13	0.13	0.08	0.08	0.08	0.13

Ahora iniciamos las iteraciones para la búsqueda de los clasificadores. En la primera iteración, $t = 1$, los pesos son normalizados no teniendo cambios con respecto a los anteriores valores. Supóngase que el clasificador en (7.6) se define con $f_j(x) = x$. El valor de ϵ_i es calculado usando

$$\epsilon_i = \sum_i w_{t,i} |h_j(x_i) - y_i|, \tag{7.7}$$

para los diferentes valores de θ_j. Esto resulta en

θ	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
$p_j = -1$	0.58	0.67	0.75	0.63	0.50	0.38	0.46	0.54	0.63
$p_j = +1$	0.42	0.33	0.25	0.38	0.50	0.63	0.54	0.46	0.38

de donde resulta que el valor mínimo $\epsilon_j = 0.25$ se da para $p_j = 1$, $\theta_j = 3.5$. La aplicación de este clasificador resulta, según

$$e_i = \begin{cases} 0 & \text{si } \mathbf{x}_i \text{ es clasificador correctamente,} \\ \\ 1 & \text{en otro caso,} \end{cases} \tag{7.8}$$

en

índice	1	2	3	4	5	6	7	8	9	10
e	0	0	0	0	0	0	1	1	1	0

Conviene recordar que clasificaciones correctas se indican con 0. Por consiguiente, con el valor de $\beta_t = 0.33$ obtenido con

$$\beta_t = \frac{\epsilon_t}{1 - \epsilon_t}, \tag{7.9}$$

los nuevos pesos resultan en

índice	1	2	3	4	5	6	7	8	9	10
$\hat{w}_2$	0.03	0.03	0.03	0.04	0.04	0.04	0.08	0.08	0.08	0.04

En la segunda iteración, $t = 2$, los pesos son normalizados resultando en

índice	1	2	3	4	5	6	7	8	9	10
w_2	0.06	0.06	0.06	0.08	0.08	0.08	0.17	0.17	0.17	0.08

Ahora, el valor de ϵ_i es calculado en (7.7) para los diferentes valores de θ_j. Esto resulta en

θ	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
$p_j = -1$	0.39	0.44	0.50	0.42	0.33	0.25	0.42	0.58	0.75
$p_j = +1$	0.61	0.56	0.50	0.58	0.67	0.75	0.58	0.46	0.25

de donde resulta que el valor mínimo $\epsilon_j = 0.25$ se da para $p_j = 1$, $\theta_j = 6.5$. La aplicación de este clasificador resulta, según (7.8) en

índice	1	2	3	4	5	6	7	8	9	10
e	1	1	1	0	0	0	0	0	0	1

Por consiguiente, con el valor de $\beta_t = 0.33$ obtenido con (7.9), los nuevos pesos resultan en

índice	1	2	3	4	5	6	7	8	9	10
$\hat{w}_2$	0.06	0.06	0.06	0.03	0.03	0.03	0.06	0.06	0.06	0.08

Después de una tercera iteración, se tienen los valores de $\theta_j = 3.5; 6.5; 9.5$ y $p_j = 1, -1, 1$.

7.3. Cascada de Clasificadores

El pseudocódigo en Algoritmo 14 describe la parte de Adaboost correspondiente a la selección de los clasificadores débiles. Una vez que se aplica este algoritmo, el clasificador fuerte es definido como

$$h(\mathbf{x}) = \begin{cases} 1 & \text{si } \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \geq 1/2 \sum_{t=1}^{T} \alpha_t, \\ \\ 0 & \text{en otro caso,} \end{cases} \tag{7.10}$$

donde $\alpha_t = -\log \beta_t$. La idea es que los clasificadores rechacen los casos negativos manteniendo al mismo tiempo los casos positivos. Así, el resultado positivo de un clasificador llama a la evaluación de la característica por un segundo clasificador. Si el resultado sigue siendo positivo, un tercer clasificador es llamado, y así sucesivamente. El rechazo por un clasificador implica que la característica analizada no corresponde al objeto que se quiere detectar. Esto da origen a una representación similar a un árbol de decisión donde una rama implica rechazo y la otra continúa con la evaluación de la característica.

En el algoritmo propuesto por Viola y Jones[14], durante la operación el clasificador trabaja a diferentes escalas, cambiando el tamaño del detector y no de la imagen, con separaciones de escala de 1.25. Para resolver múltiples detecciones se promedian las *bounding boxes*, que corresponden al lugar donde ocurre la detección, para obtener los límites de la detección.

7.4. Canales de Características

Dóllar *et al.*[5] avanzan la idea planteada por Viola y Jones mediante la inclusión de múltiples canales visuales. Aunque el esquema es más general, en su aproximación incluyen la magnitud del gradiente, la información de color y seis intervalos de histogramas de orientaciones, seguidos de la extracción de características usando el concepto de la imagen integral sobre regiones rectangulares locales. La implementación de su algoritmo se describe en [3]. Esencialmente, la idea es tener un conjunto amplio de características que puedan ser usadas para construir los clasificadores débiles. Entre las características que ahora se incluyen se encuentran histogramas de gradientes orientados e información de color. Enseguida detallamos como los histogramas pueden ser calculados de forma rápida utilizando histogramas integrales y como el mapa de color LUV puede ser obtenido.

7.4.1. Histograma Integral

Normalmente, el cálculo de un histograma entraña la visita de los elementos $x \in X$ de la muestra y su clasificación en el bin $b \in B$ que describe el rango en el cual x está incluido. Porikli [10] extendió la idea de la imagen integral al concepto de *histogramas integrales*. En su aproximación, si el tamaño de la imagen es de w columnas por h hileras, el histograma integral es representado por un arreglo tridimensional $\mathbf{H}(\mathbf{x}, b)$ de tamaño w columnas, h hileras, y $\|B\|$ intervalos.

x	1	2	3	4	5	6	7	8
$\mathbf{I}(x)$	1	1	2	1	3	2	1	3

b	$\mathbf{H}(x,b)$							
1	1	2	2	2	3	3	3	4
2	0	0	1	1	1	2	2	2
3	0	0	0	0	1	1	1	2

Figura 7.4: Histograma integral. Supóngase una imagen de una dimensión $\mathbf{I}(x)$. Bajo el renglón que describe los valores de cada posición de $I(x)$ se identifica el valor de cada intervalo b del histograma integral $H(x,b)$. Utilizando (7.11) con $d=1$, se tiene que el valor del histograma para el intervalo $T=\{4\leq x\leq 7\}$ es $\mathbf{h}(T,b)=\mathbf{H}(7,b)-\mathbf{H}(3,b)=1$.

Porikli hace notar que el histograma de una porción de la imagen puede describirse como la unión de histogramas integrales que siguen la forma general dada por

$$\mathbf{h}(T,b)=\sum_{r=0}^{d}(-1)^r\sum_{l=1}^{C_r^d}\mathbf{H}(\mathbf{x}_l^r,b), \tag{7.11}$$

donde $C_r^d=\begin{pmatrix} d \\ r \end{pmatrix}$ corresponde al número de elementos de $\mathbf{x}_l^r$ que son necesarios para describir la frontera del hipervolumen de dimensión d donde se aplica el histograma integral. Porikli desarrolla el concepto de *histogramas integrales* para espacios de d dimensiones que mediante la evaluación de las funciones f apropiadas resultan en tensores en k dimensiones. Es decir, supóngase que se tiene un hipervolumen cuya posición es descrita por coordenadas de la forma $\mathbf{x}=(x_1,\ldots,x_d)$, una función f podría mapear los valores de posición a k valores $\mathbf{y}=(y_1,\ldots,y_k)$. Un ejemplo podría ser la expresión de color RGB de un pixel en una imagen bidimensional. Para el caso de imágenes de intensidad podemos instanciar d a dos dimensiones y k a uno, correspondientes con las coordenadas de la imagen y a su valor de gris. En ese sentido, (7.11) puede simplificarse a

$$\begin{aligned}\mathbf{h}(T,b) &= \sum_{r=0}^{2}(-1)^r\sum_{l=1}^{C_r^2}\mathbf{H}(\mathbf{x}_l^r,b),\\ &= \mathbf{H}(\mathbf{x}_1^0,b)-\mathbf{H}(\mathbf{x}_1^1,b)-\mathbf{H}(\mathbf{x}_2^1,b)+\mathbf{H}(\mathbf{x}_l^2,b),\end{aligned} \tag{7.12}$$

donde $\mathbf{x}=(x,y)^T$ y por su lado el histograma integral está dado por

$$\mathbf{H}(x,y,b)=\mathbf{H}(x-1,y,b)+\mathbf{H}(x,y-1,b)-\mathbf{H}(x-1,y-1,b)+Q(\mathbf{I}(x,y),b), \tag{7.13}$$

donde $Q(\mathbf{I}(x,y),b)$ es uno si el bin que corresponde al punto (x,y) es igual a b, y ambos $\mathbf{H}(0,y,b)$ y $\mathbf{H}(x,0,b)$ son iguales a cero. Un ejemplo de la aplicación del histograma integral se presenta en la Figura 7.4. Una implementación eficiente del algoritmo del histograma integral se encuentra en la librería `VLFeat`[13].

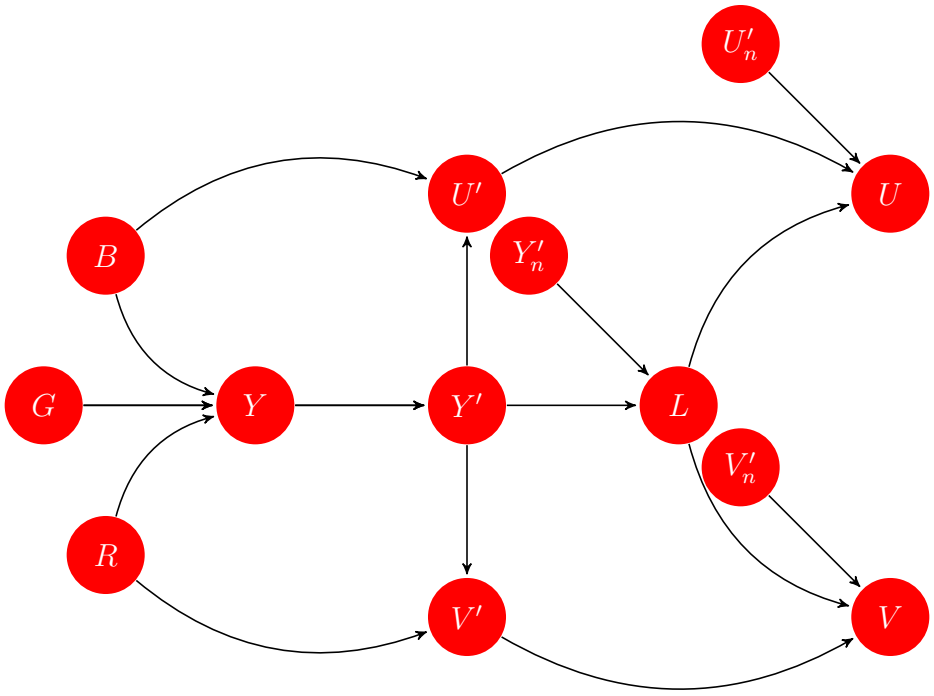


Figura 7.5: Obtención del modelo de color LUV a partir de RGB. Ver el texto para detalles de las ecuaciones utilizadas.

7.4.2. Detección Multicanal

Para Dóllar *et al.*[5], dada una imagen $\mathbf{I}$ las funciones generadoras de canales pueden ser descritas como $\mathbf{C} = \Omega(\mathbf{I})$. Enseguida, la suma de los pixeles resultantes en una región rectangular fija se denota como $f(\mathbf{C})$. En el caso de Viola y Jones, $\mathbf{C} = \mathbf{I}$, y las características son funciones parecidas a las funciones base de Haar. Los canales usados por Dóllar *et al.* incluyen la información de color, magnitud del gradiente (calculado para imágenes a color de tres bandas como el máximo valor sobre cada banda), e histogramas de orientaciones de gradiente (que serán vistos en el capítulo 8).

Para el análisis de color se utiliza el modelo de color CIE-LUV (ver Figura 7.5). Este se expresa como[15]

$$\begin{aligned} L &= \begin{cases} \left(\frac{29}{3}\right)^3 Y'/Y'_n, & Y'/Y'_n \leq \left(\frac{6}{29}\right)^3, \\ 116(Y'/Y'_n)^{1/3} - 16, & Y'/Y'_n > \left(\frac{6}{29}\right)^3, \end{cases} \\ U &= 13L(U' - U'_n), \\ V &= 13L(V' - V'_n), \end{aligned} \tag{7.14}$$

donde (U'_n, V'_n) y Y'_n son las coordenadas de crominancia, (U', V'), y luma, Y', de un punto blanco que se toma como referencia. Normalmente $U' = B - Y'$, y $V' = R - Y'$ son las diferencias de las bandas azul, B, y rojo, R, con respecto a la luma. Por su parte, la luma

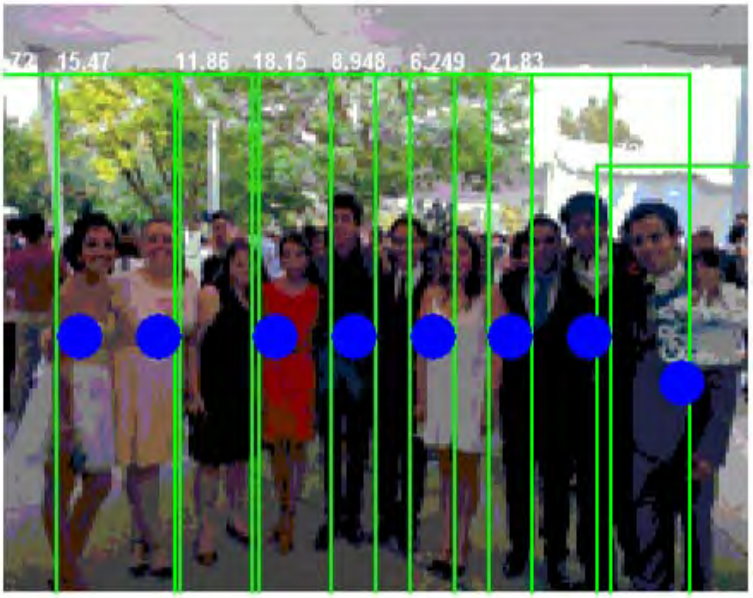


Figura 7.6: Detector de personas de Piotr Dollár *et al.* [4]. Las líneas verdes enmarcan las detecciones. Para una mejor apreciación, hemos marcado el centro de los rectángulos delimitadores. Se puede observar que hay ocho verdaderos positivos y dos falsos negativos.

es la luminosidad con gamma corregida. Es decir,

$$Y' = AY^{\gamma}, \tag{7.15}$$

donde a implica luminancia y a' luma, para una constante γ y una constante arbitraria A. Asimismo, los valores de luminancia están dados por

$$Y = 0.2126R + 0.7142G + 0.0722B. \tag{7.16}$$

Entretanto, los histogramas de gradiente se forman por la dirección del gradiente pesada por su magnitud [2].

Sumario

Este documento presenta el detector de objetos de Viola y Jones[14]. Esencialmente, estos esquemas permiten conjuntar clasificadores débiles en robustos. Algunas de las operaciones del método de Viola y Jones, como la imagen integral y el esquema de Adaboost, son presentados en detalle. Algunos avances recientes involucran el uso de canales de características[5], entendidos éstos como información adicional que incluye color o histogramas de gradientes orientados. Esto posibilita que las operaciones de caracterización se realicen muy eficientemente. Un ejemplo de la implementación de Piotr Dóllar[3] se ilustra en la Figura 7.6.

Cuadro 7.1: Matriz representando una pequeña imagen.

6	5	4	6	2
8	8	6	4	1
3	6	5	2	5
3	2	4	3	3
1	2	5	6	2

Cuadro 7.2: Posiciones correspondientes a valores para una imagen e índices para un histograma de 3 bins.

x	1	2	3	4	5	6	7	8
$\mathbf{I}(x)$	3	2	1	3	1	2	2	1

b
1
2
3

Ejercicios

1. Implementa un algoritmo para visualizar las funciones de Haar para términos arbitrarios de n y k de acuerdo a (7.2).

2. Ilustra el funcionamiento del concepto de la imagen integral para la imagen dada por la matriz en la Tabla 7.1. Obtén la suma de los valores para el subrectángulo de tres columnas y dos hileras en la esquina inferior derecha.

3. Con base en el Algoritmo 14 y los datos del ejemplo en la sección 7.2, codifica un programa para obtener los clasificadores débiles óptimos de acuerdo al criterio de Adaboost.

4. Con base en los datos en la Tabla 7.2 obtén el histograma integral y el histograma del intervalo $T = \{2 \leq x \leq 5\}$.

Llamada : $< \mathbf{H}, \Gamma > \leftarrow$ `Adaboost` $(\mathcal{X}, T)$
Entradas: El conjunto, $\mathcal{X} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, con $y_i \in \{0, 1\}$, de muestras positivas y negativas, y el número T de clasificadores débiles.
Salidas : El conjunto de clasificadores débiles $\mathbf{H} = \{h_1, \ldots, h_T\}$ y pesos asociados $\Gamma = \{\beta_1, \ldots, \beta_T\}$.

`// Inicializar los pesos, donde` m `y` l `son respectivamente el número de ejemplos negativos y positivos`

$$w_{1,i} \leftarrow \begin{cases} 1/(2m) & \text{si } y_i = 0, \\ & ; \\ 1/(2l) & \text{si } y_i = 1, \end{cases}$$

`// Obtener` T `clasificadores débiles`
for *t desde 1 hasta T* **do**
  `// Normalizar los pesos tal que puedan ser interpretados como una distribución de probabilidades`
  $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$;
  `// Para cada característica` f_j`, entrenar un clasificador` h_j`. El error es evaluado utilizando (7.7). Se escoge el clasificador` h_t `con el menor error` ϵ_t
  $h_t \leftarrow$ `Entrenar_Clasificador` $(\mathcal{X})$;
  `// actualizar los pesos, donde se aplica el criterio para` e_t `en (7.8) con la definición de` β_t `en (7.9)`
  $w_{t+1,i} \leftarrow w_{t,i}\beta_t^{1-e_i}$;
end

Algorithm 14: Algoritmo para el entrenamiento de clasificadores débiles. Dado el conjunto de entrenamiento $\mathcal{X}$ y el número de clasificadores débiles por entrenar, el proceso de Adaboost busca el mejor clasificador h_t basándose en las características disponibles f_j. Al final de cada iteración los pesos son modificados para poner más atención en los ejemplos mal clasificados.

Bibliografía

[1] Bradski, G.: *The OpenCV Library*. Dr. Dobb's Journal of Software Tools, 2000.

[2] Dalal, Navneet y Bill Triggs: *Histograms of Oriented Gradients for Human Detection*. En *IEEE Conference on Computer Vision and Pattern Recognition*, volumen 1, páginas 886–893, 2005.

[3] Dollar, Piotr: *Piotr's Image & Video Matlab Toolbox*. `http://vision.ucsd.edu/~pdollar/toolbox/doc/`, Octubre 2014.

[4] Dollár, Piotr, Ron Appel, Serge Belongie y Pietro Perona: *Fast Feature Pyramids for Object Detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014.

[5] Dollár, Piotr, Zhuowen Tu, Pietro Perona y Serge Belongie: *Integral Channel Features*. En *British Machine Vision Conference*, volumen 2, página 5, 2009.

[6] Freund, Yoav y Robert Schapire: *A Desicion-Theoretic Generalization of On-line Learning and an Application to Boosting*. En *Computational Learning Theory*, páginas 23–37. Springer, 1995.

[7] Lienhart, Rainer y Jochen Maydt: *An Extended Set of Haar-like Features for Rapid Object Detection*. En *IEEE International Conference on Image Processing*, volumen 1, páginas I–900, 2002.

[8] Long, Philip y Rocco Servedio: *Random Classification Noise Defeats All Convex Potential Boosters*. Machine Learning, 78(3):287–304, 2010.

[9] Papageorgiou, Constantine, Michael Oren y Tomaso Poggio: *A General Framework for Object Detection*. En *IEEE International Conference on Computer Vision*, páginas 555–562, 1998.

[10] Porikli, Fatih: *Integral Histogram: A Fast Way to Extract Histograms in Cartesian Spaces*. En *IEEE Conference on Computer Vision and Pattern Recognition*, volumen 1, páginas 829–836, 2005.

[11] Sarkar, Tapan, Chaowei Su, Adve, M. Salazar-Palma, L. Garcia-Castillo y Rafael Boix: *A Tutorial on Wavelets from an Electrical Engineering Perspective. I. Discrete Wavelet Techniques*. IEEE Antennas and Propagation Magazine, 40(5):49–68, 1998.

[12] Schapire, Robert: *The Strength of Weak Learnability*. Machine learning, 5(2):197–227, 1990.

[13] Vedaldi, A. y B. Fulkerson: *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. `http://www.vlfeat.org/`, 2008.

[14] Viola, Paul y Michael Jones: *Rapid Object Detection using a Boosted Cascade of Simple Features*. En *IEEE Conference on Computer Vision and Pattern Recognition*, volumen 1, páginas I–511, 2001.

[15] Wyszecki, Gunter y Walter Stanley Stiles: *Color Science*, volumen 8. Wiley New York, 1982.

CAPÍTULO 8

Clasificación HOG + SVM

Una de las técnicas actuales más sobresalientes para la detección de objetos se basa en el uso de Histogramas de Gradientes Orientados (HOG). Introducidas por Dalal y Triggs[4], en el algoritmo se codifica el comportamiento de la información del gradiente para una área de análisis. La característica obtenida se representa en un espacio de clasificación donde se define un hiperplano que distingue entre lo que pertenece o no a la clase mediante una máquina de soporte vectorial (SVM) [3] lineal. En su versión original, la detección se hacía sobre objetos completos. Recientemente, Felzenszwalb *et al.*[5] extendieron la noción de Dalal y Triggs al considerar el entrenamiento automático de partes. Esto permite identificar a los objetos aún cuando algunas características estén ocultas en la imagen por la composición de objetos en la escena.

8.1. Objetos Completos

Dalal y Triggs[4] introdujeron un método para la detección de personas basado en el uso de HOGs (ver Figura 8.1). En la región de análisis Dalal y Triggs proponen construir histogramas basados en la orientación del gradiente sobre regiones llamadas *células*. Para la detección de humanos, la acumulación de orientaciones se da en nueve intervalos, en el rango $0° - 180°$. Sin embargo, Dalal y Triggs[4] observan que para la detección de otros objetos el rango $0° - 360°$ puede funcionar mejor. Luego, las regiones llamadas *células* son agrupadas en regiones más grandes llamadas *bloques*. Dalal y Triggs encontraron que el uso de bloques con estructura circular o rectangular no hacen mayor diferencia en el desempeño del clasificador. Los histogramas se construyen en base al número de gradientes en una cierta orientación pesados por la magnitud del gradiente. Sin embargo, antes de acumular votos en las *células*, la magnitud del gradiente es pesada por un Gaussiano centrado en el *bloque*. Específicamente para la detección de humanos, Dalal y Triggs determinaron que los *bloques* de 2×2 células funcionan mejor, con cada *célula* definida sobre una región de 8×8 pixeles. El tamaño de la ventana de análisis que Dalal y Triggs usan es 64×128 pixeles. Esto significa que un vector $\mathbf{v}$ utilizado para contener un descriptor HOG tienen una dimensión de 7 (bloques en dirección horizontal) $\times$ 15 (bloques en la dirección vertical) $\times$ 4 (células/bloque) $\times$ 9 (bins/célula) = 3,780 posiciones. Para reducir el efecto de cambio de iluminación, Dalal y Triggs determinan que normalizar el vector descriptor $\mathbf{v}$ usando cualquiera de las siguientes formulaciones, da buenos y similares resultados:

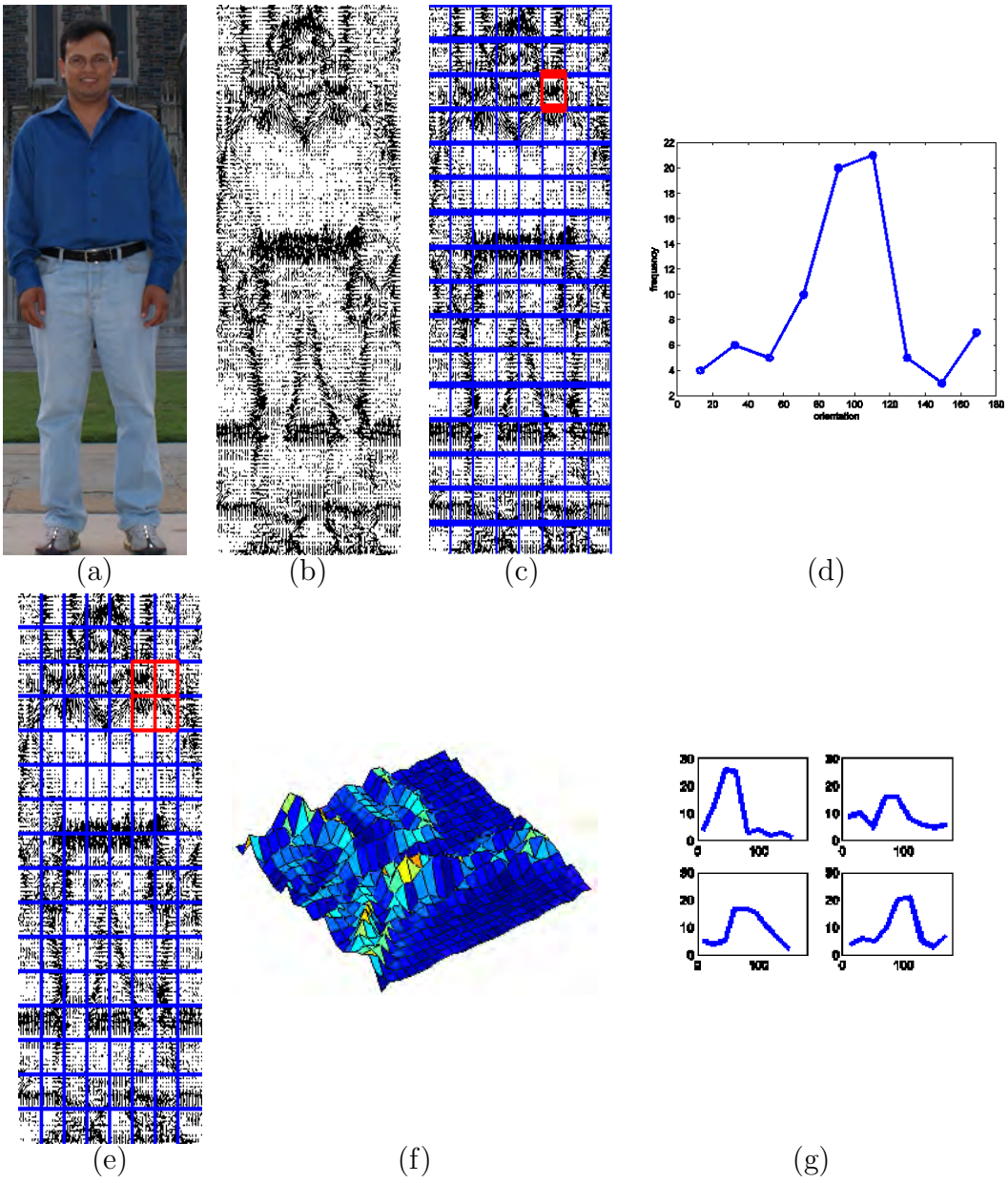


Figura 8.1: Histogramas de Gradientes Orientados de Dalal y Triggs[4]. El funcionamiento del detector fue mostrado para identificar personas paradas, sin partes ocluidas (a). El primer paso consiste en obtener el gradiente (b). Luego, para regiones de análisis, llamadas *células* (c), se calculan histogramas de orientación del gradiente. La contribución de cada vector de gradiente se da en función de la magnitud, pesada por un Gaussiano (f), sobre un conjunto de *células*, llamado *bloque* (e). Al final, se tiene un descriptor que agrupa, para toda la región de análisis, los histogramas obtenidos para cada *bloque*.

- Norma-2
$$\mathbf{v} \rightarrow \mathbf{v}/\sqrt{\|\mathbf{v}\|_2^2 + \epsilon}. \tag{8.1}$$
- Norma-2, seguido de limitar el valor máximo de una entrada de $\mathbf{v}$ a 0.2, seguido de renormalización.
- Norma-1
$$\mathbf{v} \rightarrow \mathbf{v}/(\|\mathbf{v}\|_1^2 + \epsilon). \tag{8.2}$$

Para un valor ϵ muy pequeño.

Su método parece ser indiferente a la representación de color, excepto que trabajar en niveles de gris reduce su desempeño. Ellos normalizan las imágenes usando la corrección gama definida como
$$\mathbf{I}_{\text{salida}} = \mathbf{I}_{\text{entrada}}^{\gamma}, \tag{8.3}$$
donde Dalal y Triggs encuentran que el valor de $\gamma = 1/2$ mejora el desempeño del detector. Una cosa sobresaliente es que el gradiente usado es resultado de aplicar el filtro $[-1, 0, 1]$, sin suavizaje. En imágenes de color obtienen el gradiente para cada banda y toman el que tiene la norma más grande. Finalmente, Dalal y Triggs usan un clasificador basado en una máquina de soporte vectorial lineal para determinar si la observación actual incluye o no a una persona.

Según las observaciones de Dalal y Triggs, su detector tiende a concentrarse en los contornos de la silueta contra el fondo, no en los contornos internos de la persona. Uno de los hallazgos más sobresalientes de Dalal y Triggs es que mucha de la información para la detección de personas se encuentra en los contornos que cambian muy abruptamente en resoluciones altas, y que suavizar esta información degrada el desempeño. También es notable en su trabajo la amplia variedad experimental que les lleva a identificar el mejor diseño de cada etapa del proceso.

8.2. SVM Lineal

Las SVM son clasificadores que buscan identificar el hiperplano que maximiza la distancia de separación entre dos poblaciones. Burges [2] realiza la presentación de los clasificadores SVM de la manera que a continuación se resume.

Dado un conjunto de n características de entrenamiento
$$D = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}_{i=1}^n, \tag{8.4}$$
donde el valor de y_i indica a qué clase pertenece la característica $\mathbf{x}_i \in \mathbb{R}^p$, la idea del SVM lineal es encontrar el hiperplano (ver Figura 8.3)
$$\mathbf{w}^T\mathbf{x} - b = 0, \tag{8.5}$$
que proporcione el máximo margen o distancia entre las dos clases. En esta nomenclatura $\mathbf{w}$ es la normal al hiperplano y $b/\|\mathbf{w}\|$ es la distancia que separa el plano del origen. Para estandarizar el problema, SVM define los hiperplanos
$$\mathbf{w}^T\mathbf{x} - b = 1, \text{ y } \mathbf{w}^T\mathbf{x} - b = -1, \tag{8.6}$$

Llamada : $\mathbf{H} \leftarrow$ `Detectar_Personas` $(\mathbf{I})$
Entradas: Una imagen $\mathbf{I}$
Salidas : El conjunto de regiones de interés $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_n\}$, donde $\mathbf{h}_i = (x, y, w, h)^T$ contiene la posición de la esquina superior izquierda (x, y) y la anchura, w, y altura, h, del rectángulo que inscribe la región de interés.

```
// Crear un descriptor HOG
h ← cv.HOGDescriptor ();
// Establecer el detector de personas de Dalal y Triggs
h.setSVMDetector ('Default');
// Realizar la detección en ventanas a varias escalas
H ← h.detectMultiScale (I);
```

Algorithm 15: Algoritmo de Dalal-Triggs[4] en OpenCV[1] mediante la interfaz de Yamaguchi[8]. Dada una imagen, $\mathbf{I}$, se detectan personas usando descriptores, basados en Histogramas de Gradientes Orientados que se extraen a diferentes escalas. El resultado es un conjunto de rectángulos que inscriben a las regiones candidatas.

para separar los bordes de las clases. Note que si, como en (8.5), la distancia entre la línea y el origen es $\frac{b}{\|\mathbf{w}\|}$, entonces, de manera correspondiente, la distancia entre los hiperplanos identificados en (8.6) es $\frac{2}{\|\mathbf{w}\|}$. Por ello, se dice que el objetivo en SVM es minimizar $\|\mathbf{w}\|$, pues al hacerlo se maximiza la distancia de separación entre las clases. Cuando los datos, $\mathbf{x}_i$, son clasificados correctamente, uno pudiera identificar las regiones donde se ubican de acuerdo a las relaciones

$$\mathbf{w}^T\mathbf{x}_i - b \geq 1, \text{para } y_i = 1, \text{ y } \mathbf{w}^T\mathbf{x}_i - b \leq -1, \text{para } y_i = -1. \tag{8.7}$$

En resumen, un clasificador SVM se construye maximizando la distancia de separación entre las clases

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2, \tag{8.8}$$

sujeto a la restricción de que el hiperplano identificado separe las clases, esto expresado como

$$y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1. \tag{8.9}$$

Tanto la expresión del objetivo como la restricción se pueden plantear en una sola expresión de forma elegante usando los multiplicadores de Lagrange α_i, tal como

$$\arg\min_{\mathbf{w},b,\alpha} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T\mathbf{x}_i - b) - 1) \right) \right\}. \tag{8.10}$$

En ocasiones, es imposible separar las clases por un hiperplano. Para esos casos, Cortes y Vapnik[3] introdujeron una modificación de (8.10) que permite la determinación de un hiperplano cuando las clases no son separables, admitiendo un costo. Para ello, sugirieron el empleo de variables positivas ξ_i que penalizan cada clasificación errónea. Esto es

$$y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 - \xi_i. \tag{8.11}$$

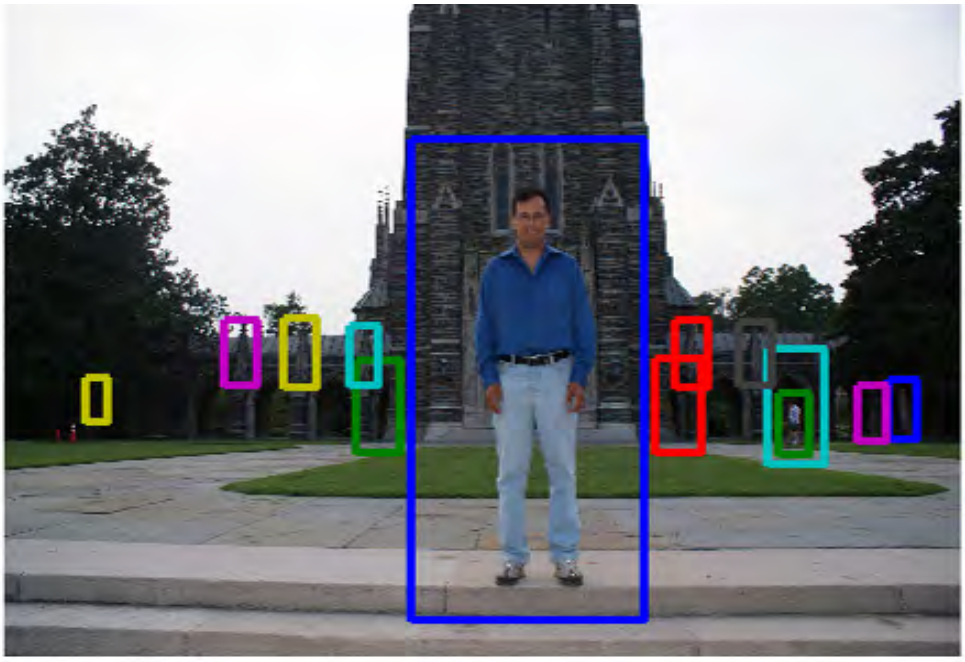


Figura 8.2: Detección de personas usando el HOG implementado en OpenCV[1]. Note como algunas detecciones incluyen personas caminando en el fondo, algunas estatuas, y algunos falsos positivos.

Con esto, la función de clasificación puede quedar definida como

$$\arg\min_{\mathbf{w},b,\xi}\left\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i\right\}, \tag{8.12}$$

sujeto a

$$y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 - \xi, \xi \geq 0. \tag{8.13}$$

O usando los multiplicadores de Lagrange α_i, β_i,

$$\arg\min_{\mathbf{w},b,\alpha,\beta,\xi,}\left\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i - b) - 1 + \xi_i\right) - \sum_{i=1}^{n}\beta_i\xi_i\right\}, \tag{8.14}$$

con $\alpha_i, \beta_i, \xi_i \geq 0$.

8.3. Detección de Partes

El método de Dalal y Triggs permite la detección de objetos cuando éstos se encuentran visibles en su totalidad. Para el caso en que alguna de sus partes ha quedado oculta, se han desarrollado estrategias que permiten su descripción como mezclas de modelos de partes. Esto es parte de una idea antigüa, presentada por primera vez por Fischler y Elschlager[6], donde las entidades son especificadas por una descripción de un conjunto de primitivas y los

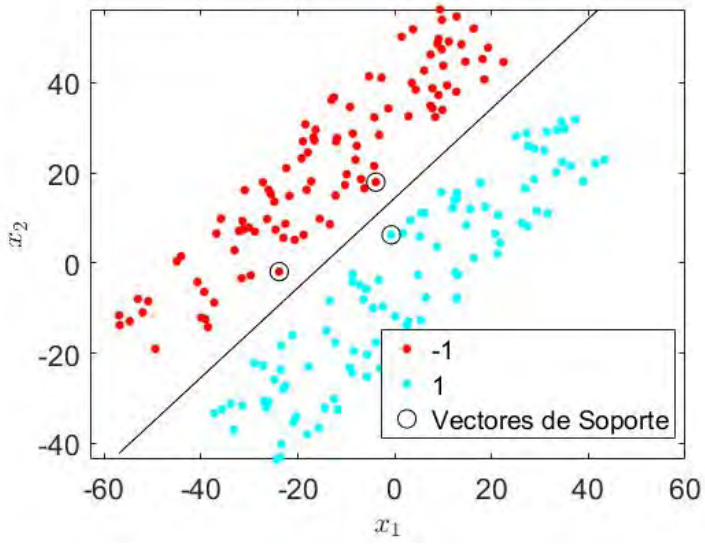


Figura 8.3: Un clasificador SVM corresponde al hiperplano que separa dos clases y cuya distancia a las clases se maximiza. En particular, los puntos correspondientes a la distancia del margen mínima definen los vectores de soporte.

elementos que los unen (ver Figura 8.4). A continuación presentamos una síntesis del trabajo presentado por Felzenszwalb *et al.*[5] (ver Figura 8.5).

Felzenszwalb *et al.*[5] definen un modelo de un objeto con n partes por la $(n+2)$-tupla

$$(\mathbf{f}_0, \mathbf{q}_1, \ldots, \mathbf{q}_n, b), \tag{8.15}$$

donde $\mathbf{f}_0$ es el detector del objeto completo, $\mathbf{q}_i$ es el modelo de la i-ésima parte y b es un término de sesgo. A su vez, cada modelo de parte se define por la 3-tupla

$$\mathbf{q}_i = (\mathbf{f}_i, \mathbf{v}_i, \mathbf{d}_i), \tag{8.16}$$

donde $\mathbf{f}_i$ es el detector para la i-ésima parte, $\mathbf{v}_i$ es su posición más frecuente, relativa a la descripción del objeto completo, y $\mathbf{d}_i$ es un vector de 4×1 conteniendo coeficientes de una función cuadrática, la cual define un costo por la desviación de la actual posición de la parte relativa a su posición más común.

La posible localización de los objetos se define por medio de una hipótesis, del tipo

$$\mathbf{Z} = (\mathbf{p}_0, \ldots, \mathbf{p}_n), \tag{8.17}$$

lo cual especifica la posición $\mathbf{p}_i = (x_i, y_i)$ del i-ésimo filtro. Para Felzenszwalb *et al.*[5], el valor que puede asignársele a una hipótesis es la suma de los valores que resultan de evaluar el detector respectivo menos el costo de detectarlo en una cierta posición relativa a la posición esperada. En otras palabras

$$score(\mathbf{I}, \mathbf{p}_0, \ldots, \mathbf{p}_n) = \sum_{i=0}^{n} \mathbf{f}_i^T \phi(\mathbf{I}, \mathbf{p}_i) - \sum_{i=1}^{n} \mathbf{d}_i^T \phi_d(dx_i, dy_i) + b, \tag{8.18}$$

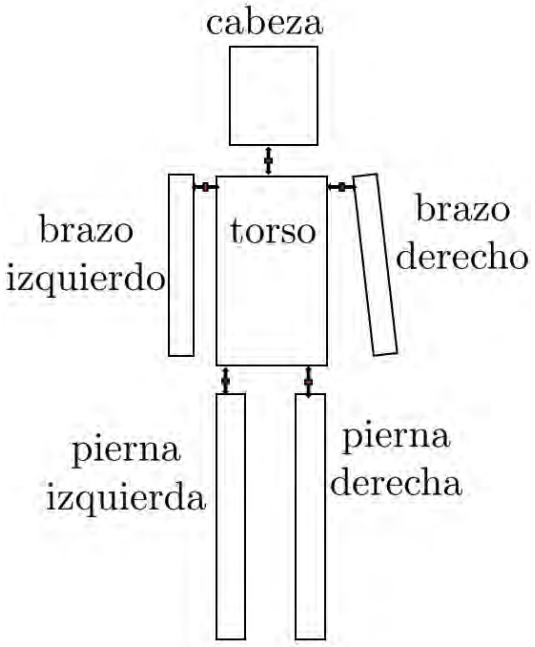


Figura 8.4: Para la detección de objetos donde porciones de ellos pueden estar ocultas, se asume que hay una estructura subyacente que describe las partes y su relación, tal como propusieron Fischler y Elschlager[6].

donde $\mathbf{I}$ es la imagen que se analiza, el primer término describe que tan bien la parte observada corresponde con su modelo y el segundo término penaliza alejarse de la posición esperada para esa parte. La función de costo toma en cuenta que

$$(dx_i, dy_i)^T = (x_i, y_i)^T - \mathbf{v}_i, \tag{8.19}$$

es el desplazamiento de la i-ésima parte relativa a su posición estándar, y

$$\phi_d(dx, dy) = (dx, dy, dx^2, dy^2)^T, \tag{8.20}$$

describe la deformación. El valor, expresado en (8.18), que resulta de evaluar una hipótesis $\mathbf{Z}$, puede ser evaluado también como

$$score(\mathbf{I}, \mathbf{p}_0, \ldots, \mathbf{p}_n) = \beta^T \psi(\mathbf{I}, \mathbf{Z}), \tag{8.21}$$

donde $\mathbf{w}$ engloba los parámetros del modelo tal que

$$\beta = (\mathbf{f}_0^T, \mathbf{f}_n^T, \mathbf{d}_1^T, \ldots, \mathbf{d}_n^T, b)^T, \tag{8.22}$$

y

$$\psi(\mathbf{I}, \mathbf{Z}) = (\phi(\mathbf{I}, \mathbf{p}_0)^T, \ldots, \phi(\mathbf{I}, \mathbf{p}_n)^T, -\phi_d(dx_1, dy_1)^T, \ldots, -\phi_d(dx_n, dy_n)^T, 1)^T. \tag{8.23}$$

La ventaja de esta representación es que el problema se interpreta como encontrar el hiperplano que distingue a la clase en base a la detección de sus partes. La nueva formulación hace factible la construcción de un clasificador lineal. En armonía con las secciones anteriores, Felzenszwalb *et al.*[5] utilizan un clasificador SVM con variables latentes.

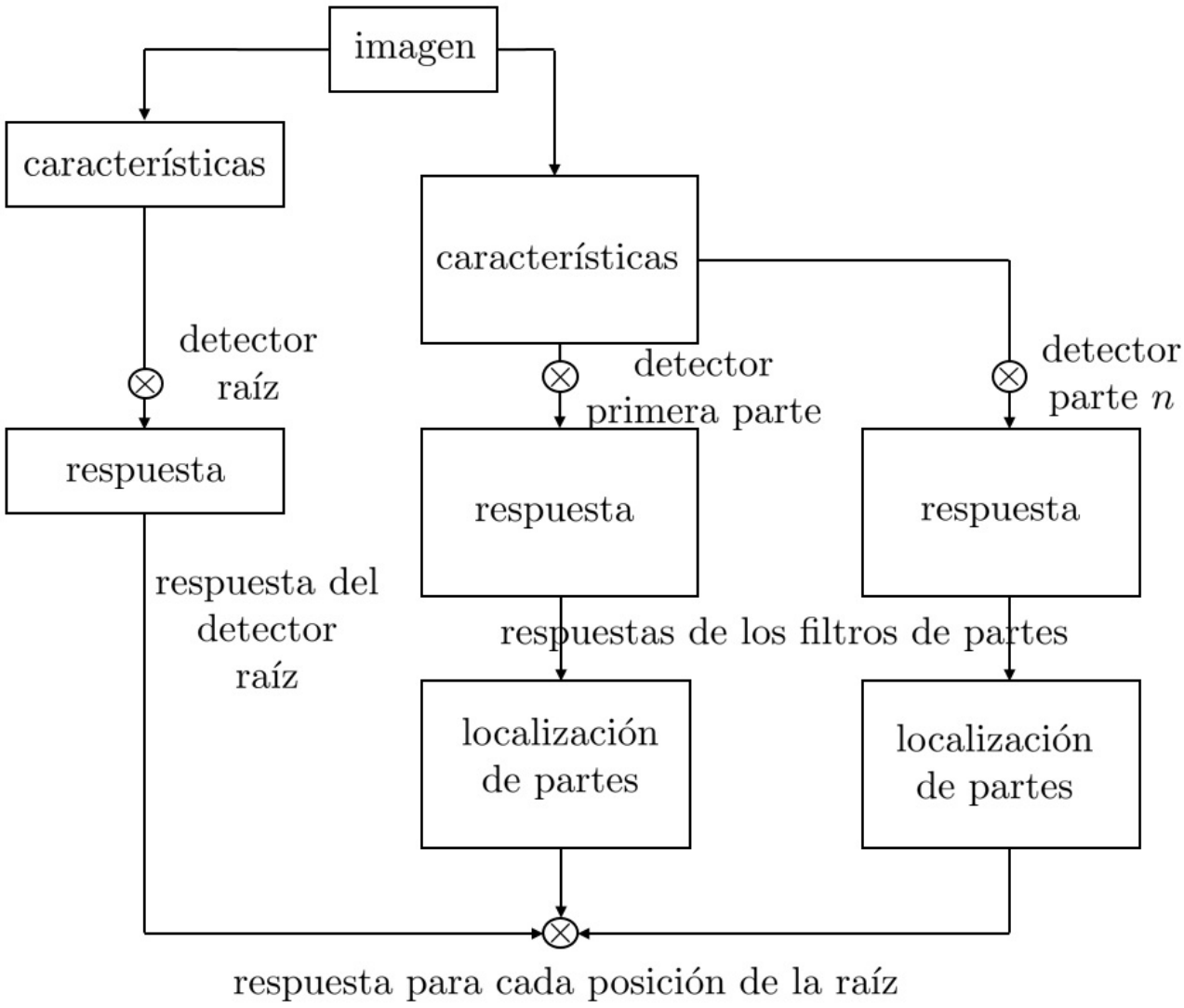


Figura 8.5: Detector de Felzenszwalb *et al.*[5]. Una imagen sirva para localizar la parte raíz y las partes (a un tamaño de 2×). La respuesta al detector raíz es combinada con la respuesta a cada parte combinada con su localización.

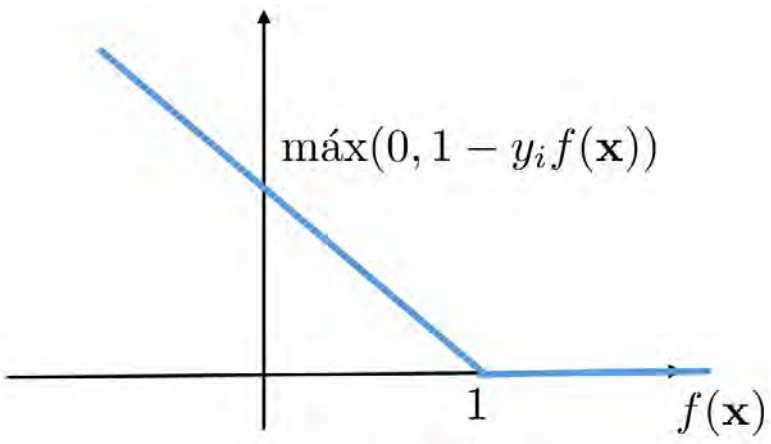


Figura 8.6: *Hinge Loss Function*. Esta función expresa que dada la evaluación de una característica $\mathbf{x}$ por alguna función f, esto resulta en algún valor, positivo o negativo, que cuando resulta en la clasificación claramente correcta, $\|f(\mathbf{x})\| \geq 1$, resulta en una pérdida cero. Sin embargo, cuando $\|f(\mathbf{x})\| < 1$ o la clasificación es incorrecta, la pérdida aumenta linealmente.

8.4. SVM Latentes

En el caso de Dalal y Triggs[4], las partes estaban involucradas en el cálculo del descriptor pero eran absorbidas por la descripción global del objeto. En el caso de Felzenswalb *et al.*[5], las partes son modeladas explícitamente, tal como lo detalla Girshick[7]. Sin embargo, la posición no está definida en la información de entrada, por ello se le trata como latente. Supóngase que para cada parte se tiene un clasificador lineal de la forma

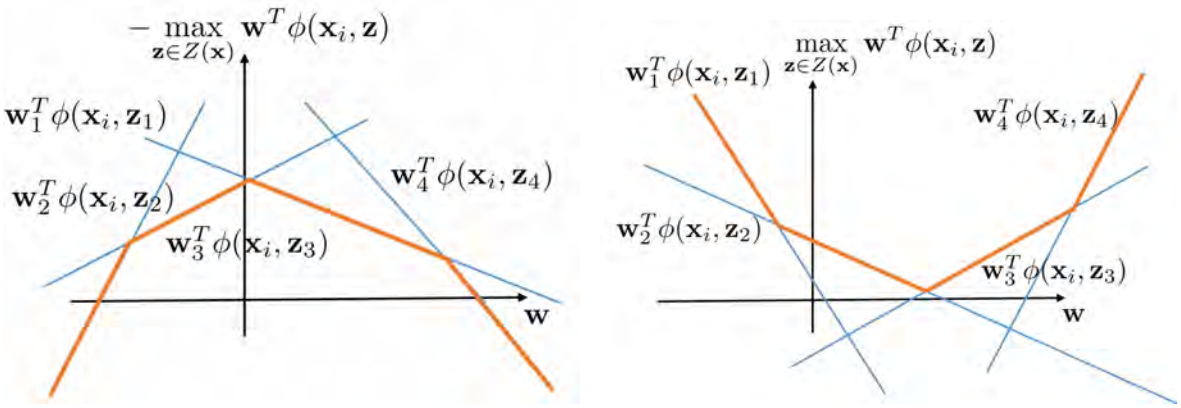
$$f_{\mathbf{w}}(\mathbf{x}) = \max_{\mathbf{z} \in Z(\mathbf{x})} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{z}), \tag{8.24}$$

donde $\mathbf{x}$ corresponde a la caracterización de una porción de la imagen $\mathbf{I}$, $\mathbf{w}$ es un modelo que permite su distinción y $\mathbf{z}$ corresponde a una variable latente relacionada con la posición, la cuales se encuentra en un conjunto $Z(\mathbf{x})$.

En un SVM ordinario, el valor de $\mathbf{w}$ se obtiene usando ejemplos etiquetados, algunos como positivos y otros como negativos, tal que $\mathbf{D} = (\langle \mathbf{x}_1, y_1 \rangle, \ldots, \langle \mathbf{x}_n, y_n \rangle)$, con $y_i \in \{-1, 1\}$, y minimizando

$$L_{\mathbf{D}}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \text{máx}(0, 1 - y_i f_{\mathbf{w}}(\mathbf{x}_i)), \tag{8.25}$$

donde $\text{máx}(0, 1 - y_i f_{\mathbf{w}}(\mathbf{x}_i))$ es una función de pérdida (*hinge loss function*, ver Figura 8.6), que permite aceptar algunas clasificaciones erróneas (tal como propusieron Cortes y Vapnik[3]). Note que esta función es cero cuando se predice la clase correcta. En otro caso, la pérdida se incrementa linealmente.



(a) Concava (b) Convexa

Figura 8.7: Funciones Cóncavas y Convexas. Los términos en (8.26) se descomponen de acuerdo a las muestras positivas y negativas, en funciones cóncavas y convexas respectivamente.

8.4.1. Semiconvexidad

Tal como menciona Girshick[7] (8.25) puede ser descompuesta en dos casos, dependiendo si la característica $\mathbf{x}_i$ pertenece a las muestras positivas o las muestras negativas, como

$$L_D(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i\in P} \text{máx}\left\{0, 1 - \underset{\mathbf{z}\in Z(\mathbf{x})}{\text{máx}}\, \mathbf{w}^T\phi(\mathbf{x}_i,\mathbf{z})\right\} + C\sum_{i\in N} \text{máx}\left\{0, 1 + \underset{\mathbf{z}\in Z(\mathbf{x})}{\text{máx}}\, \mathbf{w}^T\phi(\mathbf{x}_i,\mathbf{z})\right\}. \quad (8.26)$$

En la Figura 8.7 se ilustran los términos de la ecuación. El tercer término resulta en una función convexa pues $\mathbf{w}^T\phi(\mathbf{x}_i,\mathbf{z})$ es un hiperplano, que es convexo, y el máximo de un conjunto de funciones convexas es convexo. Por otro lado, el segundo término es, en general, cóncavo, pues su límite superior es uno y conforme el producto $\mathbf{w}^T\phi(\mathbf{x}_i,\mathbf{z})$ crece la función disminuye, hasta cero. Felzenszwalb *et al.*[5] proponen definir una sola variable latente por cada muestra positiva. En este caso la función $\mathbf{w}^T\phi(\mathbf{x}_i,\mathbf{z}_{P_i})$ es una línea recta y, por tanto, es convexa. El conjunto de variables latentes para una parte sobre el conjunto de muestras positivas quedaría definido como

$$\mathbf{Z}_P = \{\mathbf{z}_{P_1}, \mathbf{z}_{P_2} \dots\}, \quad (8.27)$$

y la función de costo en (8.26) se convierte en

$$L_D(\mathbf{w}, \mathbf{Z}_P) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i\in P} \text{máx}\left\{0, 1 - \mathbf{w}^T\phi(\mathbf{x}_i,\mathbf{z}_{P_i})\right\} + C\sum_{i\in N} \text{máx}\left\{0, 1 + \underset{\mathbf{z}_{P_i}\in Z_P}{\text{máx}}\, \mathbf{w}^T\phi(\mathbf{x}_i,\mathbf{z}_{P_i})\right\}, \quad (8.28)$$

la cual tiene un término concavos y varios convexos pero es una mejora con respecto al problema original.

La optimización de $L_D(\mathbf{w}, \mathbf{Z}_P)$ sigue una estrategia dual en la que primero se fija $\mathbf{w}$ y se encuentran los mejores valores de las variables latentes para cada ejemplo positivo, tal que

$$\mathbf{z}_{P_i} = \arg \max_{\mathbf{z}_{P_i} \in \mathbf{Z}_P} \mathbf{w}^T \phi(x_i, \mathbf{z}_{P_i}), \tag{8.29}$$

y luego se optimiza $L_D(\mathbf{w}, \mathbf{Z}_P)$ sobre $\mathbf{w}$, para todas las muestras positivas y negativas, dejando $\mathbf{Z}_P$ fija, tal que

$$\mathbf{w}_{t+1} = \arg \min_{\mathbf{w}} L_D(\mathbf{w}, \mathbf{Z}_P). \tag{8.30}$$

El primer paso involucra solucionar (8.18) y encontrar las mejores posiciones de las partes durante la detección. La segunda parte requiere optimizar (8.28) aprovechando que se ha supuesto una variable latente. Sin embargo, el problema permanece sesgado, pues el número de ejemplos negativos excede por mucho a los ejemplos positivos. Esto puede significar que se optimiza realmente para los ejemplos negativos. Es decir, dado que en el mundo hay muchísimos más negativos que positivos, puede ser que la ganancia de optimizar para los positivos sea marginal y no impacte en el resultado. Para ello, se sigue una estrategia de *bootstrapping* donde un subconjunto de ejemplos negativos se usa para entrenamiento. Con los ejemplos negativos mal clasificados se construye un conjunto de ejemplos negativos difíciles. Según se necesite, este proceso se repite varias veces.

8.5. Implementación

Una implementación completa y muy eficiente del algoritmo de Dalal y Triggs se encuentra en la librería de OpenCV[1]. Usando las rutinas de Matlab desarrolladas por Kota Yamaguchi[8], y dada una imagen $\mathbf{I}$, las instrucciones básicas serían las que se ilustran en el Algoritmo 15. Un ejemplo de la ejecución de este programa se muestra en la Figura 8.2.

Por su lado, Felzenszwalb *et al.*[5] ponen su implementación disponible en la dirección `http://www.cs.berkeley.edu/~rbg/latent`. Su código incluye detectores para aviones, bicicletas, pájaros, botes, botellas, autobuses, automóviles, gatos, sillas, vacas, mesas de cena, perros, caballos, motocicletas, personas, plantas en macetas, borregos, sillones, trenes, y monitores de TV. En la Figura 8.8 se incluye un ejemplo del detector de personas.

Sumario

Los Histogramas de Gradientes Orientados (HOG) representan una estrategia muy popular para caracterizar objetos y como un paso previo a su detección visual por computadora. Aquí revisamos el trabajo de Dalal y Triggs[4] para el caso en que los objetos aparecen de forma completa en la imagen y el trabajo de Felzenszwalb *et al.*[5], donde se considera la extensión que permite la detección de objetos cuando algunas de sus partes no son visibles. En el proceso de clasificación, ambos autores utilizan un clasificador basado en la máquina de soporte vectorial [3], como una forma de distinguir entre los objetos que pertenecen o no a la clase de interés. En el caso de Felzenszwalb *et al.*, ellos hacen la importante observación

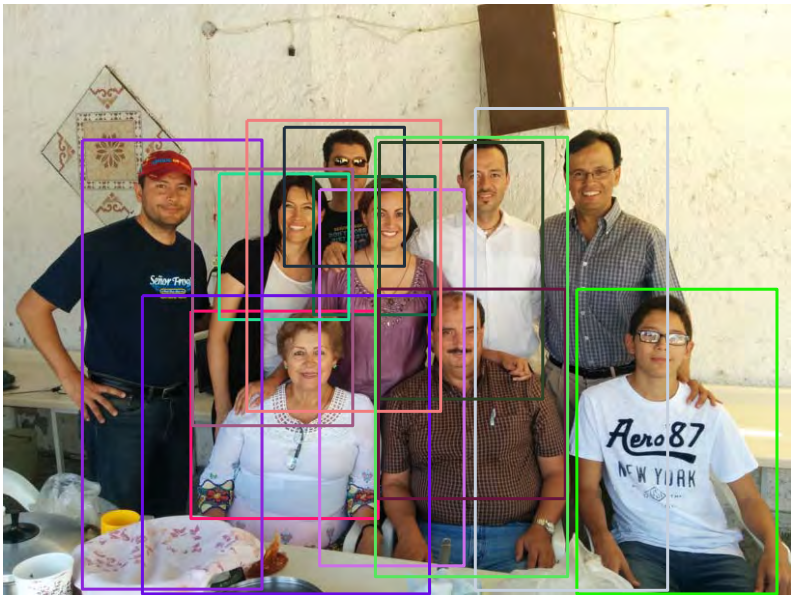


Figura 8.8: Detector de Felzenszwalb *et al.*[5] aplicado a personas. Note cómo las personas son detectadas aún cuando alguna parte de ellas no sea visible.

de que cualquiera que sea el objeto de interés son muchos más los ejemplos negativos que los positivos. Esto les lleva a recomendar estrategias de entrenamiento más efectivas.

Ejercicios

1. Ilustra la *Hinge Loss Function* para las muestras positivas y negativas. ¿ Cuál es la diferencia entra ambas?

2. Implementa el Algoritmo 15 de Dalal-Triggs para la detección de personas.

3. Implementa el Algoritmo de Felzenswalb *et al.* para la detección de personas por partes.

4. ¿De qué tamaño es el descriptor de personas de Felzenswalb *et al.*[5] ?

Bibliografía

[1] Bradski, G.: *The OpenCV Library*. Dr. Dobb's Journal of Software Tools, 2000.

[2] Burges, Christopher: *A Tutorial on Support Vector Machines for Pattern Recognition*. Data Mining and Knowledge Discovery, 2(2):121–167, 1998.

[3] Cortes, Corinna y Vladimir Vapnik: *Support-Vector Networks*. Machine Learning, 20(3):273–297, 1995.

[4] Dalal, Navneet y Bill Triggs: *Histograms of Oriented Gradients for Human Detection*. En *IEEE Conference on Computer Vision and Pattern Recognition*, volumen 1, páginas 886–893, 2005.

[5] Felzenszwalb, Pedro, Ross Girshick, David McAllester y Deva Ramanan: *Object Detection with Discriminatively Trained Part-based Models*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645, 2010.

[6] Fischler, Martin y Robert Elschlager: *The Representation and Matching of Pictorial Structures*. IEEE Transactions on computers, 22(1):67–92, 1973.

[7] Girshick, Ross: *Deformable part models*. `http://vision.stanford.edu/teaching/cs231b_spring1213/slides/dpm-slides-ross-girshick.pdf`, 2013.

[8] Yamaguchi, Kota: *mexopencv*. `http://vision.is.tohoku.ac.jp/~kyamagu/software/mexopencv/`, October 2014.

CAPÍTULO 9

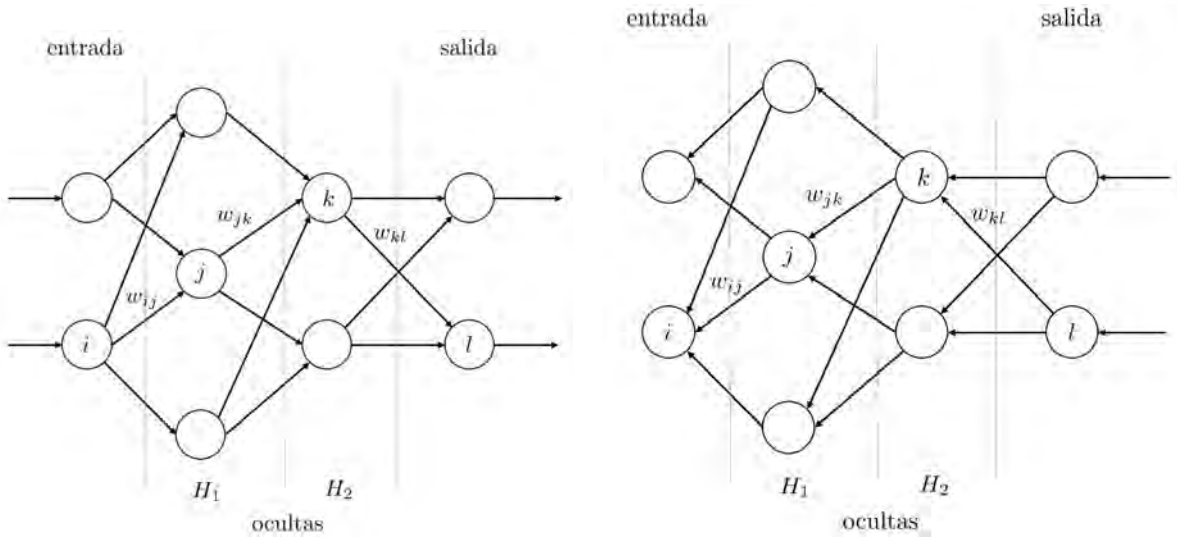
Redes Neuronales Convolucionales (CNN)

Desde hace algunos años, las CNN han tomado gran relevancia por su alto nivel de desempeño particularmente en tareas de clasificación. Especialmente notable han sido los resultados obtenidos en el *Imagenet Large Scale Visual Recognition Challenge*(ILSVRC), donde según algunas medidas han logrado resultados superiores a los humanos en lectura de signos de tráfico[4], reconocimiento de rostros[2] o interpretación de imágenes retinales[15]. La ventaja esencial de las CNN sobre otros modelos de detección es que la selección de características se realiza de forma automática. Asimismo, se ha mostrado que, con respecto a las redes neuronales (NN) tradicionales, el número de capas es un parámetro muy importante. Esto se traduce en que es normal que el número de parámetros por ajustar ronde en los millones, el tiempo de entrenamiento ronde en las semanas y el número de muestras para el entrenamiento este en el orden de las decenas de millones. Ciertamente, esto comienza a colocar la disciplina más allá de la computadora de escritorio normal. Sin embargo, los resultados obtenidos parecen apuntar a que continuará habiendo un fuerte impulso en esta dirección.

Algunas arquitecturas conocidas incluyen: LeNet, la primera CNN, desarrollada para identificar dígitos por Yan LeCun en los 1990's[11]; AlexNet, la CNN que las popularizó con su desempeño en *Imagenet Large Scale Visual Recognition Challenge*(ILSVRC) en 2012[9]; ZFNet, la ganadora en el ILSVRC 2013, donde esencialmente se afinaron los parámetros de AlexNet[20]; GoogleNet, la ganadora del ILSVRC 2014 reduciendo el número de parámetros[19]; VGGNet, la cual con sus 16 capas CONV/FC y solo ejecutaba convoluciones de 3×3 y sumarizaciones 2×2, mostró que la profundidad juega un rol primordial en las CNN[18]; ResNet, con 138M de parámetros, resultó la ganadora del ILSVRC 2015[6]; de forma interesante, en el 2016 los ganadores ensamblaron seis modelos, por lo que se puede decir que no hubo avances significativos en las arquitecturas propuestas[17].

9.1. Modelo de *Feed Forward* y Entrenamiento por *Back Propagation*

Las NN representaciones de una función no lineal mediante la cual ciertas entradas son transformadas en valores de salida. Regularmente son modeladas mediante un grafo acíclico (ver Figura 9.1), aunque hay un importante grupo de arquitecturas llamado redes neuronales recurrentes (RNN) donde se admiten ciclos[13]. Las redes están organizadas en una capa de entrada, una de salida y entre ellas varias llamadas ocultas.



(a) *Feed Forward* (b) *Back Propagation*

Figura 9.1: Representación gráfica de las etapas relacionadas a la evaluación de la función expresada en la red neuronal para una cierta entrada (a) y el cálculo de su derivada (b). En la ilustración no se muestra el nodo de sesgo en cada capa.

9.1.1. Modelo de *Feed Forward*

Una neurona puede verse como recibiendo de la capa anterior un conjunto de señales, $\mathbf{x} = (x_1, \ldots, x_n)$, pesadas por un factor $\mathbf{w} = (w_1, \ldots, w_n)$. Dentro de la neurona estas entradas son sumadas, $w_1x_1 + \cdots + w_nx_n$, y un valor de sesgo b es adicionada. Como resultado, la neurona genera una señal $f(z)$ que denota la respuesta de la neurona a las entradas. Es decir, la salida de una neurona está dada por $f(\mathbf{w}^T\mathbf{x} + b)$. Estas operaciones son repetidas en subsiguientes capas. Por ejemplo, en la Figura 9.1(a), las operaciones de *Feed Forward* tendrían la siguiente secuencia

$$z_j = \sum_{i \in \text{entradas}} w_{ij}x_i \rightarrow y_j = f(z_j) \rightarrow$$
$$z_k = \sum_{j \in H_1} w_{jk}y_j \rightarrow y_k = f(z_k) \rightarrow$$
$$z_l = \sum_{k \in H_2} w_{kl}y_k \rightarrow y_l = f(z_l), \quad (9.1)$$

donde x_i corresponde a los valores de entrada y y_l corresponde a las salidas.

Las funciones de activación tienen el papel de realizar el mapeo a espacios no lineales. En su operación, emiten la salida de las neuronas en función de las entradas que se reciben. Son esenciales para el proceso de entrenamiento. Algunas de ellas incluyen la sigmoide, la RELU, la *leaky* RELU y la *MaxOut*[16]. Hace algunos años, la sigmoide, definida como

$$\sigma(x) = \frac{1}{1 + \exp(-x)}, \quad (9.2)$$

era muy común. Sin embargo, la función no está centrada en cero. Para aliviar este problema se sugirió la *tangente hiperbólica*, definida como

$$\tanh(x) = 2\sigma(2x) - 1, \tag{9.3}$$

la cual está centrada en cero. Sin embargo, en ambos casos, durante el etapa de ajuste de los pesos por *back propagation*, cuando las activaciones se saturan en las colas los gradientes se desvanecen. Actualmente, la función más utilizada es la unidad lineal rectificada (ReLU)[10], la cual se define como

$$\text{ReLU}(x) = \text{máx}(0, x). \tag{9.4}$$

La función ha mostrado acelerar la convergencia y es sencilla de implementar.

Se ha mostrado que las NN son aproximadores universales, *i.e.*, dada una función $f(x)$ y una constante $\epsilon > 0$, existe una NN con una capa oculta $g(x)$ tal que $\forall x|f(x) - g(x)| < \epsilon$[7]. En ese sentido, uno pensaría que NN de pocas capas evitarían *overfitting* de las funciones. Sin embargo, se ha visto experimentalmente que NN de tres capas trabajan mejor que las de 2 (aunque no se ve un efecto tan dramático para 4, 5 o 6 capas). En el caso de CNN se ha visto que la profundidad es un componente importante, donde 10 capas son comunes[18]. Para evitar *overfitting*, la función objetivo se regulariza. Esto es preferido a disminuir el número de neuronas.

9.1.2. Entrenamiento por *Back Propagation*

Es generalmente aceptado que hay cuatro formas de obtener las derivadas de una ecuación: Hacer el cálculo manual, realizar el cálculo numérico, obtener el resultado simbólico, o hacer diferenciación automática[1]. Aquí revisamos este último método y nos familiarizamos con el algoritmo de aprendizaje denominado propagación hacia atrás o *back propagation*.

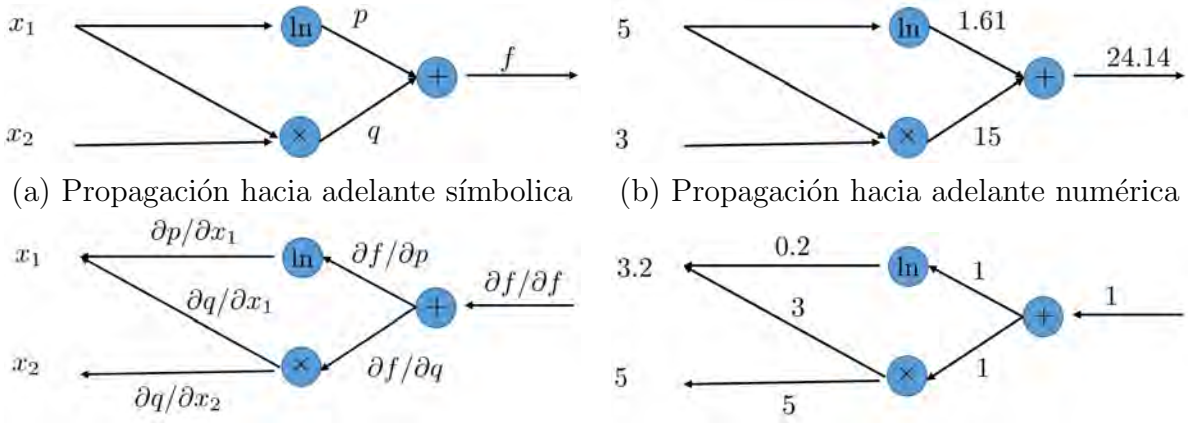
Dada una función $f(\mathbf{x})$, estamos interesados en obtener el gradiente $\nabla f(\mathbf{x})$. Por ejemplo, si $f(x, y)$, entonces $\nabla f = (\partial f/\partial x, \partial f/\partial y)^T$. Aquí estamos interesados en aplicar la regla de la cadena para resolver el problema. Por ejemplo, considere la expresión

$$f(x_1, x_2) = \ln x_1 + x_1 x_2. \tag{9.5}$$

Esta expresión se puede descomponer en operaciones sencillas tales como sumas, multiplicaciones o derivadas de funciones de una variable, tal como

Cambio de variable	Derivada parcial
$p = \ln x_1$	$\partial p/\partial x_1 = 1/x_1$
$q = x_1 x_2$	$\partial q/\partial x_1 = x_2$ $\partial q/\partial x_2 = x_1$
$f = p + q$	$\partial f/\partial p = 1$ $\partial f/\partial q = 1$

(9.6)



(a) Propagación hacia adelante símbolica (b) Propagación hacia adelante numérica

(c) Propagación hacia atrás símbolica (d) Propagación hacia atrás numérica

Figura 9.2: Ejemplo de obtención de la derivada de la función $f(x_1, x_2) = \ln x_1 + x_1 x_2$ mediante propagación hacia atrás. La función es descompuesta en términos de funciones sencillas, aplicando la regla de la cadena. Para la evaluación del gradiente se parte del final a contraflujo.

La obtención del gradiente se puede realizar mediante la regla de la cadena. Utilizando las expresiones anteriores podemos llegar al siguiente resultado:

$$\nabla f(x_1, x_2) = \begin{pmatrix} \partial f/\partial x_1 \\ \\ \partial f/\partial x_2 \end{pmatrix} = \begin{pmatrix} \frac{\partial q}{\partial x_1}\frac{\partial f}{\partial q} + \frac{\partial p}{\partial x_1}\frac{\partial f}{\partial p} \\ \\ \frac{\partial q}{\partial x_2}\frac{\partial f}{\partial q} \end{pmatrix} \tag{9.7}$$

En una red neuronal, *back propagation* sirve para ajustar el valor de los pesos de la red. Para ello, se supone que se cuenta con un conjunto $\{\mathbf{X}, \mathbf{t}\}$ de observaciones $\mathbf{X}_{n\times m} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$, con $\mathbf{x}_i = (x_1, \ldots, x_n)^T$ y sus correspondientes m valores de salida $\mathbf{t} = (t_1, \ldots, t_m)^T$. Si asumimos que nuestra función de costo está dada por la diferencia entre el valor de salida de la red y el valor deseado, el error en cierto momento puede representarse como la suma de las diferencias al cuadrado, tal como

$$E = \sum_{l=1}^{m} (y_l - t_l)^2. \tag{9.8}$$

Utilizando como ilustración la arquitectura de red presentada en la Figura 9.1, la propagación del error puede describirse como

$$\begin{aligned} \frac{\partial E}{\partial y_l} = y_l - t_l \rightarrow \frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial y_l} \cdot \frac{\partial y_l}{\partial z_l} \rightarrow \\ \frac{\partial E}{\partial y_k} = \sum_{l \in \text{salida}} w_{kl} \frac{\partial E}{\partial z_l} \rightarrow \frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial z_k} \rightarrow \\ \frac{\partial E}{\partial y_j} = \sum_{k \in H_2} w_{jk} \frac{\partial E}{\partial z_k} \rightarrow \frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_j}, \end{aligned} \tag{9.9}$$

donde recordamos que $y = f(z)$ y la derivada $\frac{\partial y}{\partial z}$ puede obtenerse evaluando la derivada de $f(z)$. Los pesos se modifican considerando como varia el error junto con ellos. Es decir, evaluando la expresión

$$\frac{\partial E}{\partial w_j k} = \frac{\partial E}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{jk}}, \tag{9.10}$$

donde el término $\dfrac{\partial z_j}{\partial w_{jk}}$ puede expresarse como

$$\frac{\partial z_k}{\partial w_j k} = \frac{\partial}{\partial w_{jk}} \left(\sum_{r=1}^{n} w_{jr} y_r \right) = y_j, \tag{9.11}$$

puesto que en la sumatoria solo un término depende de w_{jk}. De esta forma, los pesos son modificados usando la ecuación

$$w_{jk}^{(+)} = w_{jk}^{(-)} + \alpha \frac{\partial E}{\partial w_j k} = w_{jk}^{(-)} + \alpha y_j \frac{\partial E}{\partial z_k}, \tag{9.12}$$

para una constante de aprendizaje α.

9.2. Redes Neuronales Convolucionales (CNN)

Las CNN son una arquitectura de red neuronal especializada en el trabajo con imágenes. Por su origen, comparten con las NN algunos aspectos fundamentales. Por ejemplo, en ambos casos, el aprendizaje de los pesos es mediante *back propagation*. En ambos casos una neurona recibe entradas que son pesadas mediante un producto punto y con el resultado se realiza una transformación no lineal. Asimismo, en ambos casos hay una función de pérdida diferenciable.

En una CNN las neuronas son agrupadas en capas tridimensionales (que tienen una anchura, altura y profundidad) que se conectan una tras otra mediante una función diferenciable que puede requerir o no parámetros. Por ejemplo, una imagen de $m \times n \times 3$ requiere una capa de entrada de $3mn$ neuronas. Las neuronas en las capas siguientes serán conectadas a una pequeña región de la capa anterior, no a todas las neuronas. Al final, la capa de salida tendrá dimensiones $1 \times 1 \times K$, correspondiendo a las K clases de interés.

Hay tres tipos de capas en una CNN: Convoluciones (CONV), Sumarizado (POOL) y Completamente Conectadas (FC). Por ejemplo, supóngase que se tiene una CNN que toma imágenes de $32 \times 32 \times 3$, pretende distinguir entre 10 clases y tiene la siguiente arquitectura: INPUT-CONV-RELU-POOL-FC. Las capas tienen la siguiente interpretación (ver Figura 9.3):

- INPUT. Es una capa de $32 \times 32 \times 3$, la cual tendrá la imagen de color.
- CONV. Esta capa calculará la salida de las neuronas en una región de la imagen de entrada. En cada región se calcula un producto punto entre los pesos y una pequeña región conectada a la entrada. Si se tienen 8 filtros, se tendrá un volumen de $32 \times 32 \times 8$ unidades.

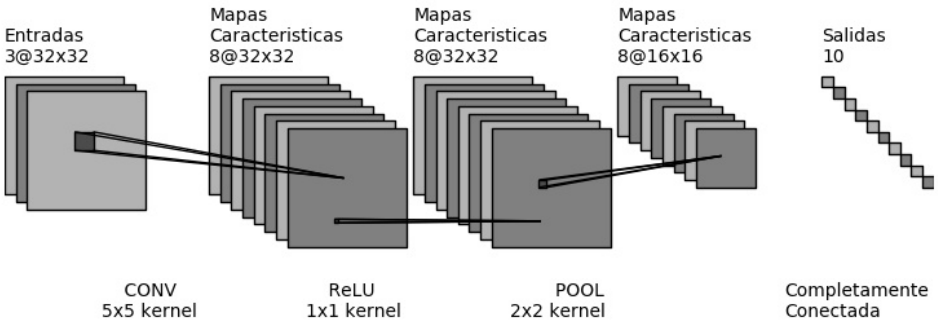


Figura 9.3: Ejemplo de arquitectura de una CNN con CONV→ ReLU → POOL. Una imagen de color con 32×32 pixeles de resolución es convolucionada con 8 filtros. Enseguida, este banco de imágenes pasa por una función no lineal ReLU. Luego, una función de POOL extrae las respuestas más grandes reduciendo la resolución de los datos espaciales a la mitad. Figura creada con `draw_convnet` de Gavin Weiguang Ding.

- RELU. En esta capa se aplica la función de activación máx$(0, x)$, elemento por elemento. Esto deja el volumen en $32 \times 32 \times 8$ unidades.
- POOL. En esta capa se hace una sumarización sobre la parte espacial de la imagen, seleccionado solo algunos elementos de ella. Por ejemplo, puede ser que el volumen resultante sea de $16 \times 16 \times 8$.
- FC. En esta capa se calculan los scores resultando en un volumen de tamaño $1 \times 1 \times 10$.

En este ejemplo, las capas CONV/FC harán transformaciones que son función del volumen de entrada, los pesos y los sesgos. Las capas son entrenadas con gradiente descendente. Las capas RELU/POOL implementan una función fija.

Las CNN pueden construirse con capas CONV, POOL, RELU y FC. Por ejemplo

$$\text{INPUT} \to [[\text{CONV} \to \text{RELU}] \times N \to \text{POOL?}] \times M \to [\text{FC} \to \text{RELU}]K \to \text{FC}, \quad (9.13)$$

donde $\times$ indica repetición, POOL? una capa de sumarización opcional, $N \geq 0$ (usualmente $N \leq 3$), $M \geq 0, K \geq 0$ (usualmente $K < 3$). Por regla de dedo, se prefiere muchos CONV con filtros pequeños que un CONV con un campo receptivo grande. Esto debido a que entre cada CONV hay normalmente una función no lineal.

Capa CONV. Durante la etapa de procesamiento de la red, las entradas con convolucionadas con filtros sobre las dimensiones de anchura y altura. Esto genera un mapa de activación en 2D. La convolución se realiza entre las entradas y los pesos. La CNN aprenderá los filtros más apropiados que se activarán cuando alguna característica, tal como un contorno en alguna orientación o una región de un cierto color esté presente en la primera capa, o algunas

Figura 9.4: Algunos ejemplos contenidos en la base de datos de MNIST.

características más complejas en capas posteriores de la red. Cada filtro aplicado produce mapas de activación bidimensionales separados. La extensión de la conectividad es llamado campo receptivo de la neurona y se regula con un hiperparámetro. Las conexiones son locales en espacio pero siempre completas sobre la profundidad del volumen de entrada.

Capa POOL. La función de la capa POOL es reducir gradualmente el tamaño espacial de la representación, reducir el número de parámetros y evitar *overfitting*. En común insertar una capa de sumarización entre capas de convolución. La capa de sumarización usa la operación máx. La forma más común es aplicarla sobre una vecindad de 2×2 con un paso de 2, logrando con ello descartar el 75 % de las activaciones.

9.3. Ejemplo de MNIST

MNIST es el ejemplo *Hola Mundo* para CNN. MNIST se compone de decenas de miles de imágenes correspondientes a dígitos numéricos escritos a mano (ver Figura 9.4). La base de datos puede dividirse cómodamente en decenas de miles de imágenes para entrenamiento, prueba y validación. Cada imagen tiene una resolución de 28×28 pixeles, cuyo valor para pixel es un número real entre cero y uno.

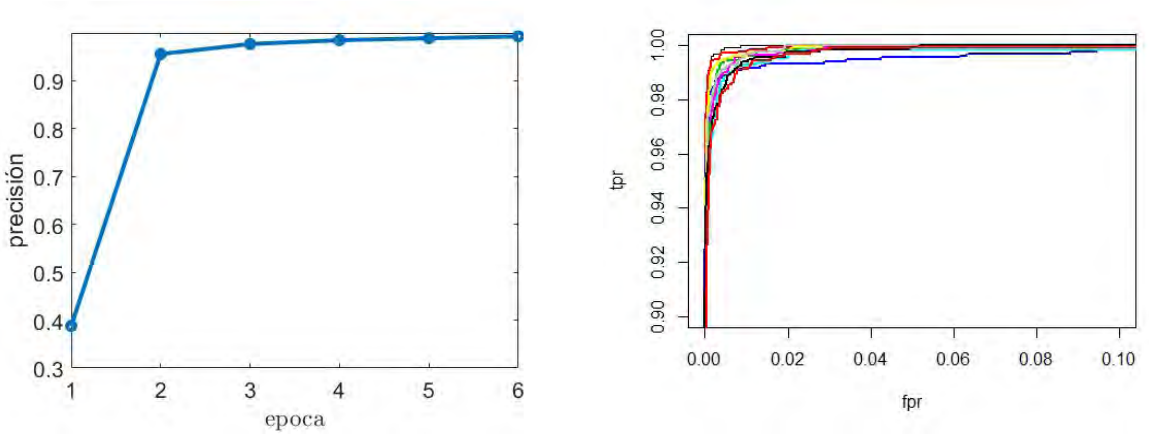
Para este ejercicio utilizamos la CNN LeNet, la cual se define como[11]

$$\text{INPUT} \to [[\text{CONV} \to \text{TANH}] \to \text{POOL}] \times 2 \to [\text{FC} \to \text{TANH}]10 \to \text{FC}, \tag{9.14}$$

donde TANH es la tangente hiperbólica, como en (9.3). Para este ejercicio en particular, utilizamos 42,000 dígitos de la base de datos MNIST. De ellos, escogimos el 70 % (29,399)

dígitos	0	1	2	3	4	5	6	7	8	9
muestras	1,272	1,389	1,273	1,316	1,285	1,104	1,279	1,258	1,186	1,239

Cuadro 9.1: Distribución de dígitos en la muestra utilizada para probar la CNN LeNet con la base de datos de MNIST



(a) (b)

Figura 9.5: MNIST usando la CNN LeNet. En (a) se muestra el incremento de la precisión por epoca de entrenamiento para la base de datos MNIST usando la CNN LeNet. Luego, en (b), se muestra la curva ROC para cada dígito clasificado. El promedio del área bajo la curva es 0.9997

de las imágenes para entrenar y el 30 % (12,601) imágenes para probar. El tiempo para entrenar una red de este tipo es de 196 segundos (3 minutos y 16 segundos) en el CPU. El uso de una GPU puede reducir en órdenes de magnitud el tiempo de procesamiento. Para el entrenamiento se utilizaron grupos de 100 imágenes escogidos aleatoriamente para calcular el gradiente, en una técnica conocida como *stochastic gradient descend*[3]. La tasa de aprendizaje α fue de 0.05. Para este ejemplo, se realizaron 6 iteraciones sobre los datos de entrenamiento. El incremento de la precisión con cada iteración se muestra en la Figura 9.5(a). Para entrenamiento, la distribución de dígitos sigue la distribución que se muestra en la Tabla 9.1, la cual exhibe homegeneidad. La evaluación del proceso de entrenamiento se muestra en la Figura 9.5(b)

En un problema de reconocimiento de patrones, mucho se puede entender de los datos por clasificar utilizando visualización de datos. Sin embargo, la visualización de datos multidimensionales es un reto. Para poder tener una intuición de esto, considere a cada pixel como representando una característica dentro de un espacio multidimensional y al valor del pixel como una posición dentro de esta dimensión. Esto resultará al conjunto de datos como un subespacio dentro del espacio de las todas las imágenes posibles. El *t-Distributed Stochastic Neighbor Embeeding* (t-SNE)[14] es una técnica de visualiación de datos que preserva su topología, *i.e.*, trata de tener los mismos vecinos, independientemente de la dimensionalidad del espacio donde se visualice. De su representación visual (ver Figura 9.6), uno podría con-

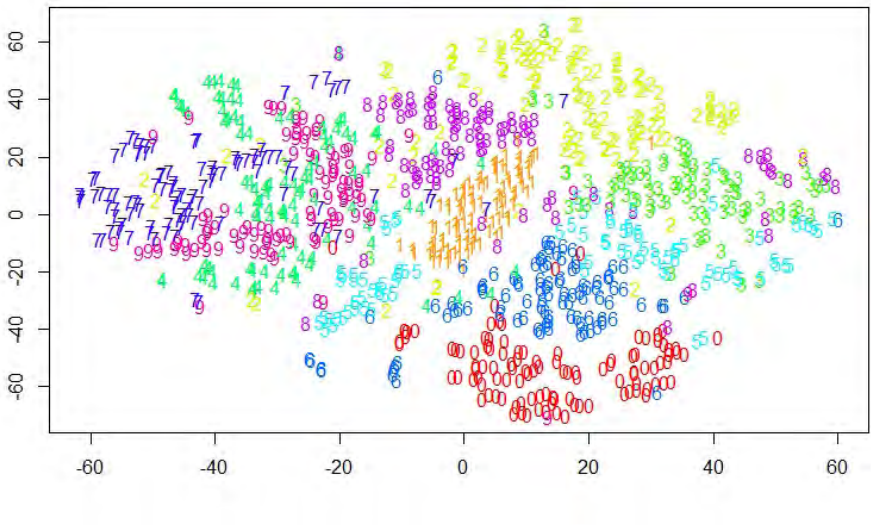


Figura 9.6: Visualización t-SNE de la base de datos del MNIST. Aquí se representa una muestra aleatoria de 1,000 dígitos de la base de datos del MNIST visualizados mediante t-SNE.

cluir que MNIST es una base de datos estructurada donde la distinción entre clases permite un alto nivel de desempeño de los clasificadores.

9.4. Clasificación Lineal

Una red neuronal funciona como una transformación no lineal de las entradas. La expectativa es que en ese nuevo espacio, uno pueda distinguir las clases mediante un hiperplano. En las arquitecturas actuales, las opciones son utilizar ya sea una SVM multiclase o un clasificador Softmax. Aquí exploramos ambos.

Primero, partimos del supuesto de que si $\mathbf{x}_i \in \mathcal{R}^D$, para $i = 1, \ldots, N$, representa una imagen que tiene una etiqueta $y_i \in \{1, \ldots, K\}$, entonces $\mathbf{f}$ podría ser definida como una función que mapea de pixeles al *score* de pertenencia a la clase,

$$\mathbf{f} : \mathcal{R}^D \rightarrow \mathcal{R}^K. \tag{9.15}$$

Un ejemplo de esta función podría ser un clasificador lineal, definido como

$$\mathbf{f}(\mathbf{x}_i, \mathbf{W}) = \mathbf{W}\mathbf{x}_i + \mathbf{b}, \tag{9.16}$$

donde $\mathbf{W}_{K \times D}$ es una matriz de pesos y $\mathbf{b}_{K \times 1}$ un vector de sesgo. En este sentido, las hileras de $\mathbf{W}$ pueden ser interpretadas como la descripción de las líneas que separan las clases. Abusando de la notación, antes de proceder a realizar las operaciones de clasificación podríamos incluir una columna de unos a $\mathbf{W}$ y un uno a cada vector $\mathbf{x}_i$ de tal forma de utilizar la representación

$$\mathbf{f}(\mathbf{x}_i, \mathbf{W}, \mathbf{b}) = \mathbf{W}\mathbf{x}_i. \tag{9.17}$$

9.4.1. Clasificador SVM

Con respecto al problema de clasificación multiclase usando SVM, el criterio de asignación es que el *score* de la clase correcta debe ser mayor que el resto por al menos un valor de margen Δ[5]. Si definimos la función de *score* como

$$s_j = f(\mathbf{x}_i, \mathbf{w}_j) = \mathbf{w}_j^T \mathbf{x}_i, \tag{9.18}$$

donde $\mathbf{w}_j$ corresponde a la j-ésima línea de discriminación, o j-ésima columna de $\mathbf{W}^T$, entonces la función de pérdida SVM para la i-ésima clase podría definirse como

$$\begin{aligned} L_i &= \sum_{j \neq y_i} \text{máx}(0, s_j - s_{y_i} + \Delta), \\ &= \sum_{j \neq y_i} \text{máx}(0, \mathbf{w}_j^T \mathbf{x}_i - \mathbf{w}_{y_i}^T \mathbf{x}_i + \Delta), \end{aligned} \tag{9.19}$$

conocida como *hinge loss function*. Por ejemplo, si se tuvieran tres clases con scores $\mathbf{s} = (13, -7, 11)$, un valor mínimo de margen $\Delta = 10$, y la primera fuera la clase verdadera, la función de pérdida sería

$$\begin{aligned} L_1 &= \text{máx}(0, s_2 - s_1 + \Delta) + \text{máx}(0, s_3 - s_1 + \Delta), \\ &= \text{máx}(0, -7 - 13 + 10) + \text{máx}(0, 11 - 13 + 10) = 21. \end{aligned} \tag{9.20}$$

Dada la interpretación geométrica de la ecuación de la línea recta, hay un factor de escala que hay que ajustar. En SVM multiclase, se quiere minimizar la magnitud de $\mathbf{W}$, $R(\mathbf{W})$, definida como

$$R(\mathbf{W}) = \sum_{k=1}^{K} \sum_{l=1}^{D} w_{kl}^2. \tag{9.21}$$

Así pues, la función de pérdida queda definida como (ver Figura 9.7)

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda R(\mathbf{W}), \tag{9.22}$$

donde λ es el multiplicador de Lagrange y N es el número de muestras. En la práctica, el valor de Δ se deja igual a uno y se deja el peso de la regularización al factor λ.

9.4.2. Clasificador Softmax

El clasificador Softmax es una generalización del clasificador binario de regresión logística para el caso multidimensional. La función de pérdida cambia a

$$L_i = -\log \left(\frac{\exp(f_{y_i})}{\sum_{j=1}^{K} \exp(f_j)} \right) = -f_{y_i} + \log \sum_{j=1}^{K} \exp(f_j), \tag{9.23}$$

donde f_j es el j-ésimo elemento del vector $\mathbf{f} = (f_1, \ldots, f_K)^T$ de *scores* para las clases. La función de pérdida completa corresponde a (9.22).

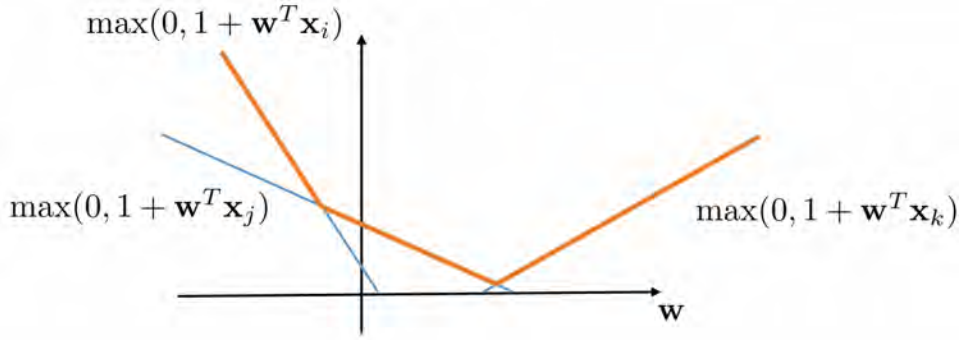


Figura 9.7: Clasificación mediante SVM multiclase. El clasificador SVM optimiza la llamada *hinge loss function* (ver Figura 8.6). Para el conjunto de los parámetros bajo consideración esto resulta en una función cóncava.

9.5. Aspectos Estáticos de una CNN

Antes de proceder a entrenar la CNN es recomendable preprocesar los datos con la finalidad de centralizarlos y que su valor medio sea igual a cero y normalizar las entradas para que su desviación estándar sea uno[12].

9.5.1. Preproceso de los Datos

Sea $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, el conjunto de datos, cuya dimensionalidad es D. El centrado requiere sustraer el valor medio, $\overline{\mathbf{x}}$, lo cual puede llevarse a cabo con la operación

$$\mathbf{X}_m = \mathbf{X} - \overline{\mathbf{x}}\mathbf{1}^T, \tag{9.24}$$

donde $\mathbf{1}$ es un vector de N dimensiones.

Para la normalización se requiere que las características tengan desviación estándar unitaria en cada una de las dimensiones. Para ello, sea $\sigma = (\sigma_1, \ldots, \sigma_D)$ el vector de $D \times 1$ con las desviaciones estándar sobre las características. En adición sea la representación de $\mathbf{X} = (\mathbf{u}_1, \ldots, \mathbf{u}_D)^T$ la representación de los renglones de $\mathbf{X}$. La operación de centralización y normalización sobre las características sería

$$\hat{\mathbf{u}} = \frac{\mathbf{u}_i - \overline{\mathbf{u}}_i}{\sigma_i}. \tag{9.25}$$

9.5.2. Inicialización de los Pesos

Si los pesos $\mathbf{W}$ se inicializan a ceros, las neuronas calcularían la misma salida, con lo cual tendrían los mismos gradientes y las mismas actualizaciones de parámetros. Hay varios esquemas de inicialización. Uno de ellos consiste en proporcionar un pequeño valor aleatorio con función de distribución de probabilidad Gaussiana que rompa la simetría. La distribución de las salidas de una neurona inicializada aleatoriamente tiene varianza que crece con el número de entradas. Lo conveniente es normalizar la varianza de cada neurona a uno

escalando su vector de pesos por la raíz cuadrada del número de entradas. Así los pesos se inicializan con

$$\mathbf{W} = \frac{\text{matrix de } n \times n \text{ con ruido Gaussiano}}{\sqrt{n}}, \tag{9.26}$$

donde n es el número de entradas.

9.5.3. Regularización

La regularización controla la capacidad de la NN para evitar `overfitting`. Las opciones de regularización se aplican sobre (9.22) e incluyen[12, 8]:

Regularización Euclideana. Sigue la formulación $\frac{1}{2}\lambda w^2$, donde λ es el grado de la regularización. Su efecto es usar todas las entradas un poco, contrario a usar algunas pocas entradas mucho.

Regularización de Manhattan. Sigue la formulación $\lambda|w|$, en la práctica algunos de los pesos se hacen cero, haciendo una suerte de selección de características.

MaxNorm. Estas restricciones forzan a que la magnitud de $\mathbf{w}$ no sobrepase de un cierto valor absoluto. En la práctica, la actualización de los parámetros es hecha como de costumbre y entonces el vector de pesos es limitado para $\|\mathbf{w}\|_2 < c$, para valores típicos de c de 2 o 4.

Dropout. Durante entrenamiento, `dropout` mantiene una neurona activa con una probabilidad P (un hiperparámetro) o de otra forma la pone a cero.

El `dropout` solo de aplica en la etapa de entrenamiento. En esa etapa, el valor esperado es $xp + (1-p)0$. Sin embargo, ya en la etapa de pruebas se tiene que hacer $x \to px$ para mantener el mismo valor esperado.

En la práctica, es común usar una regularización L_2 cuya constante λ se calcula vía *cross-validation*.

9.6. Aspectos Dinámicos de una CNN

El entrenamiento de una CNN es muy largo y puede estar en el orden de días. Algunos aspectos que pueden ser monitoreados durante ese tiempo incluyen[12, 8]:

- Hay que verificar que se obtiene la pérdida esperada al inicializar con valores pequeños para los parámetros. Es mejor verificar el término de pérdida de los datos por separado. El incrementar la contribución del término de regularización se debería incrementar el valor de la pérdida. Conviene, por ejemplo, tomar un pequeño subconjunto de datos y verificar que se puede llegar a un costo cero, al hacer *overfitting* sobre esos datos.

- Durante el proceso conviene estar verificando el comportamiento, por ejemplo con gráficas que se actualicen en cada época.

- La función de pérdida es evaluada en subconjuntos de datos de entrenamiento, por lo que puede mostrar oscilaciones. En todo caso, el cociente entre el error entre el conjunto de entrenamiento y el conjunto de validación puede ser un indicador. Por ejemplo, si hay una variación pequeña sobre el desempeño cuando analizamos el conjunto de validación puede significar *overfitting* en los datos de entrenamiento.

- El cociente entre el valor de las actualizaciones de los pesos y la magnitud de los gradientes debe ser aproximadamente 10^{-3}. Si es mayor, la tasa de aprendizaje puede ser muy alta. Si es menor, la tasa de aprendizaje puede ser muy baja.

- Una mala inicialización puede reducir o parar el proceso de aprendizaje. Una forma de hacer el diagnóstico de esta situación es graficar los histogramas de activación o gradiente para todas las capas de la NN.

- Las características de la primera capa pueden visualizarse y muestran las clases que se desea aprender.

Sumario

Las CNN han significado una revolución en el área de visión por computadora, al permitir alcanzar niveles de desempeño, en algunas tareas, comparables o superiores a los humanos. Sin embargo, su aplicación requiere de grandes cantidades de datos, procesadas con recursos computacionales de alto rendimiento por varios días. Las CNN utilizan para su operación el modelo de *feed forward* y para su entrenamiento el *back propagation*. La primera evalua una función no lineal para un conjunto de datos, tal que en ese espacio transformado las clases sean separables por un hiperplano. La segunda calcula el gradiente con la finalidad de ajustar los pesos de la red neuronal. Las CNN se componen de un conjunto de capas en donde operaciones de convolución, mapeo no lineal y sumarización son seguidas de etapas de conectividad completa. La diferencia entre las redes neuronales tradicionales y las convolucionales es precisamente las etapas iniciales de convolución. La base de imágenes de dígitos del MNIST es utilizada como un ejemplo introductorio para las CNN. Al final de la CNN hay un proceso de clasificiación en donde comúnmente se usa o un SVM multiclase o un clasificador Softmax. Dado que cada operación de entrenamiento consume grandes cantidades de recursos de cómputo, algunas operaciones de preprocesamiento de los datos deben realizarse antes del entrenamiento. Asimismo, durante el entrenamiento es conveniente observar el desarrollo del proceso mediante para monitorear su convergencia.

Ejercicios

1. Instalar Caffe, MxNet, TensorFlow o alguna otra plataforma de CNN para implementar el ejemplo de MNIST para la clasificación de imágenes de dígitos escritos a mano.

2. Explorar algunas arquitecturas como la de AlexNet o VGG para la detección de objetos.

3. Comparar el uso de CPU contra GPU en el tiempo requerido para entrenar un modelo de CNN.

Bibliografía

[1] Baydin, Atilim Gunes, Barak Pearlmutter, Alexey Andreyevich Radul y Jeffrey Mark Siskind: *Automatic Differentiation in Machine Learning: A Survey*. arXiv preprint arXiv:1502.05767, 2015.

[2] Blanton, Austin, Kristen Allen, Tim Miller, Nathan Kalka y Anil Jain: *A Comparison of Human and Automated Face Verification Accuracy on Unconstrained Image Sets*. En *To appear in the CVPR Workshop on Biometrics*, 2016.

[3] Bottou, Léon: *Large-Scale Machine Learning with Stochastic Gradient Descent*. En *Proceedings of COMPSTAT*, páginas 177–186. Springer, 2010.

[4] Cireşan, Dan, Ueli Meier, Jonathan Masci y Jürgen Schmidhuber: *A Committee of Neural Networks for Traffic Sign Classification*. En *International Joint Conference on Neural Networks*, páginas 1918–1921. IEEE, 2011.

[5] Duan, Kai Bo y Sathiya Keerthi: *Which is the Best Multiclass SVM Method? An Empirical Study*. En *International Workshop on Multiple Classifier Systems*, páginas 278–285. Springer, 2005.

[6] He, Kaiming, Xiangyu Zhang, Shaoqing Ren y Jian Sun: *Deep Residual Learning for Image Recognition*. arXiv preprint arXiv:1512.03385, 2015.

[7] Hornik, Kurt, Maxwell Stinchcombe y Halbert White: *Multilayer Feedforward Networks are Universal Approximators*. Neural Networks, 2(5):359–366, 1989.

[8] Karpathy, Andrej: *Stanford University CS231n: Convolutional Neural Networks for Visual Recognition*. Noviembre 2016.

[9] Krizhevsky, Alex, Ilya Sutskever y Geoffrey Hinton: *Imagenet Classification with Deep Convolutional Neural Networks*. En *Advances in Neural Information Processing Systems*, páginas 1097–1105, 2012.

[10] LeCun, Yann, Yoshua Bengio y Geoffrey Hinton: *Deep Learning*. Nature, 521(7553):436–444, 2015.

[11] LeCun, Yann, Léon Bottou, Yoshua Bengio y Patrick Haffner: *Gradient-based Learning Applied to Document Recognition*. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[12] LeCun, Yann, Léon Bottou, Genevieve Orr y Klaus Robert Müller: *Efficient Backprop*. En *Neural Networks: Tricks of the Trade*, páginas 9–48. Springer, 2012.

[13] Lipton, Zachary, John Berkowitz y Charles Elkan: *A Critical Review of Recurrent Neural Networks for Sequence Learning*. arXiv preprint arXiv:1506.00019, 2015.

[14] Maaten, Laurens van der y Geoffrey Hinton: *Visualizing data using t-SNE*. Journal of Machine Learning Research, 9(Nov):2579–2605, 2008.

[15] Maninis, Kevis Kokitsi, Jordi Pont-Tuset, Pablo Arbeláez y Luc Van Gool: *Deep Retinal Image Understanding*. En *International Conference on Medical Image Computing and Computer-Assisted Intervention*, páginas 140–148. Springer, 2016.

[16] Mishkin, Dmytro y Jiri Matas: *All you Need is a Good Init*. arXiv preprint arXiv:1511.06422, 2015.

[17] Ouyang, Wanli, Xiaogang Wang, Cong Zhang y Xiaokang Yang: *Factors in Finetuning Deep Model for Object Detection with Long-Tail Distribution*. En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 864–873, 2016.

[18] Simonyan, Karen y Andrew Zisserman: *Very deep convolutional networks for large-scale image recognition*. arXiv preprint arXiv:1409.1556, 2014.

[19] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke y Andrew Rabinovich: *Going Deeper with Convolutions*. En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, páginas 1–9, 2015.

[20] Zeiler, Matthew y Rob Fergus: *Visualizing and Understanding Convolutional Networks*. En *European Conference on Computer Vision*, páginas 818–833. Springer, 2014.

Índice alfabético

Visión por Computadora: Movimiento, Estructura y Detección

Joaquín Salas
jsalasr@ipn.mx

Marzo de 2016