Vediamo/Let's See

Presentazione

Vediamo/Let's See è un sistema di visione artificiale **open source** realizzato in *JavaScript* nell'intento di favorire la diffusione di questo tipo di competenze presso il pubblico.

1 – Scopo dell'applicazione

Per molti animali gli occhi sono i principali organi di senso; la vista è il senso prevalente perché non solo consente di caratterizzare l'ambiente circostante velocemente e con grande precisione ma permette di farlo mantenendosi a distanza di sicurezza, quindi evitando il rischio di mettere in allerta una potenziale preda o di farsi scoprire da un predatore.

Considerazioni analoghe valgono anche per i sistemi artificiali che potenzialmente potrebbero trovarsi ad affrontare problemi analoghi: dagli organismi cibernetici come i robot a sei zampe progettati per l'esplorazione marziana, agli arti meccanici impiegati per verniciare le portiere delle auto in catena di montaggio, per finire con applicazioni come il controllo degli accessi ad una stanza o la lettura di testi dattiloscritti.

Data l'importanza e l'indubbio fascino di questa tipologia di applicazioni sembrerebbe logico attendersi un vivace interesse del pubblico per apprendere e sperimentare questa tecnologia, tanto più che i materiali richiesti per intraprendere degli esperimenti sono alla portata di tutti (ognuno di noi porta costantemente con sé almeno un paio di *webcam* integrate nel telefono cellulare); la realtà però è ben diversa poiché ad un quarto di secolo di distanza dalla formalizzazione delle prime API per l'interfacciamento delle telecamere il novero degli programmatori professionisti per non parlare degli appassionati che si cimentano con questo tipo di strumenti continua ad essere relativamente esiguo.

Il ritardo nella diffusione delle competenze inerenti la visione artificiale viene spesso giustificato osservando che la visione è difficile poiché non solo implicano il possesso di vaste conoscenze concernenti argomenti disparati ma comportano la necessità di affrontare la complessità dei problemi cognitivi. Come dire: *andate a prendere un paio di lauree e poi ne parliamo!*

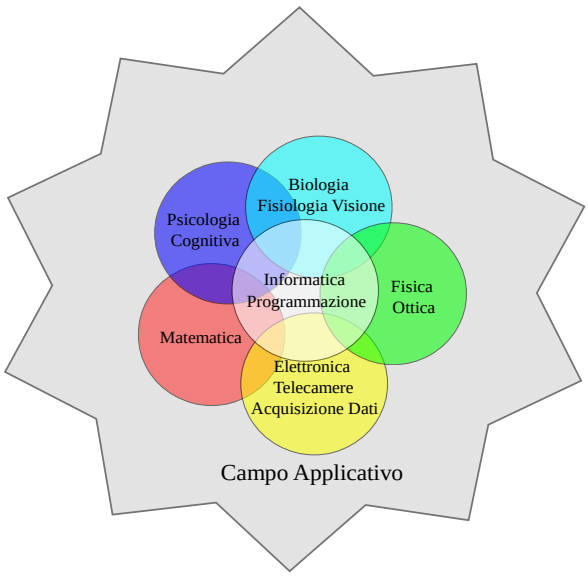

Figura-1) Le competenze della Visione Artificiale

2 – La visione artificiale NON è difficile

Detto ciò la prassi vorrebbe che mi dilungassi spiegando perché la visione artificiale è difficile ma io non lo farò; al contrario cercherò di confutare la precedente affermazione mostrando che se si imposta correttamente il discorso è possibile "domare" la complessità dell'argomento in modo da renderlo accessibile anche ad un pubblico di non specialisti.

Per ottenere questo risultato è necessario in primo luogo comprendere la natura delle difficoltà che per lungo tempo hanno fatto della visione un argomento elitario.

I problemi sono sostanzialmente tre: il costo proibitivo dei sistemi di visione professionali, le competenze richieste per usarli e la complessità intrinseca dei processi cognitivi considerati.

Come è facile intuire il problema più difficile è il terzo poiché la complessità e le esigue dimensioni delle cellule sensoriali e nervose rendono difficile ricostruire l'organizzazione dei circuiti biologici; viceversa la mente umana operando ad un livello superiore non ha coscienza del lavoro necessario per distillare l'informazione rilevata dagli organi di senso al fine di renderla fruibile per il ragionamento, per cui occorrerà probabilmente ancora molto tempo prima di riuscirci.

Preso atto di questo sconfortante risultato potremmo chiederci: ma ci serve davvero tanto oppure potrebbe già essere possibile ottenere dei risultati utili contentandoci di molto meno?

A mio avviso e a detta di molti la risposta corretta è la seconda. Se aspettiamo di avere un risultato perfetto non faremo mai nulla, viceversa sembrerebbe lecito presumere la possibilità di trarre dei benefici dallo sfruttamento delle capacità di visione delle macchine attuali.

Nel seguito assumeremo pertanto questa ipotesi di lavoro proponendoci di esplorare la possibilità di impiegare le telecamere che equipaggiano i normali PC o *smart-phone* per la nostra utilità personale. Quest'ultima considerazione fornisce incidentalmente una risposta al primo problema. Data l'ampia disponibilità di risorse sottoutilizzate (le *webcam*) possiamo escludere, almeno in un primo momento la necessità di acquistare strumenti professionali costosi, per cui non resta che da affrontare il tema delle competenze.

Tradizionalmente i testi introduttivi alla visione artificiale dedicano almeno un paio di capitoli ad argomenti di matematica avanzata, spiegando i concetti di convoluzione e di trasformata di Fourier, prima di addentrarsi nella spiegazione del funzionamento delle ottiche e dei dispositivi di ripresa. In atri termini si parte dal presupposto che lo specialista di visione possegga nozioni di matematica, fisica ottica, elettronica e informatica a livello universitario (in aggiunta alle conoscenze di fisiologia della visione e psicologia cognitiva di cui si è già parlato).

Tali requisiti avevano senso ed erano giustificati per non dire indispensabili in un'epoca in cui la realizzazione di esperimenti di visione artificiale richiedeva molto tempo e la spesa di ingenti somme di denaro, per cui non ci si poteva permettere di sbagliare, ma oggi?

La proposta costituita dal sistema **Vediamo/Let's See** si basa proprio su quest'ultima considerazione cioè sull'assunto che disponendo di risorse di visione abbondanti e a buon mercato ci si possa permettere di rovesciare l'impostazione tradizionale affrontando il tema in maniera esplorativa per tentativi ed errori.

Ma quali dovrebbero essere le caratteristiche di questo sistema di visione ideale?

In base alla mia esperienza utenti diversi hanno esigenze diverse e ciò che è buono per Tizio non va bene per Caio; allora perché non porre la gente in condizione di fare da sé configurando il sistema di visione o perlomeno la sua interfaccia utente come meglio credono.

Chiaramente se si lavora in C-Language o in un altro linguaggio di programmazione "duro" ciò non è possibile ma se ipoteticamente fosse possibile riorganizzare il processo di visione in modo da renderlo compatibile con un mezzo espressivo più vicino agli utenti finali il tutto potrebbe essere fattibile.

Dopo un rapido esame delle possibili alternative che mi venivano in mente: estensione delle funzioni di un interprete BASIC, sviluppo di una libreria per il linguaggio Scratch, Perl, eccetera ho deciso di puntare su JavaScript.

3 – Vediamo/Let's See un sistema di visione in JavaScript

Il punto di forza della proposta risiede nel connubio fra HTML e JavaScript e nella possibilità di utilizzare frammenti di codice JavaScript scritti da altri per introdurre “effetti speciali” e/o automatizzare le proprie pagine web.

Questa pratica è di una portata dirompente poiché consente praticamente a chiunque di realizzare dei *manufatti informatici*[1] complessi richiedendo come unica competenza informatica la capacità di usare un programma di scrittura e di effettuare il copia e incolla!

Le routine di visione sviluppate in JavaScript e rilasciate sotto forma di codice *open source* una volta rese disponibili saranno immediatamente fruibili da chiunque abbia la capacità di editare un testo.

A questo proposito alcuni scettici potrebbero obiettare che la soluzione proposta non offre neanche un milionesimo delle potenzialità di un sistema di visione professionale - non lo metto in dubbio - ma sul piano della divulgazione delle competenze concernenti la visione artificiale **Vediamo/Let's See** va probabilmente al di là di qualsiasi progetto precedentemente intrapreso.

4 – Condizioni di rilascio

Dal punto di vista tecnico **Vediamo/Let's See** consiste di una pagina di testo HTML-5 che fa riferimento a *tuisys.js* una libreria di visione scritta in JavaScript.

Sia il testo HTML che la libreria sono rilasciati sotto forma di codice aperto.

Nello specifico il codice di automazione web contenuto in *vediamo.html* e il codice della libreria *tuisys.js* viene rilasciato sotto le condizioni della Apache License Version 2.0 mentre la grafica e il testo del documento *vediamo.html* sono resi disponibili sotto le condizioni della Creative Commons License CC BY-SA.

Per i dettagli delle licenze si rimanda a:
- http://www.apache.org/licenses/LICENSE-2.0
- https://creativecommons.org/licenses/by-sa/2.5/it/.

1 Per *manufatto informatico* si intende una qualsiasi qualsiasi sequenza digitale indipendentemente dalla sua interpretazione come dati/informazioni piuttosto che codice programma (alcuni autori includono nella definizione anche la documentazione in formato cartaceo). Tale definizione è particolarmente calzante per le pagine HTML in cui il codice eseguibile e le informazioni sono inestricabilmente mischiate.

5 – Quick start

Poiché **Vediamo/Let's See** consiste di una pagina di testo HTML-5 per usufruirne sarà necessario aprire il documento con un browser HTML. In linea di principio si potrà utilizzare un qualsiasi browser a condizione che questo supporti HTML-5 e le estensioni multimediali di JavaScript note come getUserMedia()[2] [3].

Il funzionamento di **Vediamo/Let's See** è stato provato con successo con i seguenti browser[4]:
FireFox 17+ (Windows, Linux, Android),
Opera 12 (Windows) e Opera 18+[5] (Windows, Linux),
Chrome 21+ (Windows, Linux),
Midori (Windows).

Microsoft Internet Explorer e Apple Safari attualmente non supportano getUserMedia[6]; Microsoft Edge in base a quanto è stato dichiarato dovrebbe essere compatibile[7].

	Windows	Linux	Android	iOS
FireFox 17+	si	si	si	
Opera 12	si	si	Non testato	
Opera 18+	si	si	Non testato	
Chrome 21+	si	si	Non testato	
Midori	si	si	Non testato	
Internet Explorer	no			
Safari				no
Microsoft Edge	si			

Figura-2) schema riassuntivo

Impazienti di provare **Vediamo/Let's See**?
Cliccate sul link:

http://tuisys.com/it/index.html

e poi ancora su prova:

Figura-3) clicca su prova

2 https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia
3 https://w3c.github.io/mediacapture-main/getusermedia.html
4 NOTE: 1) In tutti i test è stato possibile acquisire le immagini ma nel caso dei browser più vecchi talvolta la sequenza di immagini acquisite in "Live" non era stabile. 2) lo standard lascia agli sviluppatori dei browser la facoltà di scegliere la risoluzione delle immagini acquisite che non si può modificare. La risoluzione
5 Il funzionamento delle chiamate getUserMedia(); in Opera 18+ differisce da ciò che accadeva in Opera 12.
6 Vedi "*Browser Compatibility*" in https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia
7 https://blogs.windows.com/msedgedev/2015/05/13/announcing-media-capture-functionality-in-microsoft-edge/

Il vostro browser caricherà il documento HTML che costituisce l'attuale interfaccia utente di **Vediamo/Let's See.**

Se tutto funziona correttamente si dovrebbe aprire una finestrella che vi inviterà a consentire l'utilizzo di una delle vostre *webcam* da parte del browser.

Figura-4) richiesta di abilitazione della videocamera in FireFox, Chromium, Opera.

Nel caso in cui il vostro computer o *smart-phone* siano dotati di più *webcam* scegliete il dispositivo che desiderate utilizzare e confermate il vostro consenso all'uso dello stesso.

Il programma dovrebbe cominciare immediatamente ad acquisire in “Live” visualizzando le immagini nella finestra o meglio nella *canvas*[8] corrispondente.

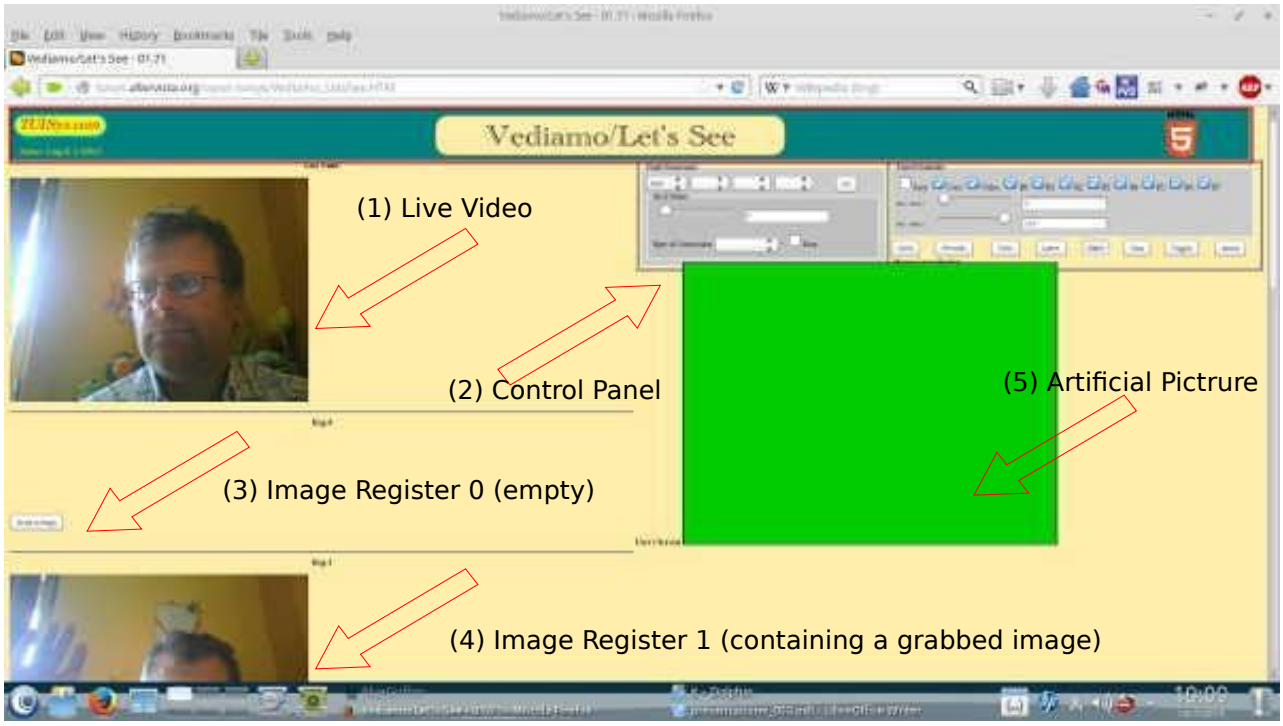

Figura-5) organizzazione dell'interfaccia utente di **Vediamo/Let's See** rel. 1.0

Nell'attuale versione il programma **Vediamo/Let's See** offre la possibilità di scegliere fra dieci *canvas* diverse di cui: una permanentemente allocata per la visualizzazione “Live” (*canvas* numero 1 nella figura 5), una permanentemente riservata per la visualizzazione di immagini artificiale cioè dipinte e otto “registri immagine” utilizzabili per memorizzare temporaneamente le immagini elaborate.

Nell'immagine di figura-5,

- l'ultima immagine acquisita dalla telecamera viene mostrata nella corrispondente *canvas* (1),

8 Nella definizione di HTML-5 vengono dette *canvas* (in italiano *tele*) ciascuna delle finestre di visualizzazione rettangolari utilizzabili per disegnare sullo schermo.

- il registro 0 è vuoto per cui la *canvas* ad esso abbinata è chiusa (3),
- la *canvas* 1 è invece aperta (4) tant'è che si intravede l'immagine in essa contenuta,
ed è aperta anche la *canvas* corrispondente all'immagine dipinta dal programma.

Per comodità degli utenti ad ognuno dei registi immagine è associato un bottone che consente di salvare l'immagine acquisita in “Live” nel corrispondente registro (3).

Continua ...