

Project Proposal

Prepared for: Davide Gariselli , Caratterizzazione della risposta spettrale del sensore RaspberryPi CAM
OmniVision OV5647

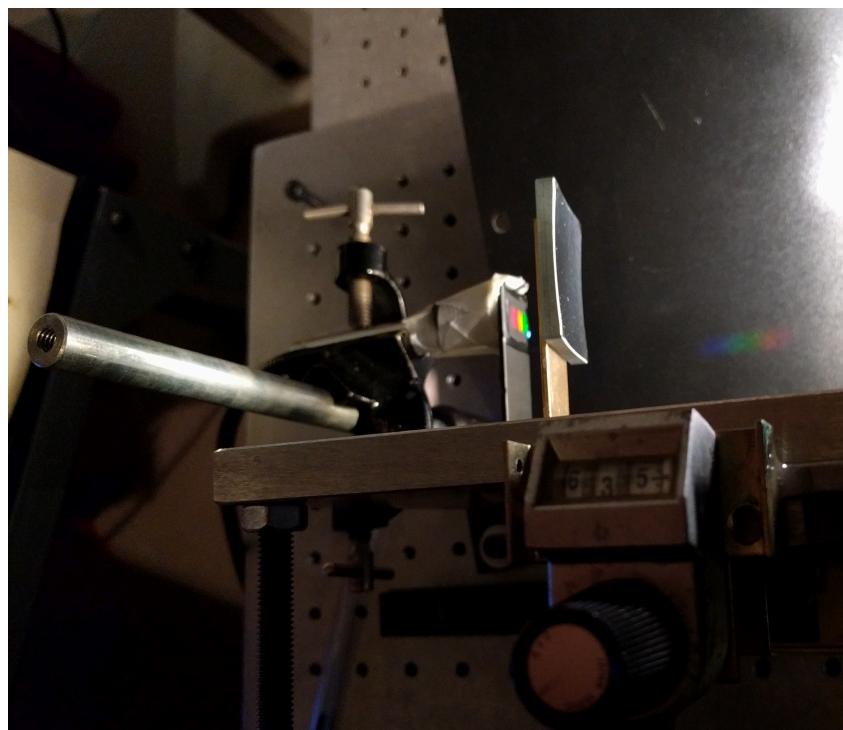
Prepared by: Davide Gariselli , laureando in Ingegneria Elettronica
20 February 2017

STUDY'S OBJECTIVE

L'obiettivo principale di questo studio è costruire uno sistema, il più possibile affidabile ed autonomo, per il calcolo della quantum efficiency/spectral response.

Nel caso specifico preso in esame, si utilizzerà una Raspberry Pi per l'acquisizione della "Raw Bayer Data" proveniente da una fotocamera OV5647, tramite uno script in python.

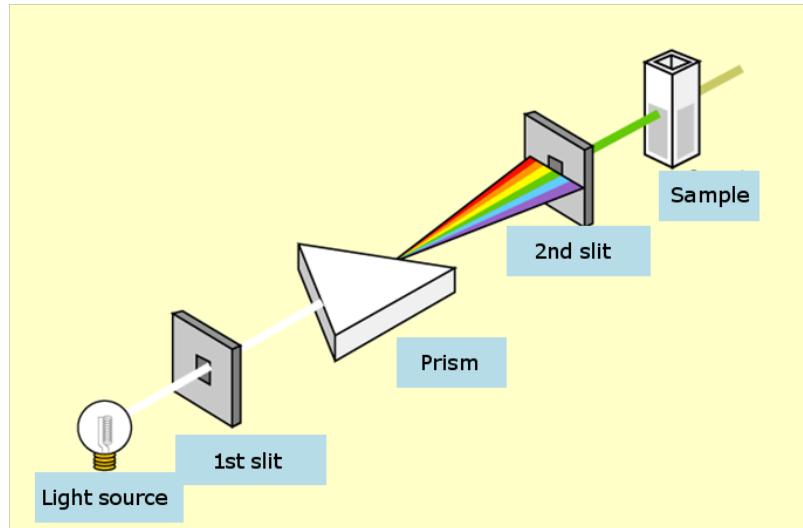
Il sistema di studio ed analisi dei dati, sviluppato in MatLab, non è vincolato a nessun tipo di fotocamera.



Indice degli argomenti:

- A. Caratterizzazione del monocromatore
 - Spettrometro e sorgente luminosa
- B. Cattura sensore Cmos
 - Cmos, bayer filter e storage dei risultati in Python script
- C. Interpolazione quantum efficiency
 - Versione semplificata
 - Versione completa con NDF

CARATTERIZZAZIONE MONOCROMATORE



Schema di principio del monocromatore

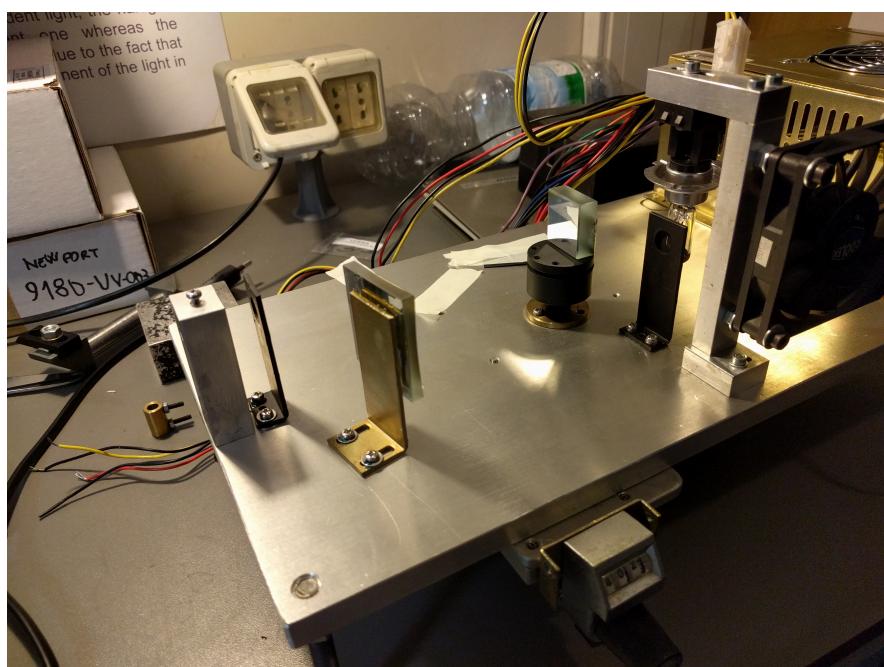
Objective

Misurare la risposta spettrale d'una sorgente luminosa, in funzione della lunghezza d'onda, all'uscita del monocromatore tramite uno spettrometro.

Goals

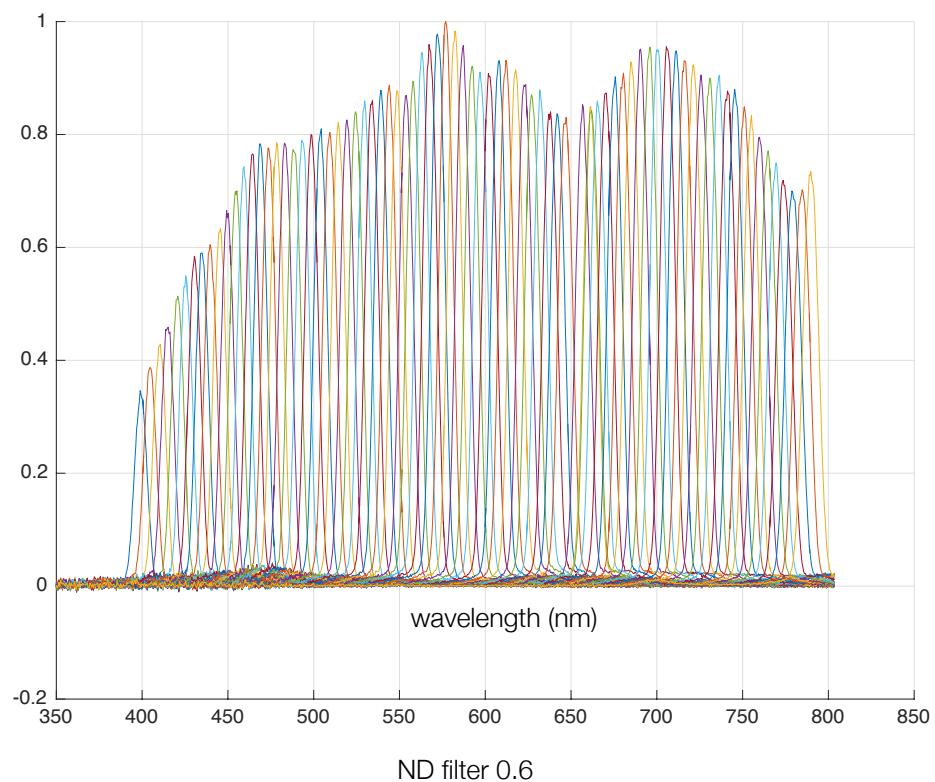
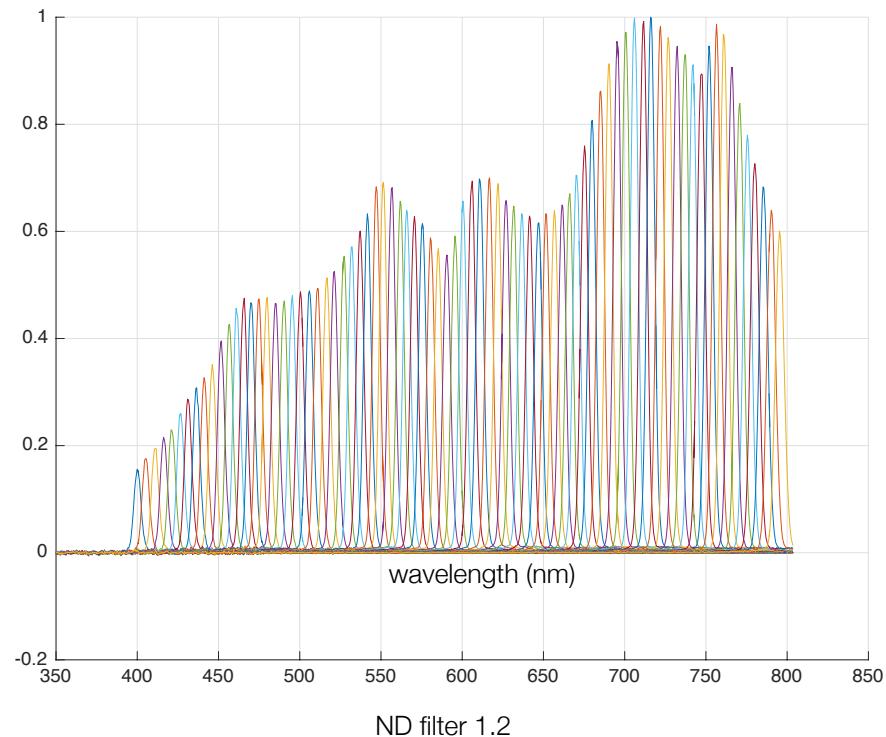
Allineare la manopola posta a fianco del monocromatore - selettore di lunghezza d'onda in nm - con la risposta creata dal software dello spettrometro.

Nel caso si voglia utilizzare un filtro a densità neutra, andrà posto tra la seconda feritoia del monocromatore ed il sensore dello spettrometro.

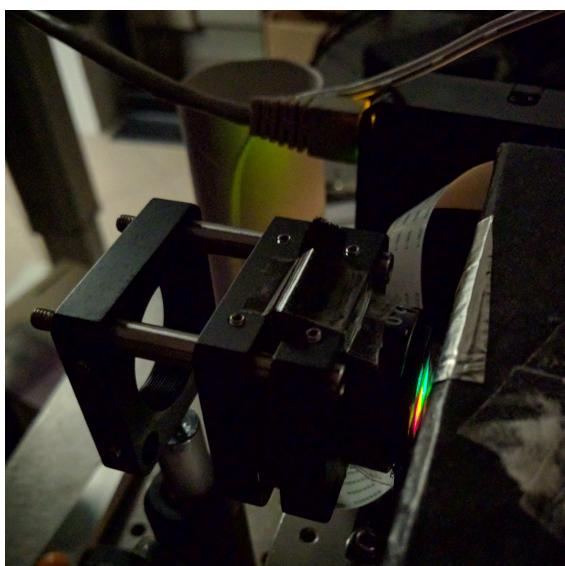


Results

Una volta allineata la manopola, lo spettrometro produrrà un file - .TXT - per ogni cattura, ogni 5 nm, dello spettro generato dalla sorgente luminosa.



CATTURA SENSORE CMOS RASPBERRY PI



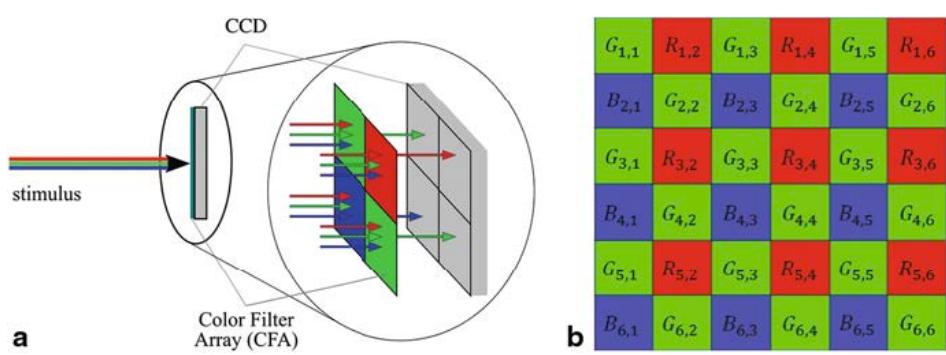
Objective

Estrarre la Raw Bayer Data "grezza" catturata dal sensore Cmos della fotocamera senza nessuna alterazione dovute alla Demosaicing.

Goals

Lo schema seguente illustra come un sensore Cmos è in grado di catturare la luce selezionata dal monocromatore, superata la seconda feritoia.

Uno script in python raccoglie i colori Red Green Blu in streaming dal sensore, incapsulandoli in un file .NPY (NumPy) denominato con la lunghezza d'onda catturata.



Funzionamento Cmos con filtro di Bayer.

La libreria utilizzata per catturare le immagini dalla raspberry è la "picamera", dopo diversi test è risultata la migliore per conservare inalterati i valori della Raw Bayer Data.

Results

Lo script in python crea una cartella con un nome a scelta, riempendola di file .NPY catturati a diverse lunghezze d'onda.

La cartella verrà successivamente compressa per agevolare il trasferimento.

La struttura NumPy si presta alla creazione d'un oggetto Array N-dimensionale, facilmente leggibile da MatLab.

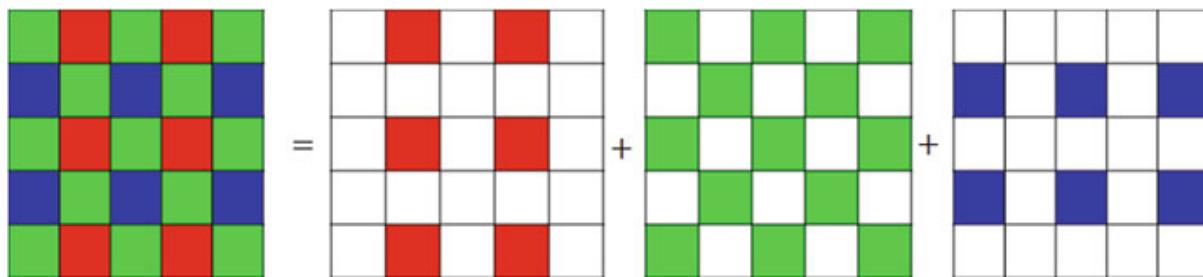
INTERPOLAZIONE QUANTUM EFFICIENCY

Objective

Sviluppo di un software in MatLab per calcolare la Quantum Efficiency, partendo dalla Raw Bayer Data - NPY - e dalla caratterizzazione del monocromatore.

Goals

Partendo dai dati di Raw Bayer, ricostruiamo l'immagine utilizzando la funzione **Demosaicing**.

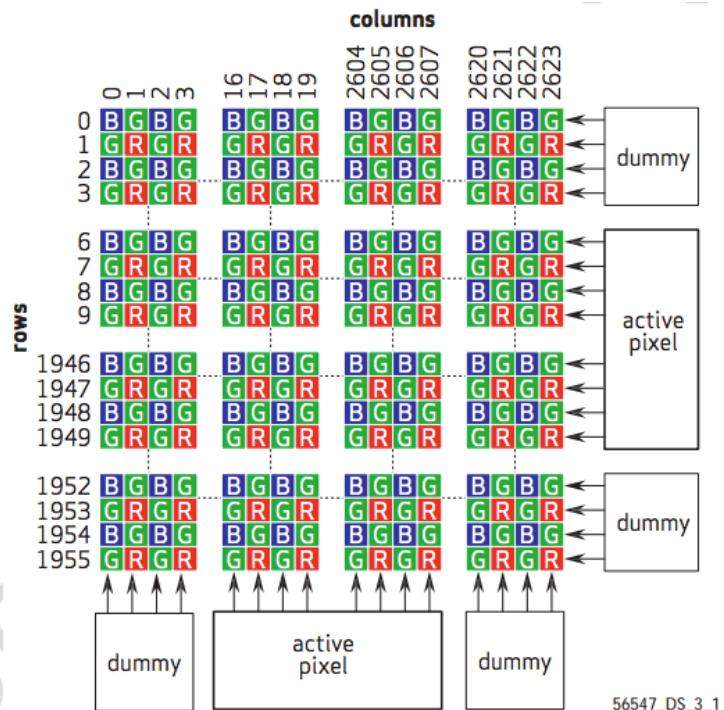


Operazione di Demosaicing, estrazione 3 colori RGB.

Nel fare questa operazione, dobbiamo estrarre i dati dall'incapsulamento NumPy.

La funzione Demosaicing ha bisogno di conoscere il Bayer pattern della fotocamera, nel nostro caso GRBG - che deve essere il medesimo ordine della cattura nel file NumPy - .

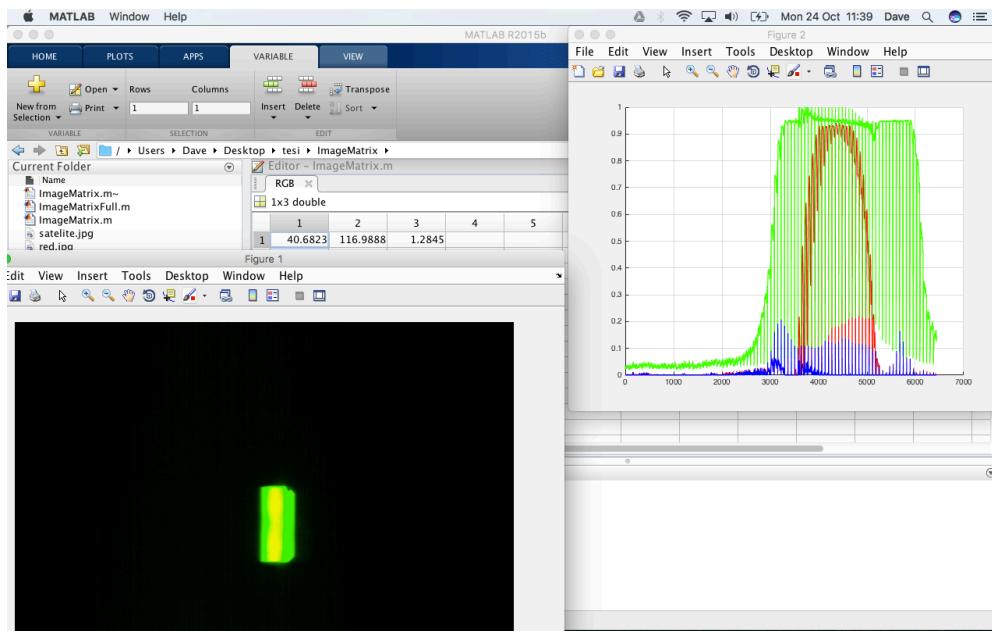
Alla fine di questo processo, si otterrà una matrice con i 3 diversi colori RGB - M-by-N-by-3 - di ogni una delle foto scattate della Rasp CAM.



56547_DS_3_1

Datasheet: <https://goo.gl/AHXpkq>

Il software procederà alla funzione di **Crap** dell'immagine. Più l'immagine verrà tagliata nel punto di massima luminosità, più accurato sarà il calcolo del Quantum Efficient.

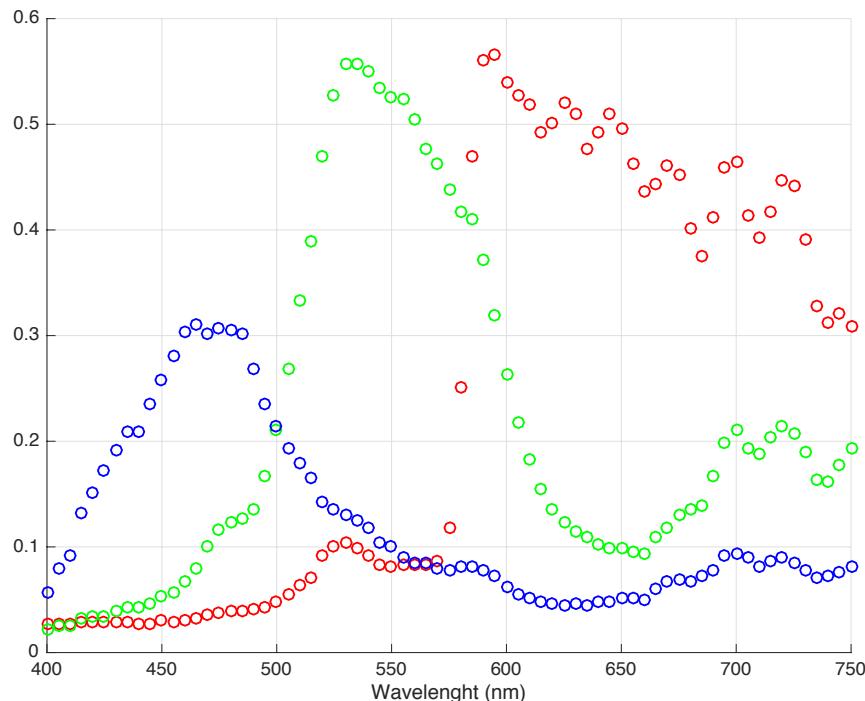


Software per la visualizzazione grafica: [Pixel Region analyzer](#)

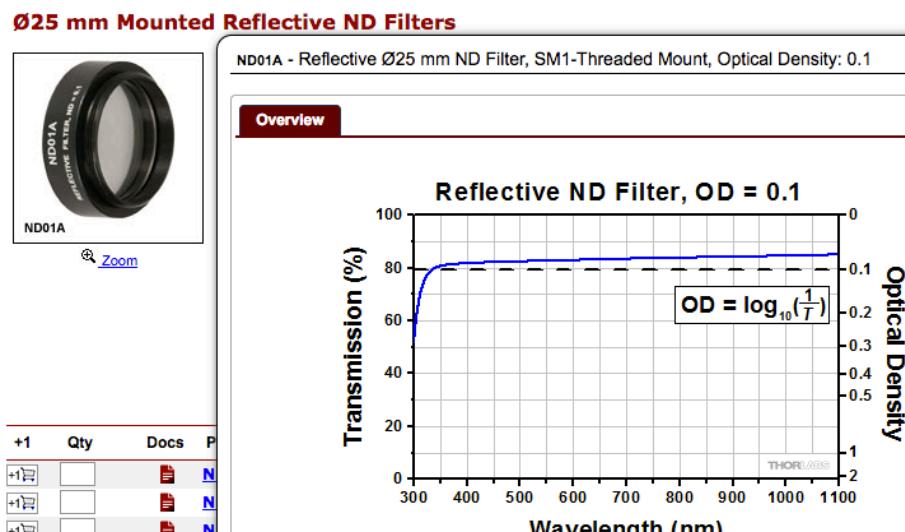
Nell'ultima stesura del software, non viene effettuato questo plot direttamente.

I risultati verranno normalizzati e mediati per ottenere un'unico valore RGB per ciascuna foto.

La prima versione del software [BayerRgb_to_RGB](#) conclude in questo punto, mostrando una poco accurata ma funzionale rappresentazione del Quantum Efficient.

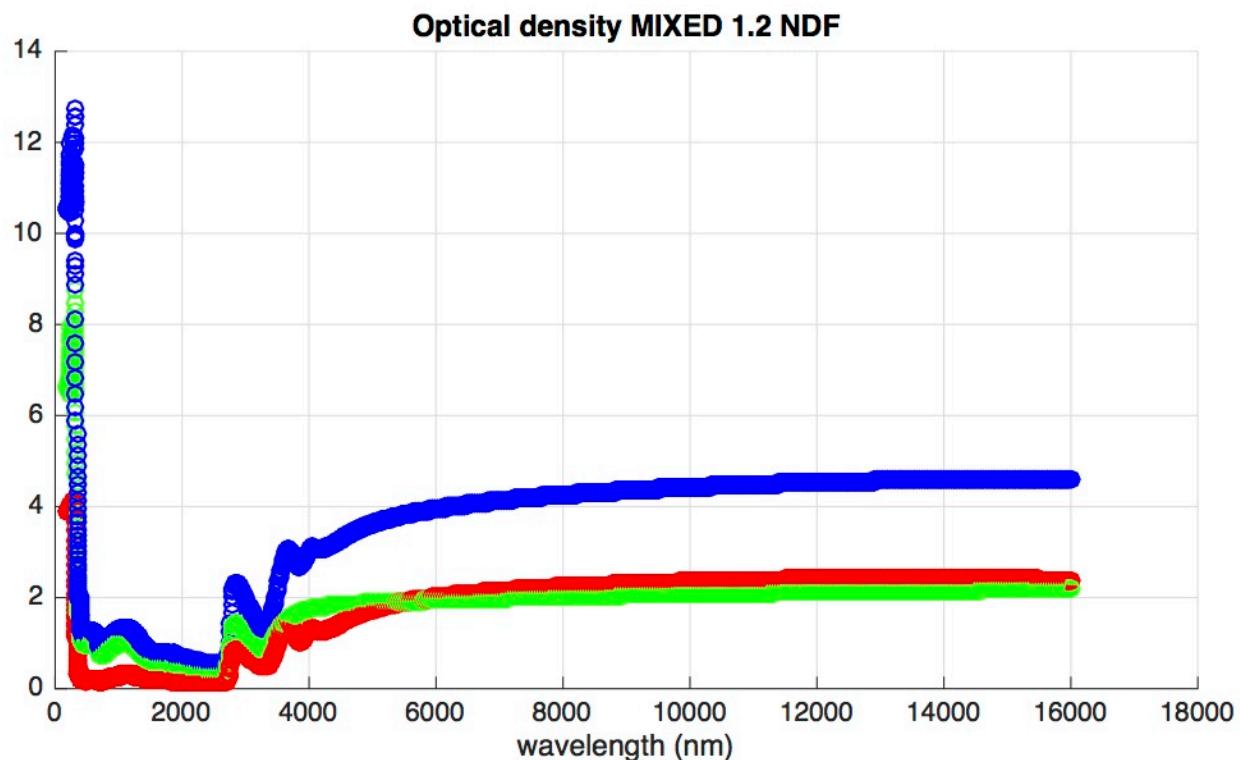


Per migliorare il calcolo del Quantum Efficient, seconda ed ultima versione del software BayerRgb to rgb+NDF, andrà ad inserire **l'interpolazione** con i risultati del **monocromatore**.



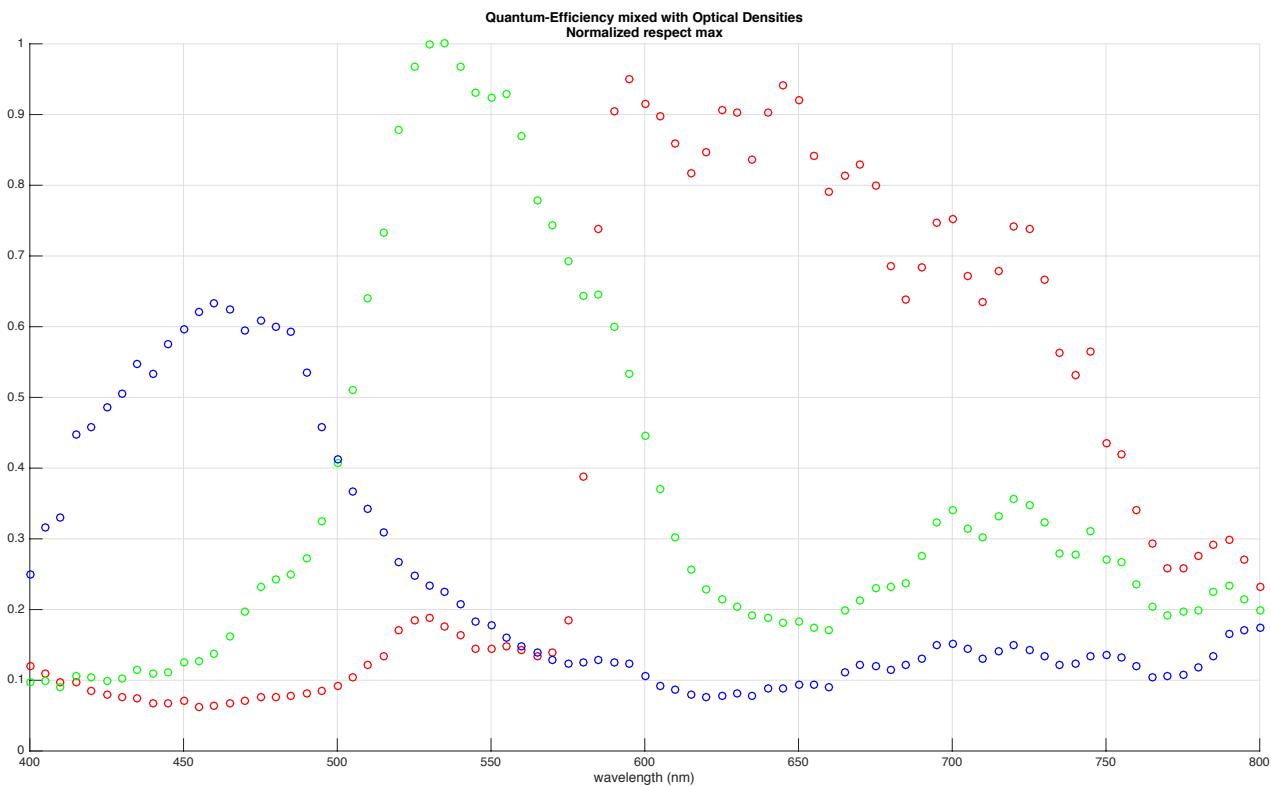
Esempio di Optical Density/Transmission(%) su NDF

Il monocromatore viene influenzato nella cattura dai Neutral-density filter - **NDF** -, applicato alla fotocamera.



Esempio di somma di due NDF, NDF somma in blu.

Il produttore del filtro fornirà l'intero grafico del relativo NDF utilizzato.



Il software è in grado di capire quanti NDF sono stati utilizzati, sommandoli automaticamente.

L'algoritmo presente nel software permetterà di **unire diverse catture**, fatte anche con più NDF, mostrando un'unico grafico risultante scalato e normalizzato.

Il grafico mostra il mix di due diverse catture, prima da 400-750nm con un NDF0.6 e seconda da 700-800nm con due NDF1.0 e NDF0.2 .

Results

Oltre alla produzione d'un grafico, mixed o no a seconda dei dati in input, che rappresenta il Quantum Efficient, il software, salva ogni risultato in un sistema a matrice.

Ogni piano sull'asse Z rappresenta una cattura - singolo file .NPY - restituito dallo script in python sulla Raspberry.

Di seguito verrà rappresentata la struttura, ripetitiva per ogni piano, della matrice 3D chiamata "mono" creata dal programma MatLab.

240	218	169	187	216	022	167	094	100	048	004	114	182	000	181	191	
136	236	011	129	205	032	066	095	068	193	012	086	043	046	076	194	
040	008	238	001	231	241	020	077	132	172	165	121	036	015	080	013	
112	083	106	128	154	027	223	217	145	184	18	250	002	242	090	023	
014	051	220	248	201	127	091	057	111	102	107	062	010	149	142	159	
108	227	029	079	139	252	044	174	138	155	175	105	126	190	073	092	
9	120	196	228	229	254	204	177	232	253	153	081	024	123	170	085	098
8	148	141	221	038	053	134	162	230	033	214	185	199	003	137	028	255
7	245	219	030	224	006	180	215	045	104	074	103	243	189	116	225	166
6	140	208	056	124	246	119	019	093	070	089	054	078	084	211	055	156
5	026	158	058	163	037	108	050	110	039	197	207	042	226	017	183	005
4	122	016	146	202	049	064	203	069	247	031	233	210	144	021	179	113
3	082	178	101	222	117	099	251	164	088	237	151	041	212	213	171	075
2	115	249	244	200	061	007	096	209	131	009	018	065	143	097	206	130
1	134	161	025	150	125	060	176	235	192	160	087	168	072	052	157	198
0	147	152	071	035	047	067	173	063	239	234	133	186	195	109	135	059

```

Variables - mono
mono x
5x81x3 double

val(:,:,1) =

Columns 1 through 11

0.1368    0.1252    0.1106    0.1105    0.0976    0.0907    0.0863    0.0845    0.0769    0.0769    0.0812
0.1104    0.1123    0.1029    0.1214    0.1191    0.1123    0.1171    0.1305    0.1254    0.1272    0.1432
0.2856    0.3613    0.3779    0.5116    0.5228    0.5552    0.5779    0.6267    0.6103    0.6572    0.6825
400.0000   405.0000   410.0000   415.0000   420.0000   425.0000   430.0000   435.0000   440.0000   445.0000   450.0000
0.6000    71.0000    0         0         0         0         0         0         0         0         0         0

Columns 12 through 22

0.0708    0.0731    0.0764    0.0803    0.0871    0.0868    0.0900    0.0924    0.0964    0.1056    0.1200
0.1444    0.1581    0.1853    0.2256    0.2658    0.2775    0.2848    0.3113    0.3716    0.4658    0.5847
0.7096    0.7234    0.7147    0.6811    0.6964    0.6852    0.6777    0.6127    0.5242    0.4723    0.4199
455.0000   460.0000   465.0000   470.0000   475.0000   480.0000   485.0000   490.0000   495.0000   500.0000   505.0000
0         0         0         0         0         0         0         0         0         0         0         0

Columns 23 through 33

0.1393    0.1541    0.1949    0.2113    0.2146    0.2020    0.1863    0.1651    0.1651    0.1685    0.1642
0.7312    0.8378    1.0055    1.1071    1.1434    1.1438    1.1069    1.0652    1.0569    1.0634    0.9956
0.3923    0.3537    0.3062    0.2839    0.2681    0.2567    0.2371    0.2085    0.2033    0.1832    0.1685
510.0000   515.0000   520.0000   525.0000   530.0000   535.0000   540.0000   545.0000   550.0000   555.0000   560.0000
0         0         0         0         0         0         0         0         0         0         0         0

```

La struttura in blu si ripete ad ogni colonna, cattura x cattura del monocromatore, mentre la riga rossa è fissa per ogni file .NPY

Red	Red	Red	Red	-->
Green	Green	Green	Green	-->
Blu	Blu	Blu	Blu	-->
Frequenza (nm)	-->			
NDF utilizzato	N° di catture			

L'ultimo piano della matrice "mono" è sempre il risultato mostrato nel grafico QE, riportato nella pagina sovrastante.