Alma Mater Studiorum · Università di Bologna

SCUOLA DI INGEGNERIA E ARCHITETTURA

Corso di Laurea in Ingegneria Informatica

Determinazione della confidenza di mappe depth tramite Deep Learning

Relatore:

Chiar.mo Prof.

Mattoccia Stefano

Correlatori:
Matteo Poggi
Fabio Tosi

Presentata da:
Fusco Alessandro

Sessione II
2016/2017

Indice

1 Introduzione **2**

2 Sistemi di visione stereo **4**

2.1 Cenni di stereo visione 4

2.2 Sistemi attivi e passivi 4

2.3 Algoritmi di matching stereo 5

2.4 Misure di confidenza per algoritmi stereo 6

3 Il progetto **7**

3.1 Dataset 7

3.2 Data preprocessing pipeline 8

3.3 La rete neurale 9

3.3.1 Strumenti utilizzati 9

3.3.2 Upsampling 9

3.3.3 Funzioni di attivazione 10

3.3.4 Funzione Loss 11

3.3.5 Architettura end-to-end 12

3.3.6 Training 12

3.4 Risultati 12

3.5 Conclusioni 12

Capitolo 1

Introduzione

Nel vasto panorama del digitale, uno degli argomenti più caldi degli anni recenti è l'intelligenza artificiale (AI), che sta ricevendo sempre più attenzione sia dal mondo accademico che da quello industriale.

In particolare il Machine Learning (apprendimento automatico), ovvero quella branca dell'intelligenza artificiale che si occupa di fornire ai calcolatori l'abilità di apprendere senza essere stati esplicitamente programmati, ha subito recentemente un boom esponenziale.

Seppur le basi matematiche del Machine Learning siano state definite e studiate lungo il corso del ventesimo secolo, solo di recente [1] le si sono potute applicare efficientemente a scopi pratici, grazie al notevole miglioramento delle tecnologie di calcolo parallelo necessarie.

Tra le tante applicazioni di Machine Learning, emerge tra le altre quella relativa alla computer vision. Nel momento in cui si vede necessaria un'interpretazione ad alto livello di quelli che sono i vari bit di informazione di un'immagine o di un video, quando è necessario definire, rilevare o ricostruire proprietà visuali d'interesse inesprimibili proceduralmente, lì i mondi della visione artificiale e dell'apprendimento automatico si vanno a fondere.

La visione stereo è quella branca della computer vision che si occupa della ricostruzione di scene tridimensionali tramite l'acquisizione di immagini bidimensionali, tentando di ricostruire l'informazione perduta nel processo di acquisizione.

L'esigenza di accuratezza nell'impiego di algoritmi stereo (detti di stereo matching) ha portato necessità della creazione di misure di confidenza, ovvero indici che riescano a valutare a monte la correttezza di un processo di ricostruzione 3D arbitrario.

I metodi di misura di misura di confidenza attuali sono molteplici; alcune strategie possono utilizzare nozioni geometriche del sistema stereoscopico; metodi più complessi, quali alberi di classificazione, possono combinare le varie strategie per valutare quale sia la più efficace in un determinato contesto. Tecniche più recenti basate su Deep Learning, anche se computazionalmente più costose, hanno dimostrato però di portare a risultati di qualità notevolmente superiore, in quanto riescono a riconoscere pattern ad alto livello dei possibili problemi che possono indurre un algoritmo stereo a riportare risultati errati, quale la presenza di superfici riflettenti o occlusioni.

Questa tesi vuole essere uno studio su come applicare tecniche di deep learning al fine della determinazione di una misura di confidenza "full-resolution", ovvero calcolata analizzando l'intera immagine stereo.

Capitolo 2

Sistemi di visione stereo

2.1 Cenni di stereo visione

Due degli scopi fondamentali della percezione spazio-visiva umana, sono la determinazione della posizione degli oggetti nell'ambiente, e la distinzione di essi.

La stereopsi, o visione stereoscopica, è ciò che ci permette di percepire la realtà tridimensionalmente, e quindi di assegnare una profondità ai punti dello spazio. Essa risulta dalla fusione delle due immagini leggermente differenti proiettate all'interno dei nostri occhi; un oggetto viene percepito come vicino se, nel confronto tra i segnali generati dalle due retine, il nostro cervello lo vede scostato. Al contrario, minore è lo scostamento, maggiore è la distanza percepita.

La replica di questo interessante meccanismo vede numerosissime applicazioni a livello ingegneristico, dalla guida autonoma [2] [3] alla realtà aumentata [4]; perciò la visione stereo è uno dei principali argomenti di studio della visione artificiale.

2.2 Sistemi attivi e passivi

Le metodologie utilizzate per raccogliere dati riguardo alla configurazione geometrica di una scena tridimensionale si suddividono in due categorie. Quelle di tipo *attivo* usufruiscono di dispositivi che interagiscono fisicamente con l'ambiente circostante, ad esempio tramite laser o ultrasuoni, acquisendo ed elaborando i dati tramite appositi rilevatori.

Queste tecniche però risultano essere troppo costose e invasive; in questa tesi si discuterà perciò delle metodologie *passive*, che sfruttano hardware molto più economico e di dimensione ridotta e utilizzano teoria della visione stereoscopica per ricostruire l'informazione tridimensionale a partire da due immagini catturate contemporaneamente in due punti discostati l'uno dall'altro di una distanza nota, detta *baseline*. Queste due immagini saranno poi elaborate da un *algoritmo di matching*, il cui compito è riottenere le informazioni sulla profondità andate perdute in fase di acquisizione.

2.3 Algoritmi di matching stereo

Il compito degli algoritmi di matching stereo è quello di ricavare l'informazione 3D a partire da una coppia di immagini stereo. Dopo aver applicato alle immagini una distorsione geometrica, detta rettificazione, che permette di limitare geometricamente lo spazio di ricerca su una dimensione sola, l'obiettivo è quello di trovare una corrispondenza per ogni pixel dell'immagine sinistra con ciascun pixel dell'immagine di destra.

Una volta trovata la corrispondenza, dalla differenza di posizionamento di due pixel corrispondenti viene creata una mappa di disparità, che contiene le informazioni sulla profondità di ciascun punto.

Le immagini, però, possono presentare delle caratteristiche che, per alcuni punti, rendono la ricerca di corrispondenza molto complessa, se non impossibile. In particolare si citano:

Rumore e distorsioni fotometriche potrebbero causare variazioni casuali di luminosità o di colore nell'immagine, rendendo difficile il matching.

Superfici non-lambertiane , ovvero che non diffondono la luce in modo uniforme lungo tutte le direzioni, quali specchi, vetri, potrebbero apparire in modo molto differente tra le due immagini, specialmente se il rapporto tra la distanza dell'oggetto e la baseline è basso.

Texture uniformi e pattern periodici rendono il lavoro particolarmente difficile perchè estendono molto lo spazio di ricerca del matching.

Occlusioni provocate da oggetti vicini, che nascondono a una delle due telecamere aree viste dall'altra. In questo caso la corrispondenza non è ammessa.

2.4 Misure di confidenza per algoritmi stereo

Sebbene in letteratura siano stati proposti numerosi approcci per risolvere il problema della corrispondenza stereo, e benchè gli algoritmi allo stato dell'arte consentano di ottenere risultati piuttosto accurati, i problemi intrinseci citati in precedenza possono risultare ad assegnamenti di disparità errati.

É sorta dunque la necessità di introdurre criteri e misure di confidenza, ovvero metodologie di predizione degli errori e misure associate all'incertezza dei risultati.

Come descritto in dettaglio in «Misure di confidenza e algoritmi per il refinement di mappe depth» [5], una misura di confidenza ideale associa alle mappe di disparità valori con le seguenti proprietà:

- Essere alti per le disparità corrette e bassi per gli errori. Se i pixel corrispondenti fossero classificati in ordine decrescente di confidenza, allora tutti gli errori dovrebbero essere classificati per ultimi. La classificazione dovrebbe essere corretta anche per pixel di interesse particolare, come quelli presso le discontinuità.
- Essere in grado di individuare pixel occlusi.
- Essere utili per selezionare le disparità corrette tra le ipotesi generate da diverse strategie di matching.

Questo progetto punta a ricavare una misura di confidenza tramite una rete neurale convoluzionale, che, a differenza di altre ricerche quali [6], vuole operare sulle immagini a full-resolution invece che su patch ridotte.

Capitolo 3

Il progetto

Nel seguente capitolo verrano descritti i metodi e i modelli con cui è stata progettata e implementata la misura di confidenza.

3.1 Dataset

Il dataset KITTI [7] è composto da una collezione di 194 coppie di immagini stereo, già rettificate, rappresentanti scene di guida e acquisite in molteplici condizioni metereologiche, durante il giorno. Le immagini sono state acquisite per mezzo di una coppia di telecamere montate, a circa 54 centimetri l'una dall'altra, sopra il tettuccio di un'auto.

L'immagine groundtruth è una mappa di disparità esatta, ovvero affidabile, in quanto è stata acquisita tramite la metodologia attiva del laser a scansione, che assegna un valore a circa il 30% dei pixel nell'immagine. Nel dataset KITTI, ai pixel per cui non è disponibile l'informazione di disparità, è assegnato il valore 0.

Come detto in precedenza, lo scopo della rete neurale proposta è quello di generare una mappa di confidenza elaborando l'immagine sinistra della coppia stereo e la corrispettiva mappa di disparità generata da un algoritmo di matching. È stato dunque utilizzato l'algoritmo Ad-Census [8] per generare le mappe di disparità di cui verrà imparata la confidenza.

È necessario precisare che, poichè il training è stato effettuato esclusivamente con questo dataset e solo con questo algoritmo di disparità, è probabile che la rete riporti

risultati poco precisi per scene in contesti differenti da quello stradale di KITTI, o per algoritmi diversi. Non è esclusa però l'ipotesi che la rete possa portare a risultati più stabili con un adeguato tuning su dataset più vari.

3.2 Data preprocessing pipeline

Prima di poter essere dati in pasto dalla rete neurale, i dati devono essere preprocessati. Il preprocessing è suddiviso in due fasi differenti:

- Il **Preprocessing offline** viene effettuato prima della fase di training. La pipeline, implementata in python tramite la libreria *Luigi* [9], si occupa in un primo momento di caricare le immagini del dataset e di adattarli tutti alla codifica a 16 bit.

  Vengono poi create le mappe di confidenza groundtruth, confrontando le mappe di disparità Ad-Census e le mappe di disparità groundtruth di KITTI, catalogando come corretti i pixel la cui disparità differisce dalla groundtruth per un valore minore della soglia arbitraria di 3 e come errati altrimenti.

  Oltre ai valori 1 e 0 come valori rispettivamente per pixel giusti e sbagliati, vengono etichettati con -1 i valori per cui non è disponibile un valore di groundtruth, a causa della bassa densità delle mappe groundtruth di kitti.

  Le triple di immagini composte da immagine left, mappa di disparità ad-census e la mappa di confidenza groundtruth generata, sono divise in tre set, uno per il training, uno per la validazione e uno per il testing.

  Sono dunque compattate e salvate in tre files binari, di formato *tfrecords*; uno per ciascun set.

- Il **Preprocessing online** è invece eseguito contemporaneamente alla fase di training. Per non rendere la lettura da disco il collo di bottiglia della computazione, 16 thread paralleli si occupano costantemente di caricare in memoria le immagini che verranno poi elaborate dalla *GPU*.

  Come frequentemente visto in letteratura, l'immagine left viene normalizzata per accelerare la discesa del gradiente.

Per aumentare virtualmente la dimensione del dataset, e per risolvere il problema delle grandezze differenti delle immagini, ciascuna tripla di immagini viene ritagliata in patch di dimensione 256x512 centrate casualmente.

3.3 La rete neurale

3.3.1 Strumenti utilizzati

La rete neurale viene progettata tramite la libreria di computazione Tensorflow [10], ideata da Google. Il concetto fondamentale su cui si basa Tensorflow è che tutto il processo di computazione viene definito tramite un grafo i cui nodi sono operazioni e i cui rami sono tensori, ovvero matrici multidimensionali. Tramite le API python viene costruito il grafo che poi verrà trasferito, per l'esecuzione, al backend scritto in C++. E' importante sottolineare che l'esecuzione di una rete neurale è molto pesante dal punto di vista computazionale, e Tensorflow permette di utilizzare efficientemente l'alto grado di parallelismo fornito dalle *GPU* tramite un'integrazione trasparente.

3.3.2 Upsampling

Poichè l'immagine di output ha la stessa risoluzione delle immagini di input, viene utilizzata un'architettura *encoder-decoder* convoluzionale, simile a quella introdotta in SegNet [11].

Reti di questo tipo sono composte da due sezioni principali: l'encoder mira a estrarre le features più importanti tramite sequenze di convoluzioni e pooling. Le convoluzioni aumentano la profondità della mappa di attivazione mentre il pooling ne riduce la risoluzione spaziale. Intuitivamente, questa sequenza di operazioni può essere interpretata come una "codifica", ovvero si tenta di trasformare l'informazione spaziale data dai pixel delle immagini in matrici profonde che ne riassumono le features ad alto livello imparate dai vari kernel.

Il decoder invece effettua l'operazione inversa; dalla codifica di features calcolata dall'encoder, tenta di riottenere l'informazione spaziale andata persa nel pooling. Questa operazione, detta *upsampling*, è un tema particolarmente studiato in letteratura, spe-

cialmente per quanto riguarda la segmentazione, ovvero la classificazione semantica dei singoli pixel in un'immagine.

Le principali metodologie per affrontare questo problema di ricostruzione sono 3:

- **Interpolazione bilineare**: viene eseguito l'upscaling di una mappa di attivazione tramite filtri bilineari, che si limitano ad aumentare la risoluzione, come avviene comunemente con le immagini normali. I problemi derivati da questo approccio sono l'alto costo di inizializzazione dei filtri e l'impossibilità intrinseca di poter cambiare (e dunque imparare) i pesi dei filtri tramite back-propagation.

- **Fully Convolutional Layer**: Noh, Hong e Han in [12] optano per un'interessante approccio che permette loro di riutilizzare reti di classificazione, rimuovendo il livello finale fully-connected e sostituendolo con un livello di convoluzione trasposta, a cui è delegato il compito della classificazione pixel-per-pixel.

- **Unpooling**: L'approccio utilizzato in SegNet prevede il salvataggio degli indici nel livello di max-pooling che vengono poi utilizzati per riassegnare spazialmente le features al pixel corretto. Ciò comporta una dipendenza del livello di unpooling al corrispondente livello di pooling. Ne risulta dunque un'architettura dove l'encoder è simmetrico al decoder.

L'approccio utilizzato in questa tesi è una combinazione tra quello utilizzato in SegNet e quello utilizzato dalle reti Fully Convolutional; viene mantenuto un decoder simmetrico all'encoder, ma che utilizza la trasposta della convoluzione come strumento per l'upsampling. Per migliorare la precisione della ricostruzione spaziale, subito dopo il livello di trasposizione, viene sommata l'attivazione del livello dell'encoder corrispondente.

3.3.3 Funzioni di attivazione

La peculiarità delle reti neurali è l'introduzione di una funzione di attivazione, ovvero una funzione differenziabile ma non lineare, che viene applicata in uscita a ciascun elemento delle mappe di attivazione. È proprio la non linearità di queste funzioni che permette alle reti neurali di inferire distribuzioni complesse.

Come è uso frequente in letteratura, viene utilizzata la funzione detta *Rectifier Linear Unit*, o ReLU, a seguito di ciascun livello di convoluzione. E' definita come segue:

$$ReLU(x) = \begin{cases} x & \text{se } x > 0 \\ 0 & \text{altrimenti} \end{cases}$$

Intuitivamente, lo scopo della Relu è quello di bloccare la retropropagazione del gradiente per attivazioni negative, ovvero non utili alla classificazione dei pixel.

Nel livello finale, invece, viene utilizzata la funzione sigmoide, definita come:

$$\text{sigmoid} : R \to [0; 1], \quad x \to \frac{1}{1 + e^{-x}}$$

La funzione mappa i valori reali a un valore compreso tra zero e uno, ed è utile per rappresentare la probabilità che il corrispondente pixel della mappa rappresenti un valore accurato (1) o no (0).

3.3.4 Funzione Loss

La funzione di loss è la funzione che rappresenta l'errore di costruzione della mappa di confidenza. In fase di training, il compito della rete neurale è quello di minimizzare la funzione di loss tramite la discesa del gradiente.

La loss è calcolata confrontando la mappa di output della rete con la mappa ground-truth ottenuta in fase di preprocessing. La formula utilizzata per la loss è quella di cross entropy, che stima con adeguata precisione la somiglianza tra due tensori.

La funzione di cross entropy è definita come:

$$\mathcal{L}(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

Dove $\hat{y} \in]0, 1[$ è la confidenza predetta di un singolo pixel e $y \in \{0, 1\}$ è il valore groundtruth relativo a quel pixel.

Intuitivamente, i termini y e $(1-y)$ sono utilizzati come selettore, mentre il logaritmo va a penalizzare valori vicino allo 0. A un valore incerto, ovvero di probabilità calcolata 0.5, verrà assegnato un valore di loss di $-\log(0.5) \simeq 0.6931$

La loss per un'immagine è calcolata come la media aritmetica della loss per tutti i pixel per cui è definito un valore groundtruth.

3.3.5 Architettura end-to-end

Ricapitolando, l'architettura proposta è composta da una rete encoder e una rete decoder, seguita da un livello di regressione finale che si occupa di inferire le probabilità di ciascun pixel della mappa.

La rete di encoding è composta da 4 livelli di convoluzione. Ogni livello di encoding esegue la convoluzione con un set di filtri 3x3 per produrre una serie di feature maps. Queste mappe vengono poi normalizzate [13]. Quindi, una funzione ReLU è applicata a ciascun elemento. A seguire, viene applicato un sub-sampling di un fattore 2, tramite max-pooling.

Nonostante il subsampling possa aiutare a conferire alla rete invarianza alle traslazioni e in generale maggiore robustezza, provoca inevitabilmente una perdita di risoluzione spaziale. Ciò può essere deleterio in operazioni dove la delimitazione della classificazione dei vari pixel è vitale. Dunque, è necessario salvare l'informazione di attivazione precedente all'operazione di pooling, al fine di poter farne un feedforward al corrispondente livello di upsampling nel decoder.

La rete di decoding è composta da 4 livelli di convoluzione trasposta, anche detta convoluzione con stride frazionali. Anche in questo caso vengono utilizzati kernel 3x3, e la trasposta è applicata con stride 2 per compensare il fattore di riduzione 2 eseguito dal subsampling.

Al livello finale è applicata la funzione di attivazione sigmoide, che si occupa della classificazione binaria di ciascun pixel.

3.3.6 Training

3.4 Risultati

3.5 Conclusioni

Bibliografia

[1] Alex Krizhevsky, Ilya Sutskever e Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks*. 2012, pp. 1097–1105.

[2] Steven B Goldberg, Mark W Maimone e Larry Matthies. «Stereo vision and rover navigation software for planetary exploration». In: *Aerospace Conference Proceedings, 2002. IEEE*. Vol. 5. IEEE. 2002, pp. 5–5.

[3] Alberto Broggi, Massimo Bertozzi e Alessandra Fascioli. «Self-calibration of a stereo vision system for automotive applications». In: *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*. Vol. 4. IEEE. 2001, pp. 3698–3703.

[4] Masayuki Kanbara et al. «A stereo vision-based augmented reality system with an inertial sensor». In: *Augmented Reality, 2000.(ISAR 2000). Proceedings. IEEE and ACM International Symposium on*. IEEE. 2000, pp. 97–100.

[5] Fabio Tosi. «Misure di confidenza e algoritmi per il refinement di mappe depth». In:

[6] Matteo Poggi e Stefano Mattoccia. «Learning from scratch a confidence measure». In: *Proceedings of the British Machine Vision Conference (BMVC)*. A cura di Edwin R. Hancock Richard C. Wilson e William A. P. Smith. BMVA Press, 2016, pp. 46.1–46.13. ISBN: 1-901725-59-6. DOI: 10.5244/C.30.46. URL: https://dx.doi.org/10.5244/C.30.46.

[7] Andreas Geiger, Philip Lenz e Raquel Urtasun. «Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite». In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[8] Ramin Zabih e John Woodfill. ≪Non-parametric local transforms for computing visual correspondence≫. In: *European conference on computer vision*. Springer. 1994, pp. 151–158.

[9] Spotify. *Luigi, a tool to build complex pipelines of batch jobs.* `https://github.com/spotify/luigi`. 2017.

[10] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[11] Vijay Badrinarayanan, Ankur Handa e Roberto Cipolla. ≪SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling≫. In: *arXiv preprint arXiv:1505.07293* (2015).

[12] Hyeonwoo Noh, Seunghoon Hong e Bohyung Han. ≪Learning Deconvolution Network for Semantic Segmentation≫. In: *CoRR* abs/1505.04366 (2015). URL: `http://arxiv.org/abs/1505.04366`.

[13] Evan Shelhamer, Jonathan Long e Trevor Darrell. ≪Fully Convolutional Networks for Semantic Segmentation≫. In: *CoRR* abs/1605.06211 (2016). URL: `http://arxiv.org/abs/1605.06211`.