Università degli Studi di Padova

3D AUGMENTED REALITY

PROJECT REPORT - B.2

Casagrande Nicola, De Toni Alberto
... - 1236657

MASTER DEGREE IN ICT

02 · 2021

Local feature compression using autoencoders - SfM tests

Design a compression strategy for local SURF descriptors using autoencoders. Training data can be generated using the images of dataset Portello and Castle. Testing must be done on dataset FountainP-11 and Tiso (available at https://github.com/openMVG/SfM_quality_evaluation/tree/master/Benchmarking_Camera_Calibration_2008 and http://www.dei.unipd.it/~sim1mil/materiale/3Drecon/). Software must be implemented in MATLAB, Keras or Pytorch.

Testing on 3D reconstruction using SfM: The reconstructed descriptors (only for the test set) are used to perform a SfM reconstruction using COLMAP (using the two test dataset).

Programming languages: MATLAB/Python/C++.

1 Introduction

1.1 What are the descriptors

In computer vision, visual descriptors or image descriptors are descriptions of the visual features of the contents in images, videos, or algorithms or applications that produce such descriptions. They describe elementary characteristics such as the shape, the color, the texture or the motion, among others[1]. A feature detector (*extractor*) is an algorithm taking an image as input and outputting a set of regions ("local features" = "Interest Points" = "Keypoints" = "Feature Points"). A descriptor is computed on an image region defined by a detector. The descriptor is a representation of the image function (colour, shape, ...) in the region (typically an array). Two key operations are related to feature extraction:

- Feature detection: extract the features of interest
- Feature description: associate a descriptor to each feature in order to distinguish from the others

There are many pre-defined descriptors available to the users. Some of them are reported in fig. 1.

1.2 What is an autoencoder

What were our choices (programming language,...) cons of using matlab as a Machine Learning tool The objective of the project is

[1] https://en.wikipedia.org/wiki/Visual_descriptor, 01/02/21

Desc.	Binary	Invariant	Size	Complexity	Robustness	Patents
SIFT	no	Scale, rotation, mild comp.	128 float= 512 B (128B in some impl)	High	High	Yes
SURF	no	Scale, rotation, mild comp.	64 float = 256 B or 128 float = 512 B (conv. to bytes sometimes)	Medium/High	High	No
BRIEF	yes	Scale, rotation	256 or 128 bits	low	Low	no
BRISK	yes	Scale, rotation	512 bits	low	low	no
FREAK	yes	Scale, rotation	512 bits	very low	low	no

Figure 1: Common algorithms for feature description.

2 Data

The data of the experiment are collected in 10 files for both regimes (still and spinning wheel). Each file contains a dataset with two columns, being the first the time tags in *machine time* ($\simeq 81ps$) and the other the channel, like this:

$$\begin{array}{cc} 0, & 1 \\ 18009, & 1 \\ 18835, & 1 \\ \dots & \dots \end{array}$$

In this case there is only one channel collecting the data so the last column isn't useful. To convert the time tags from machine time to actual time it's sufficient to multiply the first column by $81ps$ obtaining the time tags in seconds. The data can be merged together by putting in column all the datasets and making sure that at each iteration the time begins where it left (for example, if the first dataset ends at $2s$ the second must begin at $2s$).

3 Development

Having the time tags in seconds it is sufficient to take a time bin ($T = 15\mu s$), and count how many of the dataset "clicks" fall inside the range $[nT, (n+1)T]$, with $n = 0, 1, 2\dots$. Plotting on the x-axis the number of clicks and on the y-axis the number of occurrences inside the time slot, an histogram is derived. The empirical distribution $P(n)$ can be found by dividing the number of clicks $h(n)$ by N, number of occurrences (a normalization factor): $P(n) = \dfrac{h(n)}{N}$. The result is shown on figure 2 and 3.

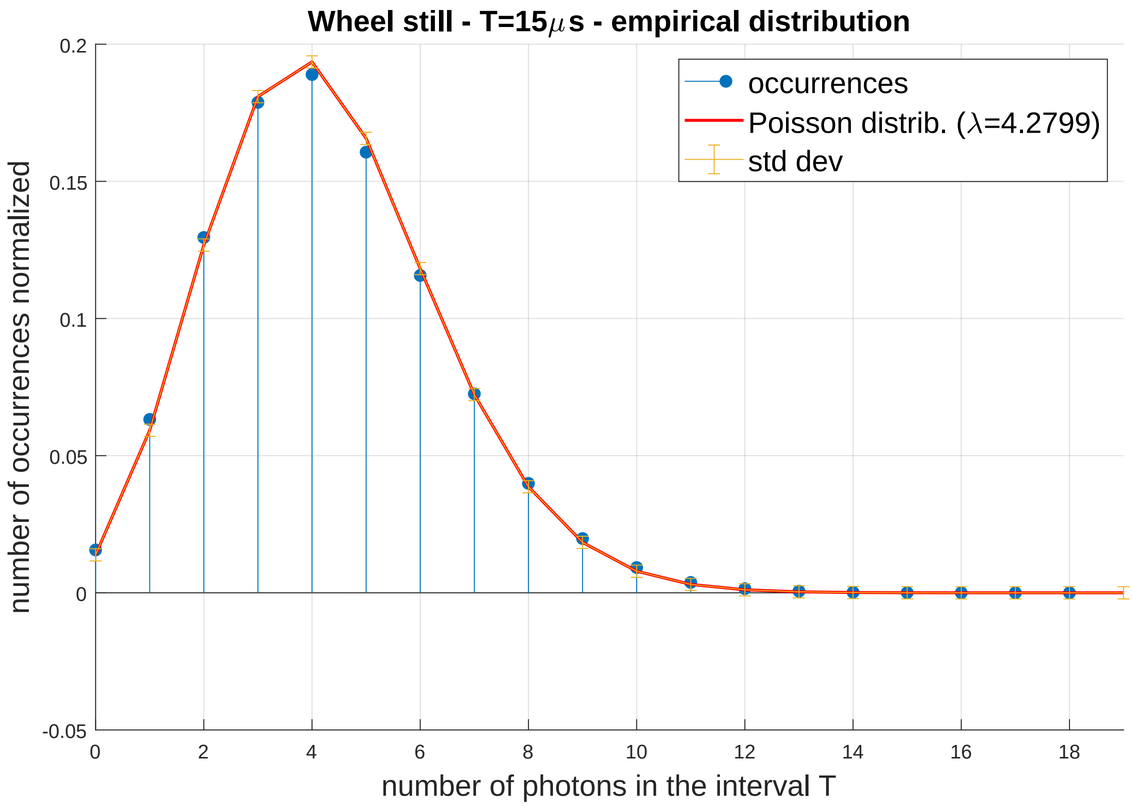

Figure 2: Empirical distribution with $T = 15\mu s$.

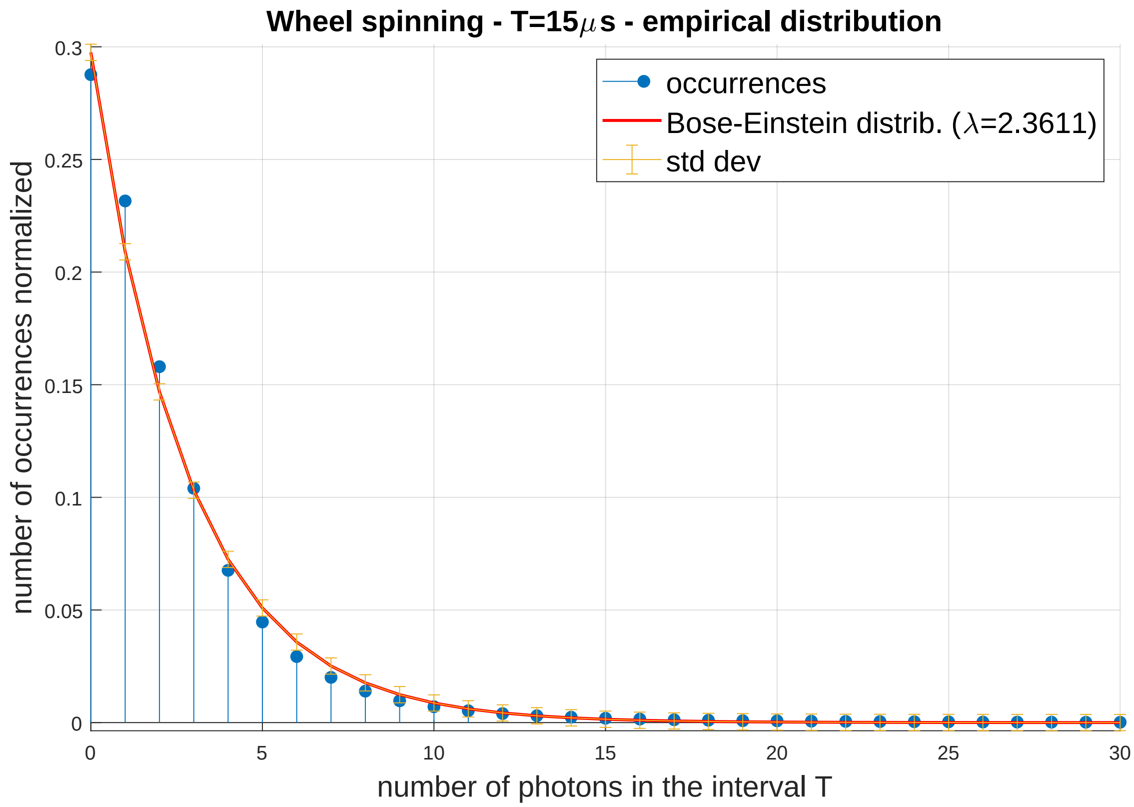

Figure 3: Empirical distribution with $T = 15\mu s$.

In both regimes a plot of the relative distribution is shown for comparison.

By changing the size of the time bin, for example to $T = 30\mu s$, it is possible to see how the values spread out and the theorical distributions tend to drift away from the empirical data (by increasing the time bin it is increasing the mean and also the variance). The results are shown on figures 4 and 5.

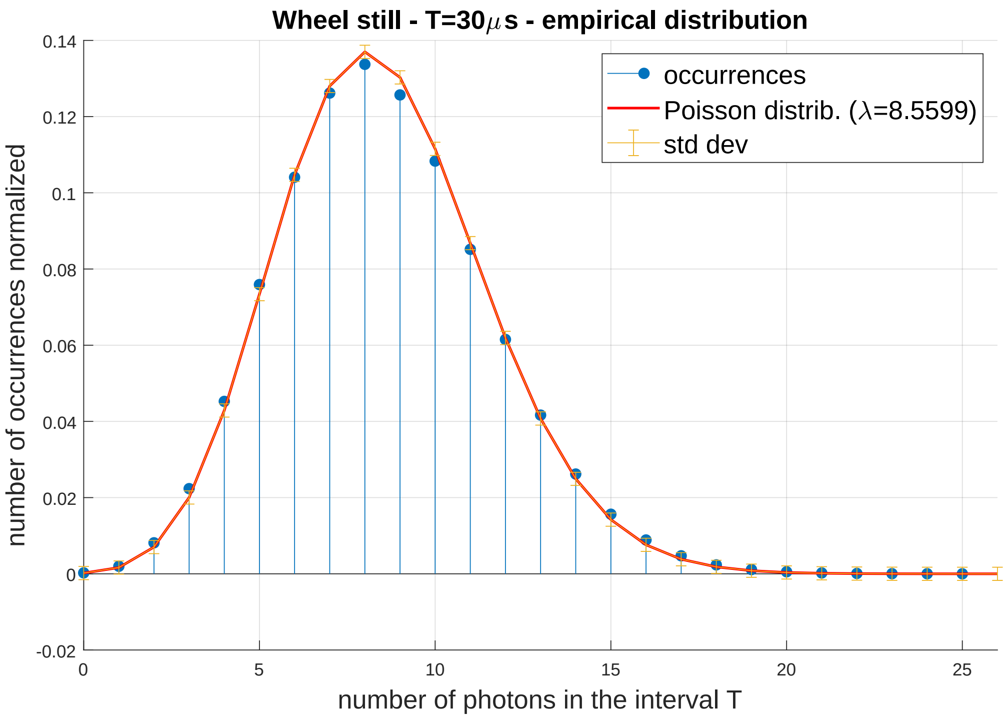

Figure 4: Empirical distribution with $T = 30\mu s$.

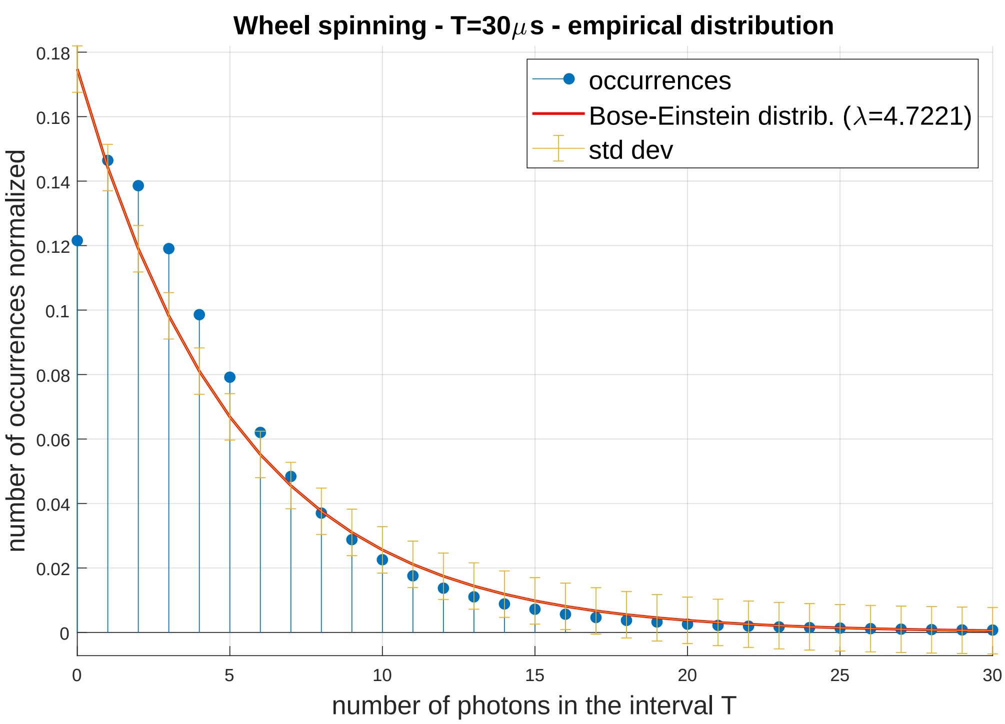

Figure 5: Empirical distribution with $T = 30\mu s$.

4 Momenta Analysis

A statistical comparison between the theoretical and empirical model can be performed by analyzing superior orders of the distributions momenta.

4.1 Poisson distribution

The first four momenta are described as:

$$m_{1,P} = \sum_n \frac{n \cdot e^{-\lambda} \cdot \lambda^n}{n!} \implies \lambda$$

$$m_{2,P} = \sum_n \frac{(n-\lambda)^2 \cdot e^{-\lambda} \cdot \lambda^n}{n!} \implies \lambda$$

$$m_{3,P} = \sum_n \frac{(n-\lambda)^3 \cdot e^{-\lambda} \cdot \lambda^n}{n!} \implies \lambda$$

$$m_{4,P} = \sum_n \frac{(n-\lambda)^4 \cdot e^{-\lambda} \cdot \lambda^n}{n!} \implies \lambda + 3\lambda^2$$

where λ is the mean of the Poisson distribution and $n = 0, 1, 2..$. The comparison can be done by subtracting the terms on the left with the terms on the right. The results are:

$$|m_{1,P} - \lambda| \simeq 1.0 \cdot 10^{-5}$$

$$|m_{2,P} - \lambda| \simeq 1.4 \cdot 10^{-4}$$

$$|m_{3,P} - \lambda| \simeq 2.6 \cdot 10^{-3}$$

$$|m_{4,P} - (\lambda + 3\lambda^2)| \simeq 4.5 \cdot 10^{-2}$$

Values can be affected by the choice of n: the more n increases, the more the differences become close to zero (for the calculations was set to its maximum value).

4.2 Bose-Einstein distribution

The first four momenta are described as:

$$m_{1,BE} = \sum_n \frac{n}{1+\lambda} \cdot \left(\frac{\lambda}{1+\lambda}\right)^n \implies \lambda$$
$$m_{2,BE} = \sum_n \frac{(n-\lambda)^2}{1+\lambda} \cdot \left(\frac{\lambda}{1+\lambda}\right)^n \implies \lambda + \lambda^2$$
$$m_{3,BE} = \sum_n \frac{(n-\lambda)^3}{1+\lambda} \cdot \left(\frac{\lambda}{1+\lambda}\right)^n \implies \lambda + 3\lambda^2 + 2\lambda^3$$
$$m_{4,BE} = \sum_n \frac{(n-\lambda)^4}{1+\lambda} \cdot \left(\frac{\lambda}{1+\lambda}\right)^n \implies \lambda + 10\lambda^2 + 18\lambda^3 + 9\lambda^4$$

where λ is the mean of the Bose-Einstein distribution and $n = 0, 1, 2..$. The comparison can be done by subtracting the terms on the left with the terms on the right. The results are:

$$|m_{1,BE} - \lambda| \simeq 1.3 \cdot 10^{-9}$$
$$|m_{2,BE} - (\lambda + \lambda^2)| \simeq 9.0 \cdot 10^{-8}$$
$$|m_{3,BE} - (\lambda + 3\lambda^2 + 2\lambda^3)| \simeq 6.3 \cdot 10^{-6}$$
$$|m_{4,BE} - (\lambda + 10\lambda^2 + 18\lambda^3 + 9\lambda^4)| \simeq 4.5 \cdot 10^{-4}$$

Values can be affected by the choice of n: the more n increases, the more the differences become close to zero (for the calculations was set to its maximum value).

5 Conclusions

The plots and the momenta analysis display a strong similarity between the data collected and the theoretical models. The Arecchi experiment exhibits the difference between a coherent and a thermal source: the statistical description of a coherent source follows a Poisson distribution of mean λ, while a thermal source follows a Bose-Einstein distribution of parameter λ. This can be verified empirically because by reducing the time bin the less photons are collected and the Bose-Einstein distribution follows this kind of behaviour (strong peak around 0), while the Poisson distribution tends to keep a bell-shape around the mean (like a coherent light source) and by changing T all it does is shifting and spreading the “bell”. Note that the differences between the empirical and theoretical momenta are really small: the twos can be considered approximately equal despite some error due to the small set of n.

6 MATLAB Code

```
%% STILL

clear all;
close all;
clc;


files = dir('Wheel_still_2020\*.txt');

datasets=[];
for i=1:length(files)

    data=importdata(strcat(files(i).folder, '\', files(i).name));
    if i>2 && length(data(:,1))<length(datasets(:,i-1))
        datasets(length(data(:,1))+1:length(datasets(:,i-1)),:)=[];
    end
    datasets(:,i)=data(:,1);

    clear data;
end
clear files i;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        T=15; %us
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


tags=datasets.*81e-12*1e6; %tags in us (microseconds)

clicks=tags(:,1);
for i=2:10
    clicks=[clicks; tags(2:length(tags(:,i)),i)+clicks(length(clicks))];
end
tags=clicks; clear clicks i;



m=discretize(tags,0:T:max(tags));
figure('Renderer', 'painters', 'Position', [500 300 900 600]);
hold on; grid on;

    [GC,GR] = groupcounts(m);
    GC(length(GC)+1:max(GR))=0; %zero padding

    [vals_y,¬]=histcounts(GC, 'BinMethod', 'Integers');

    vals_x=0:length(vals_y)-1;

    N=sum(vals_y);

    vals_x=vals_x';
    vals_y=vals_y';

    stem(vals_x,vals_y/N, 'filled')

    lam=length(tags)/tags(length(tags))*T;


n=0:length(vals_x); n=n';
poisson=exp(-lam).*(lam.^n)./factorial(n);

plot(n,poisson, 'r', 'LineWidth', 1.5)

```

```
xlim([0 length(vals_x)])
title(strcat('Wheel still - T=', string(T), ...
    '\mus - empirical distribution'), 'FontSize', 15)
ylabel('number of occurrences normalized', 'FontSize', 15);
xlabel('number of photons in the interval T', 'FontSize', 15);

pbaspect([1.5 1 1])
clear i j initial final datasets;


%%%%%%%% MOMENTA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mu=lam;
m1P=sum((n.*exp(-mu).*mu.^n)./(factorial(n)));
diff1P=m1P-mu;
m2P=sum(((n-mu).^2.*exp(-mu).*mu.^n)./(factorial(n)));
diff2P=m2P-mu;
m3P=sum(((n-mu).^3.*exp(-mu).*mu.^n)./(factorial(n)));
diff3P=m3P-mu;
m4P=sum(((n-mu).^4.*exp(-mu).*mu.^n)./(factorial(n)));
diff4P=m4P-(mu+3*mu^2);

errorbar(n,poisson,...
    std(vals_y/N-poisson(1:length(vals_y)))*ones(size(poisson)))

legend('occurrences',strcat('Poisson distrib. (\lambda=', string(lam),...
    ')'), 'std dev', 'FontSize', 15);

%% SPINNING

clearvars -except T;
clc;


files = dir('Wheel_spinning_2020\*.txt');

datasets=[];
for i=1:length(files)

    data=importdata(strcat(files(i).folder, '\', files(i).name));
    if i>2 && length(data(:,1))<length(datasets(:,i-1))
        datasets(length(data(:,1))+1:length(datasets(:,i-1)),:)=[];
    end
    datasets(:,i)=data(:,1);

    clear data;
end
clear files i;

tags=datasets.*81e-12*1e6; %tags in us (microseconds)

clicks=tags(:,1);
for i=2:10
    clicks=[clicks; tags(2:length(tags(:,i)),i)+clicks(length(clicks))];
end
tags=clicks; clear clicks i;

m=discretize(tags,0:T:max(tags));
figure('Renderer', 'painters', 'Position', [500 300 900 600]);
hold on; grid on;

    [GC,GR] = groupcounts(m);
    GC(length(GC)+1:max(GR))=0; %zero padding


    [vals_y,¬]=histcounts(GC, 'BinMethod', 'Integers');

    N=sum(vals_y);
```

```

    vals_x=0:length(vals_y)-1;

    vals_x=vals_x';
    vals_y=vals_y';

    stem(vals_x,vals_y/N, 'filled')

     lam=length(tags)/tags(length(tags))*T;

n=0:length(vals_x); n=n';
BE=(1/(lam+1)).*(lam/(lam+1)).^n;

plot(n,BE, 'r', 'LineWidth', 1.5)

title(strcat('Wheel spinning - T=', string(T), ...
    '\mus - empirical distribution'), 'FontSize', 15)
ylabel('number of occurrences normalized', 'FontSize', 15);
xlabel('number of photons in the interval T', 'FontSize', 15);
xlim([0 30]); ylim([-inf inf]);

pbaspect([1.5 1 1])
clear i j initial final datasets h;

%%%%%%%%%% MOMENTA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mu=lam;
m1BE=sum(n.*1./(1+mu).*(mu./(1+mu)).^n);
diff1BE=m1BE-mu;
m2BE=sum((n-mu).^2.*1./(1+mu).*(mu./(1+mu)).^n);
diff2BE=m2BE-(mu+mu^2);
m3BE=sum((n-mu).^3.*1./(1+mu).*(mu./(1+mu)).^n);
diff3BE=m3BE-(mu+3*mu^2+2*mu^3);
m4BE=sum((n-mu).^4.*1./(1+mu).*(mu./(1+mu)).^n);
diff4BE=m4BE-(mu + 10*mu^2 + 18*mu^3 + 9*mu^4);

errorbar(n,BE,std(vals_y/N-BE(1:length(vals_y)))*ones(size(BE)))

legend('occurrences',strcat('Bose-Einstein distrib. (\lambda=', ...
    string(lam), ')'), 'std dev', 'FontSize', 15)
```