National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cloud Computing
(MSCCLOUD1_JAN21)

Sonal Dipak Patil
Student ID: X21120285

School of Computing
National College of Ireland

Supervisor: Rashid Mijumbi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sonal Dipak Patil
Student ID:	X21120285
Programme:	Masters in Cloud Computing **Year:** Jan 2022-23
Module:	Research Project
Lecturer:	Mr. Rashid Mijumbi
Submission Due Date:	15th December 2022
Project Title:	Efficient cloud data transfer via prediction and selective data dropping in a three-tier distributed cloud architecture for smart traffic management
Word Count:	767 **Page Count: 10**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Sonal Dipak Patil

Date: 15th December 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sonal Dipak Patil
Student ID: X21120285

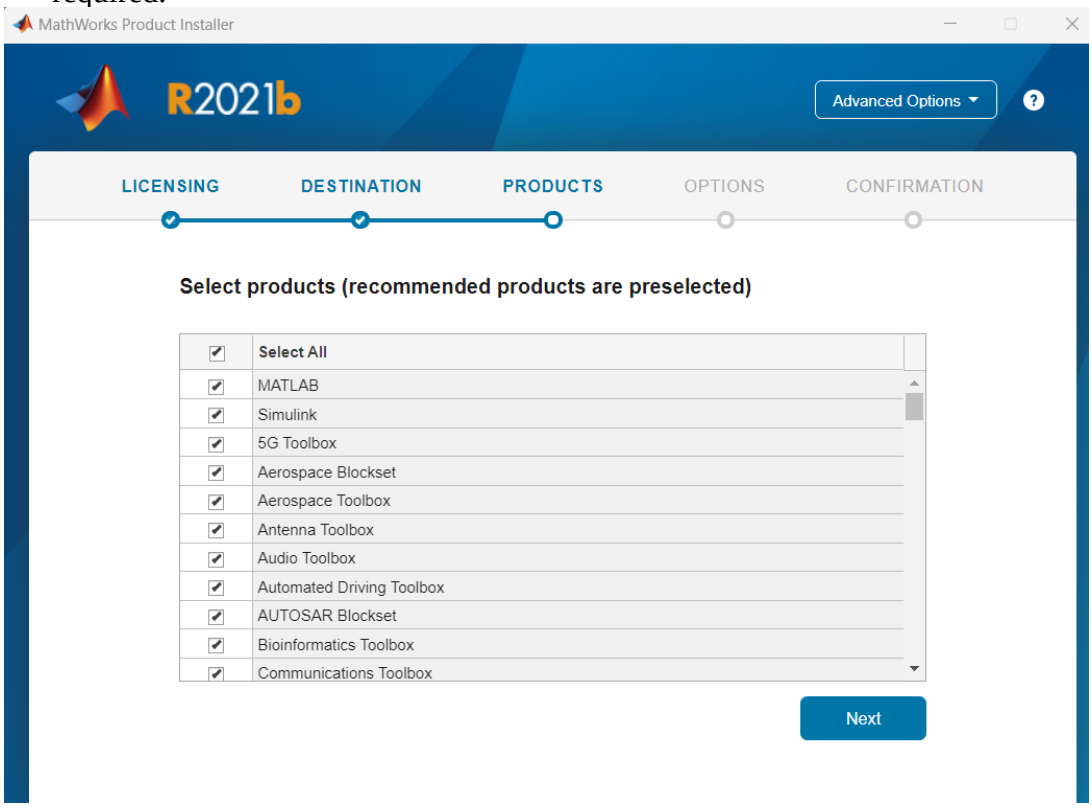
1 Introduction

The configuration manual contains the system requirements for implementing the research project. It contains the software and hardware specifications and configurations made for the execution of code. It also explains the implementation of the code used to achieve the results.

2 System Configuration

2.1 Software Specification

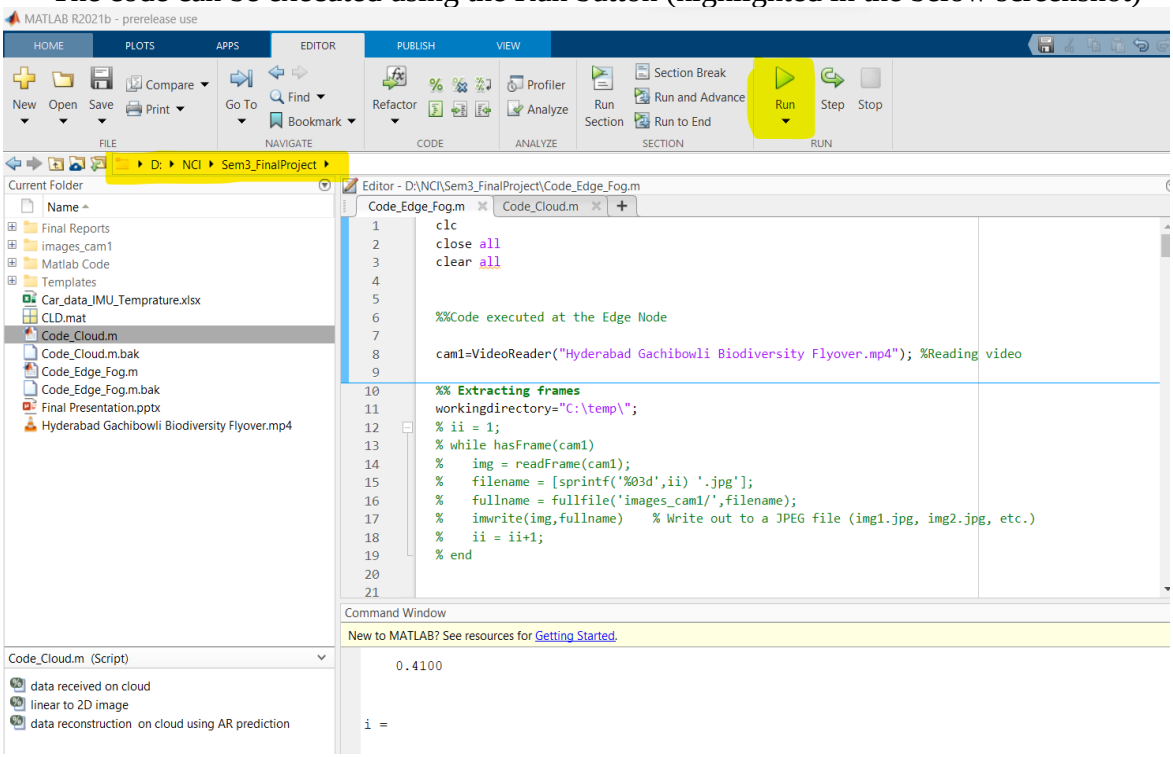
- Obtain student version account from MathWorks.com
- Install MATLAB installer for MATLAB R2021b version from MathWorks Downloader
- Select the folder to install the software and run the installer along with the Products required.



- Once installed, open Matlab and navigate to the location where the code is located as by clicking on the highlighted section (in this case my source is

D:\NCI\Sem3_FinalProject) and double click on the code files, Code_Edge_Fog.m and Code_Cloud.m

- Code_Edge_Fog.m contains the code executed at the Edge and Fog nodes and Code_Cloud.m contains code executed in the cloud.
- The code can be executed using the Run button (highlighted in the below screenshot)



2.2 Hardware Specification

- Dell Inspiron 15 (5510), 512GB SSD, DDR4 8GB RAM
- Processor: Intel Core i5 with 3.10GHz clock speed
- GPU: NVIDIA MX450 2GB GDDR5
- Operating System: Windows 11

3 Data Generation

- Video CCTV footage of a road located in India named “Hyderabad Gachibowli Biodiversity Flyover” was downloaded from YouTube.
- Create a folder named images_cam1 in the same directory where the code is. This is where all the extracted video frames will be stored. For the test video provided there are 2999 frames.
- Below code was executed to extract the frames from the CCTV video footage and then read the frames back into the buffer for further implementation in Edge node.

```
%%Code executed at the Edge Node

cam1=VideoReader("Hyderabad Gachibowli Biodiversity Flyover.mp4"); %Reading video

%% Extracting frames
workingdirectory="C:\temp\";
ii = 1;
while hasFrame(cam1)
   img = readFrame(cam1);
   filename = [sprintf('%03d',ii) '.jpg'];
   fullname = fullfile('images_cam1/',filename);
   imwrite(img,fullname)    % Write out to a JPEG file (img1.jpg, img2.jpg, etc.)
   ii = ii+1;
end


imageNames = dir(fullfile("images_cam1/",'*.jpg')); %% Read the frames images
imageNames = {imageNames.name}';
[totalnooframesframes, n]=size(imageNames);
tempimageNames=(imageNames{1,1});
i=imread(strcat("D:\NCI\Sem3_FinalProject\images_cam1\",tempimageNames));
imshow(i)
```

- Vehicle sensor dataset was downloaded from Kaggle (URL mentioned) and stored in local drive where the code is stored. https://www.kaggle.com/datasets/indiadatabases/car-temprature-and-imu-data.
- Below code is executed to fetch the sensor data and store only relevant (accelerometer Gx, Gy, Gz and gyroscope readings Anglex, Angley, Anglez).

```
%%Car sensor data received at the fog
T=xlsread("Car_data_IMU_Temprature.xlsx");
[m n]=size(T);
numberofvehicles=m;
maxsensorsinacar=n;
%% Total size computation
totaldatasizeatfogfromcar=m*n*8;
timepointindex=1:1:m;
finaluncompressedsize=totaldatasizeatfogfromcar+totalsize;
%% accelerometer and gyroscope reading extraction
for i=1:1:numberofvehicles
    Anglex(i)=T(i,6);
    Angley(i)=T(i,7);
    Anglez(i)=T(i,8);
    Gx(i)=T(i,10);
    Gy(i)=T(i,11);
    Gz(i)=T(i,12);
end
```

4 Implementation

4.1 Implementation at Edge node:

- Below code snippet is used to read each image from the buffer and then the subtraction of 2 consecutive frames is carried out.
- All the differences are stored in the variable.
- Maximum value of difference is found from the stored variable.

```
%% Calculate total frame buffer size
[m n z]=size(i);
totalsize=m*n*z*totalnooframesframes*8;

for itr=1:1:totalnooframesframes-1
    tempimageNames=(imageNames{itr,1});
    im=imread(strcat("D:\NCI\Sem3_FinalProject\images_cam1\",tempimageNames));
    tempimageNames=(imageNames{itr+1,1});
    im2=imread(strcat("D:\NCI\Sem3_FinalProject\images_cam1\",tempimageNames));
    imdiff=abs(double(im)-double(im2));  %Frame subtraction with the difference
    imdiff=uint8(imdiff);
    imdiff=imdiff(:);

    %Difference Variable
    diff=sum(sum(sum(imdiff)));
    val(itr)=diff;

end


%max differance
maxvalinval=max(val)

netlength=length(val);

framestobetrasmitted=1;
count=1;
```

- Next code snippet is used normalize the values which brings all the values between 0 to 1.
- The vehicle detection limit is set to 0.7 and if the value (vnormal) is greater than 0.7, it means that the frame consists of new information and can be transmitted.
- Frames to be transmitted are then written to a file and data drop rate is calculated.
- Experiments were carried out to find out the best vehicle detection limit(varying it from 0.5 to 0.8) and 0.7 was selected as the optimized value.

```
%Normalization in amplitude
for i=1:1:netlength
    vnormval(i)=val(i)/maxvalinval;
    %selection level- Vehicle Detection Limit=0.7
    if(vnormval(i)>0.7)
        framestobetrasmitted=[framestobetrasmitted i];
        count=count+1;
    end
end


ii = 1;
while hasFrame(cam1)
   img = readFrame(cam1);
   for i=1:1:length(framestobetrasmitted)
       if(framestobetrasmitted(i)==ii)
           % frames to be transmitted are saved here (Run once)
           disp("frame transmitted to fog")
           disp(ii)
           imshow(img)
           filename = [sprintf('%03d',ii) '.jpg'];
           fullnameC = fullfile('images_cam1/',filename);
           imwrite(img,fullname)
       end
   end
   ii = ii+1;

end
%dropped data frames
datadropratecam1=1-(count/totalnooffframesframes);
```

4.2 Implementation at the Fog Node

- After reading all the frames transmitted from the Edge node, the frames are resized to achieve compression to reduce computation load.
- All the resized frames are stored in variable imR.

```
%% resize image to reduce computation load
[m n z]=size(i);
totalsize=m/10*n/10*z*totalnooffframesframes*8;
 imR=[m/10,n/10];
for itr=1:33:totalnooffframesframes
    tempimageNames=(imageNames{itr,1});
    im=imread(strcat("D:\NCI\Sem3_FinalProject\images_cam1\",tempimageNames));
    im=imresize(im,0.1);
    im=im(:)';
    imR=[imR im];
end
```

- Below snippet is used to validate the sensor data. For both the sensors, the invalid and missing values are changed to an invalid indicator (1200 in this case) as it is easy to skip such values.
- The same code is repeated for all the sensors data.

```
%% range validation for each sensor
totaldatalengthsensor= length(Anglex);
TF = isnan( Anglex );
for itr=1:1:totaldatalengthsensor

    if(TF(itr)==1)
     Anglex(itr)=1200;
    end
    if(Anglex(itr)>360)
     Anglex(itr)=1200;
    end
end

totaldatalengthsensor= length(Angley);
TF = isnan( Angley );
for itr=1:1:totaldatalengthsensor

    if(TF(itr)==1)
     Angley(itr)=1200;
    end
    if(Angley(itr)>360)
     Angley(itr)=1200;
    end
end
```

- After removing the invalid values, the repeated values also need to be dropped. Below snippet is used to drop such repeated values
- The same process is carried out for all the sensors' data (gyroscope and accelerometer)

```
%% data repeatation
newlen=length(Anglex)
oldval=Anglex(1);
counter=1;
for i=1:1:newlen
    newval=Anglex(i);
    if(newval==oldval)
        oldval=newval;
    else
    tramittanglex(counter)=Anglex(i);
    counter=counter+1
    oldval=newval;
    end
end

newlen=length(Angley)
oldval=Angley(1);
counter=1;
for i=1:1:newlen
    newval=Angley(i);
    if(newval==oldval)
        oldval=newval;
    else
    tramittangley(counter)=Angley(i);
    counter=counter+1
    oldval=newval;
    end
end
```

- Once all the sensors data is filtered in the previous steps, interlacing of the data is started which includes adding an escape character to identify the type of the data.

- Interlaced data is stored in variable dataCLD.

```
%% Escape character based Interlacing of data
dataCLD=double(imR(1:2));
mzip=double(imR(1:1));
nzip=double(imR(2:2));
imR=imR(3:length(imR));
for tp=96:1:180 %length(timepointindex)
    if(Gx(tp)==1200)
    else
        dataCLD=[dataCLD 0 Gx(tp)];

        if(Gy(tp)==1200)
        else
            dataCLD=[dataCLD 1 Gy(tp)];

            if(Gz(tp)==1200)
            else
                dataCLD=[dataCLD 2 Gz(tp)];

                if(Anglex(tp)==1200)
                else
                    dataCLD=[dataCLD 3 Anglex(tp)];

                    if(Angley(tp)==1200)
                    else
                        dataCLD=[dataCLD 4 Angley(tp)];

                        if(Anglez(tp)==1200)
                        else
                           dataCLD=[dataCLD 5 Anglez(tp)] ;

                           dataCLD=[dataCLD double(imR(1:(mzip*nzip*3)))];
                           dataCLD=double(dataCLD);
                           imR=imR((mzip*nzip*3+1):length(imR));
                        end
                    end
                end
            end
        end
    end
```

- All the interlaced data is saved to the local drive in the same working directory with name **CLD.mat** which is a binary file.

```
dataCLD=double(dataCLD);
DATADROPRATEEDGE=length(dataCLD)/finaluncompressedsize;

%% Store interlaced data
save('CLD.mat','dataCLD')
```

4.3 Implementation in Cloud

- Below code snippet is used to load the interlaced data which was stored in the local directory from CLD.mat.

```
%% data received on cloud
load("CLD.mat")
jams=0;
m=double(dataCLD(1:1));
dataCLD=dataCLD(2:length(dataCLD));
n=double(dataCLD(1:1));
dataCLD=dataCLD(2:length(dataCLD));
counter=1;
five=5;
```

- Below code snippet extracts the data from the interlaced data stored retrieved in above code

```
% search escape character for frame
while(espChar==5)
    if(length(dataCLD)>5)
        %CAR SENSOR DATA reproduction
        zero=double(dataCLD(1:1));
        dataCLD=dataCLD(2:length(dataCLD));
        Gx(counter)=double(dataCLD(1:1));
        dataCLD=dataCLD(2:length(dataCLD));
        one=double(dataCLD(1:1));
        dataCLD=dataCLD(2:length(dataCLD));
        Gy(counter)=double(dataCLD(1:1));
        dataCLD=dataCLD(2:length(dataCLD));
        two=double(dataCLD(1:1));
        dataCLD=dataCLD(2:length(dataCLD));
        Gz(counter)=double(dataCLD(1:1));
        dataCLD=dataCLD(2:length(dataCLD));
        three=double(dataCLD(1:1));
        dataCLD=dataCLD(2:length(dataCLD));
        Anglex(counter)=double(dataCLD(1:1));
        dataCLD=dataCLD(2:length(dataCLD));
        four=double(dataCLD(1:1));
        dataCLD=dataCLD(2:length(dataCLD));
        Angley(counter)=double(dataCLD(1:1));
        dataCLD=dataCLD(2:length(dataCLD));
        espChar=double(dataCLD(1:1));
        dataCLD=dataCLD(2:length(dataCLD));
        Anglez(counter)=double(dataCLD(1:1));
        dataCLD=dataCLD(2:length(dataCLD));
        %frame data
        Total1frame=double(m*n*3);
        framedata=uint8(dataCLD(1:Total1frame));
        dataCLD=dataCLD(Total1frame+1:length(dataCLD));
        %% linear to 2D image
        frame = reshape(framedata,m,n,3);
        imshow(frame);
        drawnow
        counter=counter+1
    else
        break
        espChar=6;
    end
end
```

- Traffic management is carried out based on the ratio between each gyroscopic value and the maximum value in that particular direction which is defined as the traffic factor.
- Experiments are carried out to find the optimum value of the traffic factor. All the results are plotted and published in the results section.

```
%%Traffic management
lengthGx=length(Gx);
maxGx=abs(max(Gx));
for i=1:1:lengthGx
    trafficfactor=abs(Gx(i)/max(Gx))
    if(trafficfactor<0.4)
        display("Traffic jam on this road"); jams=jams+1;
        pause(1)
    else
        display(i);
    end
end

lengthGy=length(Gy);
maxGy=abs(max(Gy));
for i=1:1:lengthGy
    trafficfactor=abs(Gy(i)/max(Gy))
    if(trafficfactor<0.4)
        display("Traffic jam on this road"); jams=jams+1;
        pause(1)
    else
        display(i);
    end
end

lengthGz=length(Gz);
maxGz=abs(max(Gz));
for i=1:1:lengthGz
    trafficfactor=abs(Gz(i)/max(Gz))
    if(trafficfactor<0.4)
        display("Traffic jam on this road"); jams=jams+1;
        pause(1)
    else
        display(i);
    end
end
actGx=Gx(70:length(Gx));
Gx=Gx(1:70);
```

- Using below code the future values are predicted using the Auto-regression process. In the code below, value of Gx is predicted using the previous values.
- The number of previous values used to predict the new value is varied to find out the optimum result with least error.
- In this case, 5 previous values are used to predict the new value.

```
%% data reconstruction  on cloud using AR prediction
data = iddata((Gx)',[]);
plot(data)
sys = ar(data,5);
K = 2;
p = forecast(sys,data,K);
plot(data,'b',p,'r'), legend('measured','forecasted')
```

References

Installation and licensing (no date) *Installation and Licensing Documentation - MathWorks United Kingdom*. Available at: https://uk.mathworks.com/help/install/ (Accessed: December 13, 2022).