UNIVERSITY OF MUMBAI

Practical Journal of

Big Data Analytics, Modern Networking & Computer Vision

M.Sc. (Information Technology) Part-I

Submitted by

PATIL PAVAN MAHESH

Seat No: 26MSCIT203010

DEPARTMENT OF INFORMATION TECHNOLOGY

PILLAI HOC COLLEGE OF ARTS,SCIENCE & COMMERCE(AUTONOMOUS),

RASAYANI, 410207 MAHARASTRA

(Affiliated to Mumbai University)

RASAYANI-410207, MAHARASHTRA

NAAC Accredited with 'A+' Grade in Cycle

2025-2026

MAHATMA EDUCATION SOCIETY'S
Pillai HOC College of Arts, Science & Commerce (Autonomous),
Rasayani 410207
(Affiliated to Mumbai University)
NAAC Accredited with 'A+' Grade in Cycle

DEPARTMENT OF

INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the experiment work entered in this journal is as per the syllabus in **M.Sc. (Information Technology) Part-I, Semester-II**; class prescribed by University of Mumbai for **Big Data Analytics**, of Mahatma Education Society's **Pillai HOC College of Arts, Science & Commerce(Autonomous), Rasayani** by **Patil Pavan Mahesh** during Academic year **2025-2026.**
Seat No: **26MSCIT203010**

Subject In-Charge **Co-Ordinator**

External Examiner **Principal**

Date **College Seal**

BIG DATA ANALYTICS

INDEX

PRACTICAL NO: 01

AIM: REGRESSION MODEL Import a data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in an institute based on his or her GRE score, GPA obtained and rank of the student. Also check the model is fit or not. require (foreign), require(MASS).

To perform Logistic Regression in R, we will use the UCLA Graduate School Admissions dataset, which is a standard dataset for this type of analysis.

Despite its name, **Logistic Regression** is used for **classification** (e.g., Admit vs. Not Admit). In R, we achieve this using the `glm()` function with a "binomial" family.

Step 1: Load Libraries and Import Data

We will use `foreign` to handle data structures and `MASS` for statistical functions.

How to execute: Open RStudio, go to `File > New File > R Script`, and paste the following:

R
```
# Load required libraries
require(foreign)
require(MASS)

# Import dataset from web storage
# This is a CSV file containing admit, gre, gpa, and rank
admission_data <- read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")

# View the first few rows
head(admission_data)
```

Step 2: Data Preparation

For a proper logistic regression, the `rank` of the school should be treated as a **factor** (categorical variable), not a continuous number.

R
```
# Convert rank to a factor
admission_data$rank <- factor(admission_data$rank)

# Summary of data to check for missing values
summary(admission_data)
```

Step 3: Run the Logistic Regression

We want to predict `admit` (the dependent variable) based on `gre`, `gpa`, and `rank` (the independent variables).

```
R
# Build the Logit model
# admit ~ . means predict admit using all other columns
logit_model <- glm(admit ~ gre + gpa + rank,
                   data = admission_data,
                   family = "binomial")

# View the results
summary(logit_model)
```

Note for the results:

- **P-values (Pr(>|z|)):** If a p-value is less than **0.05**, that variable significantly affects admission.
- **Coefficients:** A positive estimate means an increase in that score increases the probability of admission.

Step 4: Evaluate Model Fit

In logistic regression, we don't use R^2 like in linear regression. Instead, we check the **Deviance** and use the **Likelihood Ratio Test** to see if the model is a good fit.

```
R
# 1. Check for Overall Model Significance
# Comparing our model to a 'null' model (a model with no predictors)
with(logit_model, pchisq(null.deviance - deviance, df.null - df.residual, lower.tail = FALSE))

# 2. Confusion Matrix to check accuracy
# We predict probabilities; if > 0.5, we classify as 'Admitted'
predictions <- predict(logit_model, type = "response")
predicted_class <- ifelse(predictions > 0.5, 1, 0)

# Create a table of Actual vs Predicted
table(Actual = admission_data$admit, Predicted = predicted_class)
```

Interpreting Fit:

- **Chi-Square Test:** If the result of the `pchisq` code above is very small (less than 0.05), your model fits significantly better than an empty model.
- **Deviance:** Lower Residual Deviance indicates a better fit.

OUT PUT:-

```
head(admission_data)
  admit gre  gpa rank
1     0 380 3.61    3
2     1 660 3.67    3
3     1 800 4.00    1
```

```
   4      1 640 3.19     4


   5      0 520 2.93     4
   6      1 760 3.00     2
   glm(formula = admit ~ gre + gpa + rank, family = "binomial",
       data = admission_data)

   Coefficients:
               Estimate  Std. Error z value Pr(>|z|)
(Intercept) -3.989979   1.139951  -3.500 0.000465 ***
gre          0.002264   0.001094   2.070 0.038465 *
gpa          0.804038   0.331819   2.423 0.015388 *
rank2       -0.675443   0.316490  -2.134 0.032829 *
rank3       -1.340204   0.345306  -3.881 0.000104 ***
rank4       -1.551464   0.417832  -3.713 0.000205 ***
---
   Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

   (Dispersion parameter for binomial family taken to be 1)

       Null deviance: 499.98 on 399 degrees of freedom
   Residual deviance: 458.52 on 394 degrees of freedom
   AIC: 470.52

   Number of Fisher Scoring iterations: 4

    [1] 7.578194e-08
   >
   > # 2. Confusion Matrix to check accuracy
         Predicted
   Actual  0   1
        0 254  19
        1  97  30
```

PRACTICAL NO: 02

AIM : MULTIPLE REGRESSION MODEL Apply multiple regressions, if data have a continuous independent variable. Apply on above dataset.

In Multiple Linear Regression, we predict a **continuous** dependent variable. However, in the admissions dataset used previously, the variable `admit` is binary (0 or 1).

To apply **Multiple Regression** properly to this dataset, we will change our objective: We will predict the **GRE score** (a continuous variable) based on the student's **GPA** and the **Rank** of their school.

Step 1: Prepare the Environment

We will use the same dataset but treat `gre` as our target variable.

R

```
# Ensure libraries are loaded
library(MASS)

# Use the dataset from the previous step
# (Assuming admission_data is already loaded)
# If not: admission_data <- read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")

# Ensure rank is a factor
admission_data$rank <- factor(admission_data$rank)
```

Step 2: Build the Multiple Linear Regression Model

We use the `lm()` function (Linear Model) for continuous outcomes.

R

```
# Formula: gre ~ gpa + rank
# We are seeing how GPA and Rank together influence the GRE score
multiple_reg_model <- lm(gre ~ gpa + rank, data = admission_data)

# Display the statistical results
summary(multiple_reg_model)
```

Step 3: Interpret the Relation Between Variables

When you run the `summary()`, look for the following:

1. **Coefficients (Estimate):** This tells you the "slope." For example, if the GPA estimate is **50**, it means for every 1-point increase in GPA, the GRE score is expected to increase by 50 points, holding school rank constant.
2. **ZP-values (Pr(>|t|)):** If this is <0.05, the variable is a significant predictor of the GRE score.

3. **Adjusted R-squared:** This tells you what percentage of the variation in GRE scores is explained by your model.

Step 4: Evaluate Model Fit (Diagnostics)

To check if a linear model is "fit" for the data, we must check the residuals (the errors).

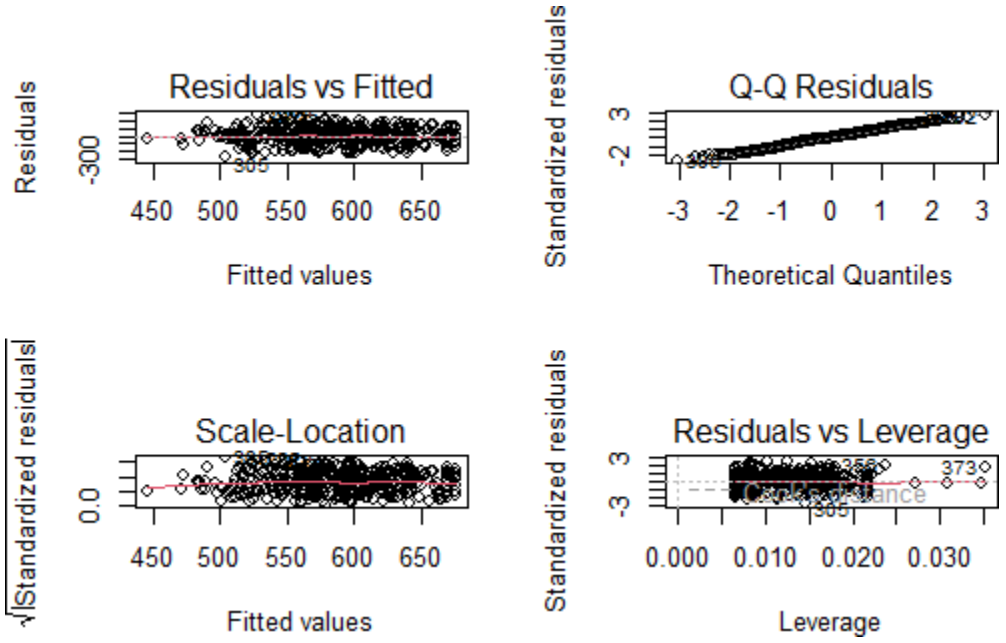
R

```
# Plot diagnostic graphs
# This will show 4 plots; press Enter in the console to see each one
par(mfrow = c(2, 2)) # Arrange plots in a 2x2 grid
plot(multiple_reg_model)
```

How to check if the model is fit:

- **Residuals vs Fitted:** You want to see a random scatter of points. If you see a curve, the relationship might not be linear.
- **Normal Q-Q:** The points should follow the diagonal dashed line. If they do, your errors are normally distributed (which is good).

OUTPUT:



PRACTICAL NO: 03

AIM: CLASSIFICATION MODEL a. Install relevant package for classification. b. Choose classifier for classification problem. c. Evaluate the performance of classifier.

Building a classification model in R is a streamlined process thanks to the `caret` (Classification And REgression Training) package, which provides a consistent interface for hundreds of different algorithms.

Here is a step-by-step guide to building, running, and evaluating a **Random Forest** classifier using the classic `iris` dataset.

Step 1: Install and Load Relevant Packages

First, you need to install the tools. The `caret` package is the industry standard for machine learning in R, and `randomForest` is the specific engine we will use.

How to execute: Type these lines into your R script or the Console and press **Enter**.

```
# Install packages (only need to do this once)
install.packages("caret")
install.packages("randomForest")

# Load the libraries
library(caret)
library(randomForest)
```

Step 2: Prepare the Data

Before choosing a classifier, we must split the data into a **Training set** (to teach the model) and a **Test set** (to see if it actually learned).

```
# Load built-in dataset
data(iris)

# Set a seed for reproducibility (so you get the same results every time)
set.seed(123)

# Create an index to split 80% for training, 20% for testing
trainIndex <- createDataPartition(iris$Species, p = 0.8, list = FALSE)
trainData <- iris[trainIndex, ]
testData  <- iris[-trainIndex, ]
```

Step 3: Choose and Train the Classifier

We will use **Random Forest** because it is robust, handles non-linear data well, and is highly accurate for classification tasks.

R

```
# Train the model
# Species ~ . means "Predict Species using all other variables"
model <- train(Species ~ .,
               data = trainData,
               method = "rf",
               trControl = trainControl(method = "cv", number = 5)) # 5-fold cross-validation

# View model details
print(model)
```

Step 4: Evaluate Performance

To evaluate the model, we use it to predict the species of the "unseen" test data and compare those predictions to the actual results using a **Confusion Matrix**.

R

```
# Make predictions on the test set
predictions <- predict(model, testData)

# Evaluate using a Confusion Matrix
evaluation <- confusionMatrix(predictions, testData$Species)

# Print the results
print(evaluation)
```

OUTPUT:

```
Confusion Matrix and Statistics

            Reference
Prediction    setosa versicolor virginica
  setosa          10          0         0
  versicolor       0         10         2
  virginica        0          0         8

Overall Statistics

               Accuracy : 0.9333
                 95% CI : (0.7793, 0.9918)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : 8.747e-12

                  Kappa : 0.9

 Mcnemar's Test P-Value : NA

Statistics by Class:
```

	Class: setosa	Class: versicolor	Class: virginica
Sensitivity	1.0000	1.0000	0.8000
Specificity	1.0000	0.9000	1.0000
Pos Pred Value	1.0000	0.8333	1.0000
Neg Pred Value	1.0000	1.0000	0.9091
Prevalence	0.3333	0.3333	0.3333
Detection Rate	0.3333	0.3333	0.2667
Detection Prevalence	0.3333	0.4000	0.2667
Balanced Accuracy	1.0000	0.9500	0.9000

Understanding the Output

- **Accuracy:** The percentage of correct predictions. (e.g., 0.96 means 96% correct).
- **Kappa:** A measure of how well the classifier performed compared to how well it would have performed by simply guessing.
- **Confusion Matrix Table:** Shows exactly which species were misclassified (e.g., if a *Versicolor* was mistaken for a *Virginica*).

PRACTICAL NO: 04

AIM: CLUSTERING MODEL a. Clustering algorithms for unsupervised classification. b. Plot the cluster data using R visualizations.

Clustering is an unsupervised learning technique used to group data points based on similarities without using pre-defined labels. The most common algorithm for this is **K-Means Clustering**.

We will use the `iris` dataset again, but we will hide the "Species" column to see if the algorithm can discover the groupings on its own.

Step 1: Prepare the Data

Clustering requires numeric data. Since the iris dataset has 4 numeric columns and 1 categorical (Species), we will strip the species names so the model works "blind."

How to execute: Copy this into a new R Script and run it.

```
R
# Load the dataset
data(iris)

# Remove the Species column (column 5) for unsupervised learning
iris_cluster <- iris[, -5]

# Scale the data (Standardization)
# This ensures variables with larger numbers don't dominate the model
iris_scaled <- scale(iris_cluster)
```

Step 2: Choose the Number of Clusters (The "Elbow" Method)

Before running K-Means, we need to decide on **k** (the number of groups). We use the Elbow Method to find the point where adding more clusters doesn't significantly improve the fit.

```
R
# Determine number of clusters
set.seed(123)
wss <- sapply(1:10, function(k){kmeans(iris_scaled, k, nstart=20 )$tot.withinss})

# Plot the elbow curve
plot(1:10, wss, type="b", pch = 19, frame = FALSE,
    xlab="Number of clusters K",
    ylab="Total within-clusters sum of squares")
```

Look for the "bend" in the arm. For Iris, the bend is usually at ***k = 3****.*

Step 3: Apply the K-Means Algorithm

Now we apply the algorithm using 3 clusters.

R

```
# Apply K-Means
set.seed(123)
km_result <- kmeans(iris_scaled, centers = 3, nstart = 25)

# View the cluster assignments
print(km_result$cluster)
```

Step 4: Visualize the Clusters

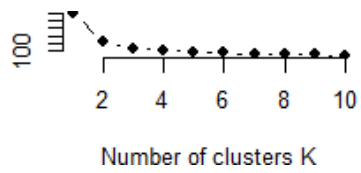
Since we have 4 variables, we can't easily graph them in 2D. We use the `factoextra` package to perform Principal Component Analysis (PCA) automatically and plot the clusters clearly.

How to execute: You may need to install `factoextra` first.

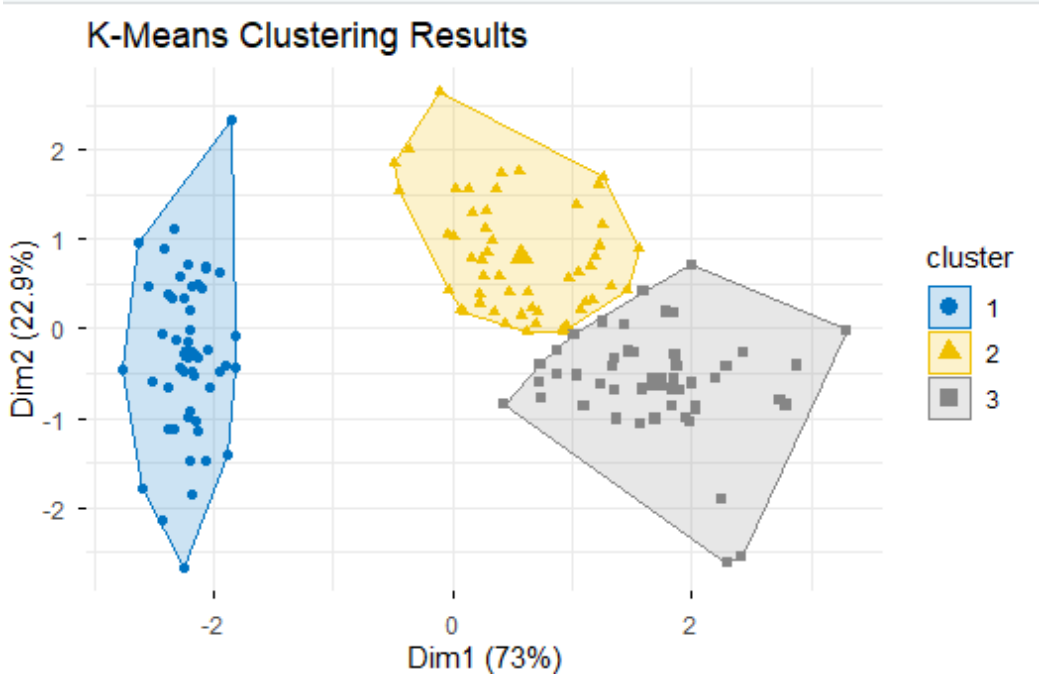
R

```
# Install and load visualization package
if(!require(factoextra)) install.packages("factoextra")
library(factoextra)

# Plot the clusters
fviz_cluster(km_result, data = iris_scaled,
             palette = "jco",
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_minimal(),
             main = "K-Means Clustering Results")
```

OUTPUT:



```
> print(km_result$cluster)
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [34] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 2 2 2 3 2 2 2 2 2 2 2 2 3
 [67] 2 2 2 2 3 2 2 2 2 3 3 3 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2
[100] 2 3 2 3 3 3 3 2 3 3 3 3 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 3 3 3 3 3
[133] 3 2 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 2
```



PRACTICAL NO: 05

AIM: Install, configure and run Hadoop and HDFS ad explore.

1. Install Java JDK 1.8
2. Download Hadoop and extract and place under C drive
3. Set Path in Environment Variables
4. Config files under Hadoop directory
5. Create folder datanode and namenode under data directory
6. Edit HDFS and YARN files
7. Set Java Home environment in Hadoop environment
8. Setup Complete. Test by executing start-all.cmd

There are two ways to install Hadoop, i.e.

9. Single node
10. Multi node

Here, we use multi node cluster.

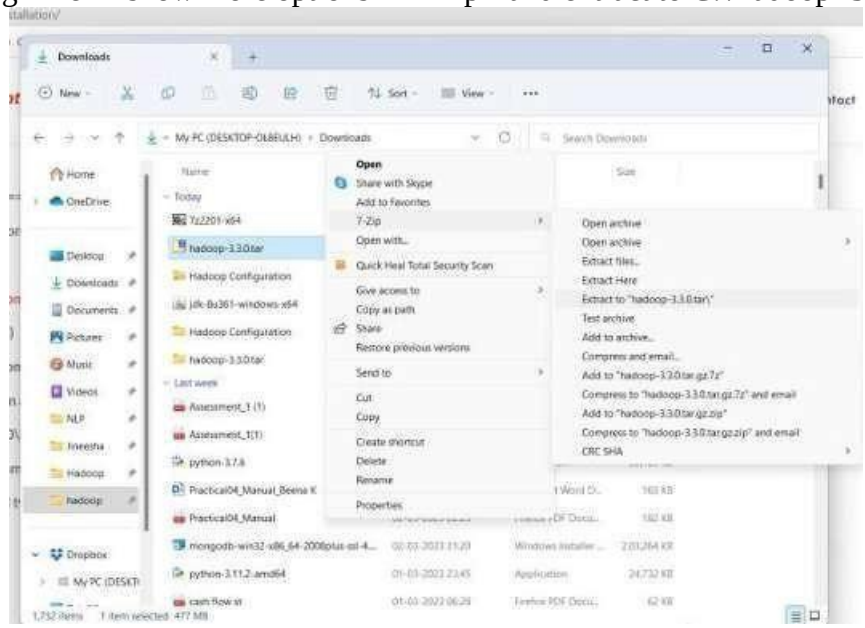
1. Install Java

11. – Java JDK Link to download https://www.oracle.com/java/technologies/javase-jdk8-downloads.html
12. – extract and install Java in C:\Java
13. – open cmd and type -> javac -version

```
C:\Users>cd Beena

C:\Users\Beena>java -version
java version "1.8.0_361"
Java(TM) SE Runtime Environment (build 1.8.0_361-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.361-b09, mixed mode)
```

2. Download Hadoop

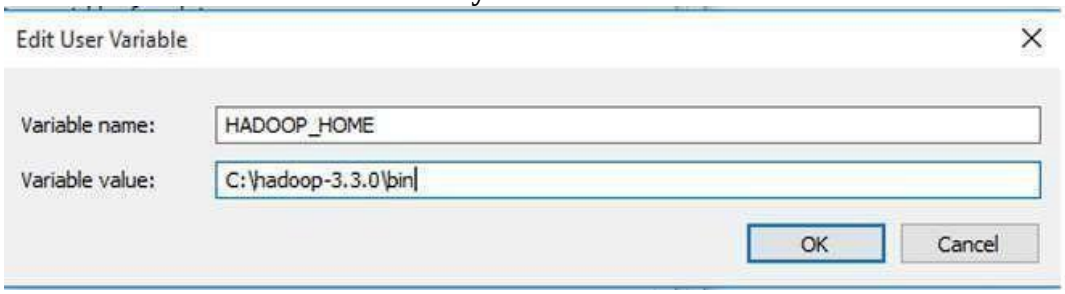
https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.3.0/hadoop- 3.3.0.tar.gz right click .rar.gz file -> show more options -> 7-zip->and extract to C:\Hadoop- 3.3.0\



3. Set the path JAVA_HOME Environment variable

4. Set the path HADOOP_HOME Environment variable

Click on New to both user variables and system variables.



Click on user variable -> path -> edit-> add path for Hadoop and java upto "bin"
Click Ok, Ok, Ok.

5. Configurations

Edit file C:/Hadoop-3.3.0/etc/hadoop/core-site.xml, paste the xml code in folder and save

```
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```

Rename "mapred-site.xml.template" to "mapred-site.xml" and edit this file C:/Hadoop3.3.0/etc/hadoop/mapred-site.xml, paste xml code and save this file.

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

Create folder "data" under "C:\Hadoop-3.3.0"

Create folder "datanode" under "C:\Hadoop-3.3.0\data"

Create folder "namenode" under "C:\Hadoop-3.3.0\data"

Edit file C:\Hadoop-3.3.0/etc/hadoop/hdfs-site.xml, paste xml code and save this file.

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>/hadoop-3.3.0/data/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/hadoop-3.3.0/data/datanode</value>
</property>
</configuration>
```

Edit file C:/Hadoop-3.3.0/etc/hadoop/yarn-site.xml, paste xml code and save this file.

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>127.0.0.1:8032</value>
<name>yarn.resourcemanager.scheduler.address</name>
<value>127.0.0.1:8030</value>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>127.0.0.1:8031</value>
</property>
</configuration>
```

6. **Edit file C:/Hadoop-3.3.0/etc/hadoop/hadoop-env.cmd**

Find "JAVA_HOME=%JAVA_HOME%" and replace it as

set JAVA_HOME="C:\Java\jdk1.8.0_361"

7. **Download "redistributable" package**

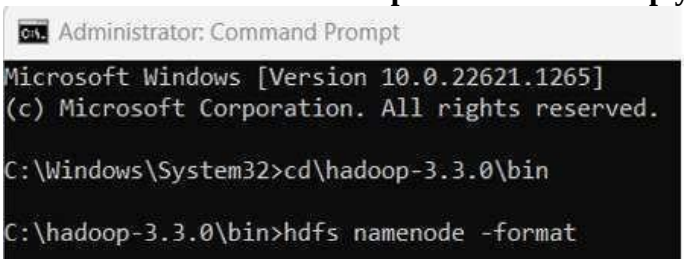
Download and run VC_redist.x64.exe

This is a "redistributable" package of the Visual C runtime code for 64-bit applications, from Microsoft. It contains certain shared code that every application written with Visual C expects to have available on the Windows computer it runs on.

8. **Hadoop Configurations**

Download bin folder from https://github.com/s911415/apache-hadoop-3.1.0-winutils – Copy the bin folder to c:\hadoop-3.3.0. Replace the existing bin folder.

9. copy "hadoop-yarn-server-timelineservice-3.0.3.jar" from ~\hadoop-3.0.3\share\hadoop\yarn\timelineservice to ~\hadoop-3.0.3\share\hadoop\yarn folder.




```
2023-03-07 21:31:34,685 INFO namenode.FSImageFormatProtobuf: Saving image file C:\hadoop-3.3.0\data\namenode\current\fs
mage.ckpt_0000000000000000000 using no compression
2023-03-07 21:31:34,844 INFO namenode.FSImageFormatProtobuf: Image file C:\hadoop-3.3.0\data\namenode\current\fsimage.c
pt_0000000000000000000 of size 400 bytes saved in 0 seconds .
2023-03-07 21:31:34,860 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2023-03-07 21:31:34,869 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2023-03-07 21:31:34,870 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at DESKTOP-OL8EULH/192.168.1.19
************************************************************/
```

10. Format the NameNode

– Open cmd „Run as Administrator" and type command "hdfs namenode –format"

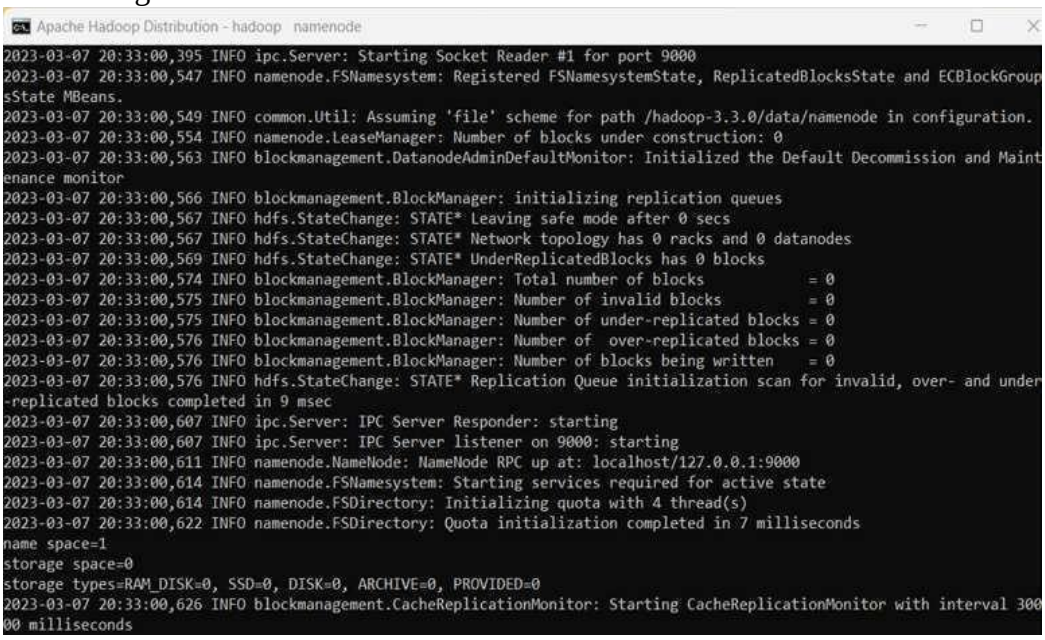
11. Testing

– Open cmd „Run as Administrator" and change directory to C:\Hadoop-3.3.0\sbin

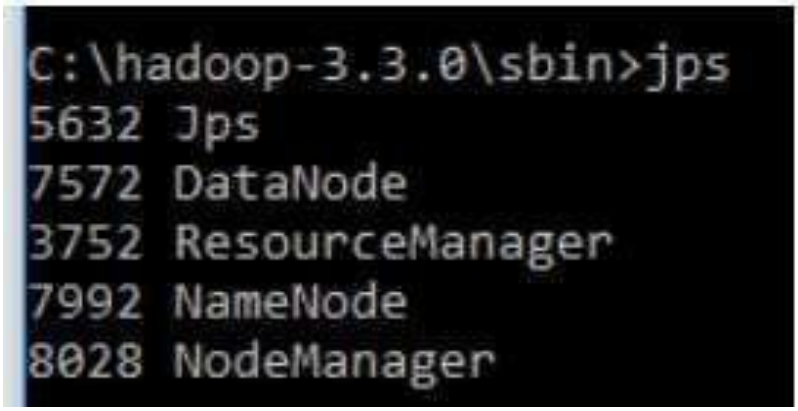
– type start-all.cmd OR type start-dfs.cmd

– type start-yarn.cmd

– You will get 4 more running threads for Datanode, namenode, resouce manager and node manager

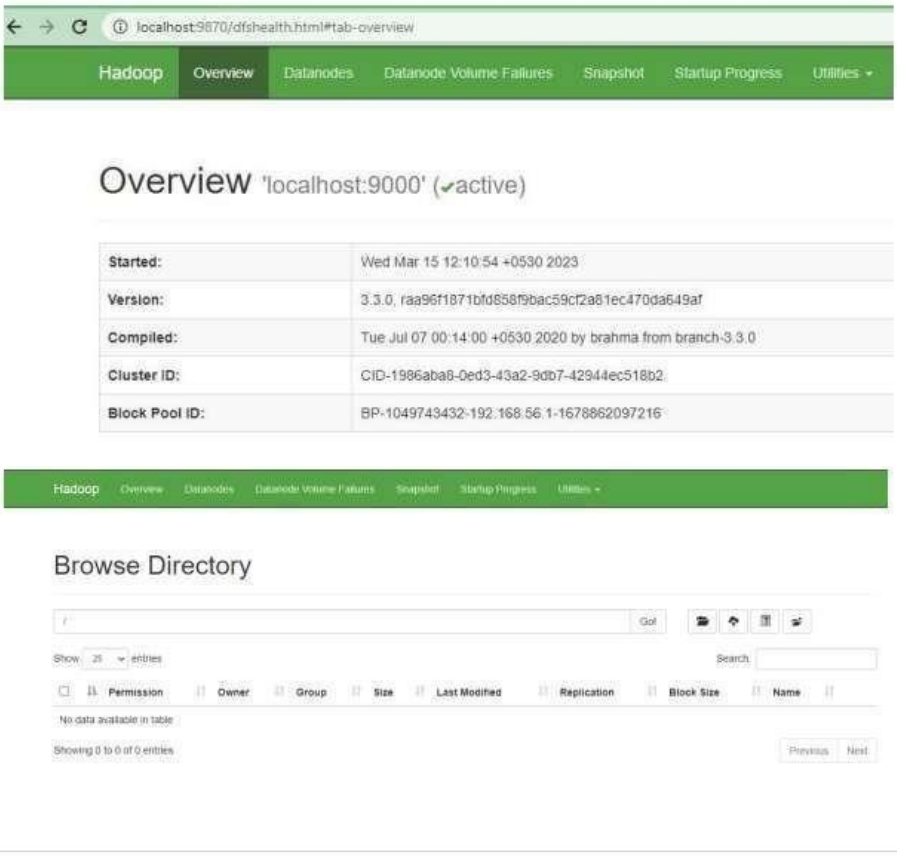


Output :



10. Type JPS command to start-all.cmd command prompt, you will get following output.

11. Run http://localhost:9870/ from any browser



PRACTICAL NO: 06

AIM: Implement word count / frequency programs using MapReduce

Steps:

- C:\hadoop-3.3.0\sbin>start-dfs.cmd
- C:\hadoop-3.3.0\sbin>start-yarn.cmd
- Open a command prompt as administrator and run the following command to create an input and output folder on the Hadoop file system, to which we will be moving the sample.txt file for our analysis.
- C:\hadoop-3.3.0\bin>cd\
- C:\>hadoop dfsadmin -safemode leave
- DEPRECATED: Use of this script to execute hdfs command is deprecated.
- Instead use the hdfs command for it.
- Safe mode is OFF
- C:\>hadoop fs -mkdir /input_dir
- Check it by giving the following URL at browser http://localhost:9870
- Now apply the following command at c:\>
- C:\> hadoop fs -put C:/input_file.txt /input_dir
- Verify input_file.txt available in HDFS input directory (input_dir).
  C:\>Hadoop fs -ls /input_dir/
- C:\>hadoop jar C:/hadoop-3.3.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.0.jar wordcount /input_dir /output_dir
- In case, there is some error in executing then copy the file MapReduceClient.jar in C:\ and run the program with the jar file using existing MapReduceClient.jar file as:
- C:\> hadoop jar C:/MapReduceClient.jar wordcount /input_dir /output_dir
- Now, check the output_dir on browser as follows:
- Click on output_dir à part-r-00000 à Head the file (first 32 K) and check the file content as the output.
- Alternatively, you may type the following command on CMD window as:
- C:\> hadoop dfs -cat /output_dir/*

You can get the following output

Output:

```
hadoop dfs -cat /output_dir/*
Hadoop  1
Window  1
version  1
is  1
easy  1
compared  1
to  1
Ubuntu  1
```

PRACTICAL NO: 07

AIM: Implement Decision tree classification techniques

A **Decision Tree** is one of the most intuitive classification techniques because it mimics human decision-making. We will use the `rpart` package to build the tree and `rpart.plot` to visualize it.

For this example, we will use the built-in `iris` dataset, which classifies flowers based on their physical measurements.

Step 1: Install and Load Required Packages

We need `rpart` for the logic and `rpart.plot` for a clean, easy-to-read visualization.

How to execute: Copy these lines into your R script and run them.

R

```
# Install relevant packages
install.packages("rpart")
install.packages("rpart.plot")

# Load the libraries
library(rpart)
library(rpart.plot)
```

Step 2: Prepare and Split the Data

We split the data to ensure our model can generalize to new, unseen information.

R

```
# Load data
data(iris)

# Set seed for reproducibility
set.seed(42)

# Split data: 70% Training, 30% Testing
sample_index <- sample(1:nrow(iris), 0.7 * nrow(iris))
train_data <- iris[sample_index, ]
test_data  <- iris[-sample_index, ]
```

Step 3: Build the Decision Tree Classifier

We use the `rpart()` function. The formula `Species ~ .` tells R to predict the "Species" using all other available variables (Sepal Length, Width, etc.).

R

 # Train the Decision Tree model
dt_model <- rpart(Species ~ ., data = train_data, method = "class")

 # View a text summary of the tree
print(dt_model)

Step 4: Visualize the Tree

The best part of a Decision Tree is the visualization. This makes it "Explainable AI."

R
 # Plot the tree
rpart.plot(dt_model, type = 4, extra = 101, under = TRUE, cex = 0.8, box.palette = "GnBu")

Step 5: Evaluate the Performance

Finally, we test the model on the 30% of data we held back to see its accuracy.

R

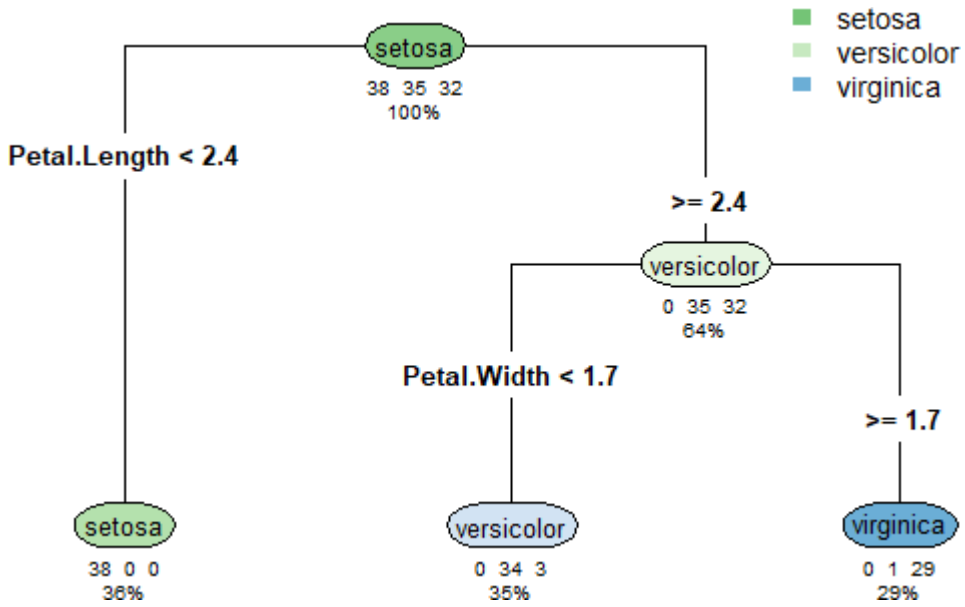
 # Make predictions on the test set
dt_predictions <- predict(dt_model, test_data, type = "class")

 # Create a Confusion Matrix
conf_matrix <- table(test_data$Species, dt_predictions)
print(conf_matrix)

 # Calculate Accuracy
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
print(paste("Accuracy:", round(accuracy, 4)))

OUTPUT:-

```
print(conf_matrix)
            dt_predictions
             setosa versicolor virginica
  setosa         12          0         0
  versicolor      0         14         1
  virginica       0          1        17
```




```
print(paste("Accuracy:", round(accuracy, 4)))
[1] "Accuracy: 0.9556"
```

PRACTICAL NO: 08

AIM: Implement SVM classification techniques

Support Vector Machines (SVM) are powerful supervised learning models used for classification. SVM works by finding the **hyperplane** (a boundary) that best separates different classes with the maximum possible margin.

To implement SVM in R, the most popular and reliable package is `e1071`.

---

1. Required Packages

In R, SVM is commonly implemented using the **`e1071`** package.

```
install.packages("e1071")    # run once
library(e1071)
```

---

2. Load Dataset (Iris Dataset)

```
data(iris)
head(iris)
```

- **Features:** Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
- **Target/Class:** Species

---

3. Split Dataset into Training & Testing

```
set.seed(123)

index <- sample(1:nrow(iris), 0.7 * nrow(iris))
train_data <- iris[index, ]
test_data  <- iris[-index, ]
```

---

4. Linear SVM Classification

```
svm_linear <- svm(
  Species ~ .,
  data = train_data,
  kernel = "linear"
)

pred_linear <- predict(svm_linear, test_data)

table(Predicted = pred_linear, Actual = test_data$Species)
mean(pred_linear == test_data$Species)
```

Confusion Matrix

```
library(caret)

confusionMatrix(pred-rbf, test_data$Species)
```

OUTPUT:

```
pred_linear <- predict(svm_linear, test_data)
> table(Predicted = pred_linear, Actual = test_data$Species)
            Actual
Predicted    setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         17         0
  virginica       0          1        13
> mean(pred_linear == test_data$Species)
[1] 0.9777778
```

- **Rows = Predicted class**
- **Columns = Actual class**

Correct Predictions (Diagonal Values)

These are the **correct classifications**:

- **Setosa:** 14 correctly classified
- **Versicolor:** 17 correctly classified
- **Virginica:** 13 correctly classified

Incorrect Predictions (Off-diagonal Values)

- **1 Versicolor was misclassified as Virginica**

All other samples are classified correct

UNIVERSITY OF MUMBAI

Practical Journal of

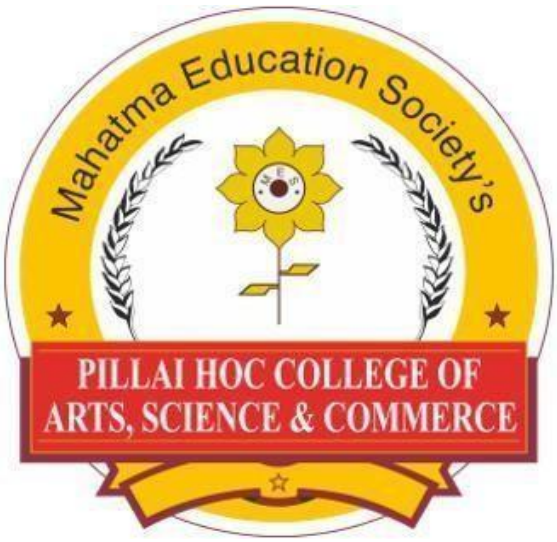
Big Data Analytics , Modern Networking & Computer Vision

M.Sc. (Information Technology) Part-I

Submitted by

PATIL PAVAN MAHESH

Seat No: 26MSCIT203010



DEPARTMENT OF INFORMATION TECHNOLOGY

PILLAI HOC COLLEGE OF ARTS,SCIENCE & COMMERCE(AUTONOMOUS),

RASAYANI, 410207 MAHARASTRA

(Affiliated to Mumbai University)

RASAYANI-410207, MAHARASHTRA

NAAC Accredited with 'A+' Grade in Cycle

2025-2026

MAHATMA EDUCATION SOCIETY'S
Pillai HOC College of Arts, Science & Commerce (Autonomous),
Rasayani 410207
(Affiliated to Mumbai University)
NAAC Accredited with 'A+' Grade in Cycle

DEPARTMENT OF

INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the experiment work entered in this journal is as per the syllabus in **M.Sc. (Information Technology) Part-I, Semester-II**; class prescribed by University of Mumbai for **Modern Networking**, of Mahatma Education Society's **Pillai HOC College of Arts, Science & Commerce(Autonomous), Rasayani** by **Patil Pavan Mahesh** during Academic year **2025-2026.**
Seat No: **26MSCIT203010**

Subject In-Charge **Co-Ordinator**

External Examiner **Principal**

Date **College Seal**

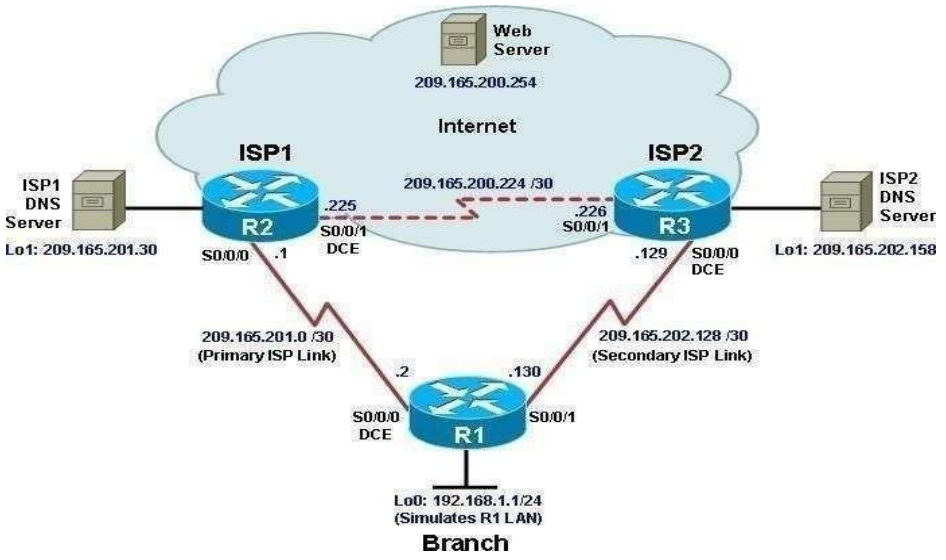
MODERN NETWORKING

INDEX

PRACTICAL NO: 01

AIM: Configure IP SLA Tracking and Path Control

Topology



Objectives

- Configure and verify the IP SLA feature.
- Test the IP SLA tracking feature.
- Verify the configuration and operation using show and debug commands.

Background

You want to experiment with the Cisco IP Service Level Agreement (SLA) feature to study how it could be of value to your organization.

At times, a link to an ISP could be operational, yet users cannot connect to any other outside Internet resources. The problem might be with the ISP or downstream from them. Although policy-based routing (PBR) can be implemented to alter path control, you will implement the Cisco IOS SLA feature to monitor this behavior and intervene by injecting another default route to a backup ISP.

To test this, you have set up a three-router topology in a lab environment. Router R1 represents a branch office connected to two different ISPs. ISP1 is the preferred connection to the Internet, while ISP2 provides a backup link. ISP1 and ISP2 can also interconnect, and both can reach the web server. To monitor ISP1 for failure, you will configure IP SLA probes to track the reachability to the ISP1 DNS server. If connectivity to the ISP1 server fails, the SLA probes detect the failure and alter the default static route to point to the ISP2 server.

Note: This lab uses Cisco 1841 routers with Cisco IOS Release 12.4(24) T1 and the Advanced IP Services image c1841-advipservicesk9-mz.124-24.T1.bin. You can use other routers (such as a 2801 or 2811) and Cisco IOS Software versions if they have comparable capabilities and features. Depending on the router and Cisco IOS Software version, the commands available and output produced might vary from what is shown in this lab failure, you will configure IP SLA probes to track the reachability to the ISP1 DNS server. If connectivity to the ISP1 server fails, the SLA probes detect the failure and alter the default static route to point to the ISP2 server.

Note: This lab uses Cisco 1841 routers with Cisco IOS Release 12.4(24)T1 and the Advanced IP Services image c1841-advipservicesk9-mz.124-24.T1.bin. You can use other routers (such as a 2801 or 2811) and Cisco IOS Software versions if they have comparable capabilities and

features. Depending on the router and Cisco IOS Software version, the commands available and output produced might vary from what is shown in this lab.

Required Resources

- 3 routers (Cisco 1841 with Cisco IOS Release 12.4(24)T1 Advanced IP Services or comparable)
- Serial and console cables

Step 1: Prepare the routers and configure the router hostname and interface addresses.

A] Cable the network as shown in the topology diagram. Erase the startup configuration and reload each router to clear the previous configurations. Using the addressing scheme in the diagram, create the loopback interfaces and apply IP addresses to them as well as the serial interfaces on R1, ISP1, and ISP2.

You can copy and paste the following configurations into your routers to begin.

Note: Depending on the router model, interfaces might be numbered differently than those listed. You might need to alter them accordingly.

Router R1

```
hostname R1
interface Loopback 0
description R1 LAN
ip address 192.168.1.1
255.255.255.0
interface Serial0/0/0
description R1 --> ISP1
ip address 209.165.201.2
255.255.255.252
clock rate 128000
bandwidth 128
no shutdown
interface Serial0/0/1
description R1 --> ISP2
ip address 209.165.202.130
255.255.255.252
bandwidth 128
no shutdown
```

Router ISP1 (R2)

```
hostname ISP1
interface Loopback0
description Simulated Internet
Web Server
ip address 209.165.200.254
255.255.255.255
interface Loopback1
description ISP1 DNS Server
ip address 209.165.201.30
255.255.255.255
interface Serial0/0/0
description ISP1 --> R1
ip address 209.165.201.1
255.255.255.252
bandwidth 128
no shutdown
```

interface Serial0/0/1
description ISP1 --> ISP2
ip address 209.165.200.225
255.255.255.252
clock rate 128000
bandwidth 128
no shutdown

Router ISP2 (R3)
hostname ISP2
interface Loopback0
description Simulated
Internet Web Server
ip address 209.165.200.254
255.255.255.255
interface Loopback1
description ISP2 DNS
Server
ip address 209.165.202.158
255.255.255.255
interface Serial0/0/0
description ISP2 --> R1
ip address 209.165.202.129
255.255.255.252
clock rate 128000
bandwidth 128
no shutdown
interface Serial0/0/1
description ISP2 --> ISP1
ip address 209.165.200.226
255.255.255.252
bandwidth 128
no shutdown

B] Verify the configuration by using the show interfaces description command. The output from router R1 is shown here as an example.

R1# show interfaces description

Interface Fa0/0 Fa0/1	Status admin down admin down	Protocol down down	Description
Se0/0/0	up	up	R1 --> ISP1
Se0/0/1	up	up	R1 --> ISP2
Lo0	up	up	R1 LAN

All three interfaces should be active. Troubleshoot if necessary.

C] The current routing policy in the topology is as follows:

- Router R1 establishes connectivity to the Internet through ISP1 using a default static route.
- ISP1 and ISP2 have dynamic routing enabled between them, advertising their respective public address pools.
- ISP1 and ISP2 both have static routes back to the ISP LAN.

Note: For the purpose of this lab, the ISPs have a static route to an RFC 1918 private network address on the branch router R1. In an actual branch implementation, Network Address Translation (NAT) would be configured for all traffic exiting the branch LAN. Therefore, the static routes on the ISP routers would be pointing to the provided public pool of the branch

office. This is covered in Lab 7-1, ―Configure Routing Facilities to the Branch Office. Implement the routing policies on the respective routers. You can copy and paste the following configurations.

Router R1

ip route 0.0.0.0 0.0.0.0 209.165.201.1

Router ISP1 (R2)

router eigrp 1
network 209.165.200.224 0.0.0.3
network 209.165.201.0 0.0.0.31
no auto-summary
ip route 192.168.1.0 255.255.255.0 209.165.201.2

Router ISP2 (R3)

router eigrp 1
network 209.165.200.224 0.0.0.3
network 209.165.202.128 0.0.0.31
no auto-summary
ip route 192.168.1.0 255.255.255.0 209.165.202.130

EIGRP neighbor relationship messages on ISP1 and ISP2 should be generated. Troubleshoot if necessary.

%DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 209.165.200.225 (Serial0/0/1) is up: new adjacency.

Step 2: Verify server reachability.

D] The current routing policy in the topology is as follows:

- Router R1 establishes connectivity to the Internet through ISP1 using a default static route.
- ISP1 and ISP2 have dynamic routing enabled between them, advertising their respective public address pools.
- ISP1 and ISP2 both have static routes back to the ISP LAN.

Note: For the purpose of this lab, the ISPs have a static route to an RFC 1918 private network address on the branch router R1. In an actual branch implementation, Network Address Translation (NAT) would be configured for all traffic exiting the branch LAN. Therefore, the static routes on the ISP routers would be pointing to the provided public pool of the branch office. This is covered in Lab 7-1, ―Configure Routing Facilities to the Branch Office. Implement the routing policies on the respective routers. You can copy and paste the following configurations.

Router R1

ip route 0.0.0.0 0.0.0.0 209.165.201.1

Router ISP1 (R2)

router eigrp 1
network 209.165.200.224 0.0.0.3
network 209.165.201.0 0.0.0.31
no auto-summary
ip route 192.168.1.0 255.255.255.0 209.165.201.2

Router ISP2 (R3)

router eigrp 1
network 209.165.200.224 0.0.0.3
network 209.165.202.128 0.0.0.31
no auto-summary

ip route 192.168.1.0 255.255.255.0 209.165.202.130

EIGRP neighbor relationship messages on ISP1 and ISP2 should be generated. Troubleshoot if necessary.

%DUAL-5-NBRCHANGE: IP-EIGRP(0) 1: Neighbor 209.165.200.225 (Serial0/0/1) is up: new adjacency

Step 2: Verify server reachability.

The Cisco IOS IP SLA feature enables an administrator to monitor network performance between Cisco devices (switches or routers) or from a Cisco device to a remote IP device. IP SLA probes continuously check the reachability of a specific destination, such as a provider edge router interface, the DNS server of the ISP, or any other specific destination, and can conditionally announce a default route only if the connectivity is verified.

a. Before implementing the Cisco IOS SLA feature, you must verify reachability to the Internet servers. From router R1, ping the web server, ISP1 DNS server, and ISP2 DNS server to verify connectivity. You can copy the following Tcl script and paste it intoR1.

```
foreach address {
209.165.200.254
209.165.201.30
209.165.202.158
} {
ping $address source 192.168.1.1
}
R1(tcl)# foreach address {
+>(tcl)# 209.165.200.254.
+>(tcl)# 209.165.201.30
+>(tcl)# 209.165.202.158
+>(tcl)# } {
+>(tcl)# ping $address source 192.168.1.1
+>(tcl)#}
Type escape sequence to abort.
Sending 5, 100
-
byte ICMP Echos to 209.165.200.254, timeout is 2 seconds: Packet sent with
a source address of 192.168.1.1
!!!!!
Success rate is 100 percent (5/5), round
-
trip min/avg/max = 12/15/16 ms Type escape
sequence to abort.
Sending 5, 100
-
byte ICMP Echos to 209.165.201.30, timeout is 2 seconds: Packet sent with a
source address of 192.168.1.1
!!!!!
Success rate is 100 percent (5/5), round
-
trip min/avg/max = 12/14/16 ms Type escape
```

sequence to abort.
Sending 5, 100
-
byte ICMP Echos to 209.165.202.158, timeout is 2 seconds: Packet sent with
a source address of 192.168.1.1
!!!!!
Success rate is 100 percent (5/5), round
-
trip min/avg/max = 20/21/24 ms
B] Trace the path taken to the web server, ISP1 DNS server, and ISP2 DNS server. You can copy the following Tcl script and paste it into R1.
foreach address { 209.165.200.254
209.165.201.30
209.165.202.158 } {
trace $address source 192.168.1.1 }
R1(tcl)# foreach address {
+>(tcl)# 209.165.200.254
+>(tcl)#
209.165.201.30
+>(tcl)# 209.165.202.158
+>(tcl)# } {
+>(tcl)# trace $address source 192.168.1.1
+>(tcl)# }

Step 3: Configure IP SLA probes.
When the reachability tests are successful, you can configure the Cisco IOS IP SLAs probes. Different types of probes can be created, including FTP, HTTP, and jitter probes. In this scenario, you will configure ICMP echo probes.
A] Create an ICMP echo probe on R1 to the primary DNS server on ISP1 using the ip sla command.
Note: With Cisco IOS Release 12.4(4)T, 12.2(33)SB, and 12.2(33)SXI, the ip sla command has replaced the previous ip sla monitor command. In addition, the icmp-echo command has replaced the type echo protocol ipIcmpEcho command.
R1(config)# ip sla 11
R1(config-ip-sla)# icmp-echo 209.165.201.30
R1(config-ip-sla-echo)# frequency 10
R1(config-ip-sla-echo)# exit
R1(config)# ip sla schedule 11 life forever start-time now
The operation number of 11 is only locally significant to the router. The frequency 10 command schedules the connectivity test to repeat every 10 seconds. The probe is scheduled to start now and to run forever.

B] Verify the IP SLAs configuration of operation 11 using the show ip sla configuration 11 command.
Note: With Cisco IOS Release 12.4(4)T, 12.2(33)SB, and 12.2(33)SXI, the show ip sla configuration
command has replaced the show ip sla monitor configuration command.
R1# show ip sla configuration 11 IP SLAs, Infrastructure Engine-II. Entry number: 11
Owner:
Tag:
Type of operation to perform: icmp-echo

Target address/Source address:
209.165.201.30
/0.0.0.0 Type Of Service parameter: 0x0
Request size (ARR data portion): 28 Operation timeout (milliseconds): 5000 Verify data: No
Vrf Name:
Schedule:
Operation frequency (seconds): 10 (not considered if randomly scheduled) Next Scheduled Start Time: Start Time already passed
Group Scheduled : FALSE Randomly Scheduled : FALSE Life (seconds): Forever
Entry Ageout (seconds): never Recurring (Starting Everyday): FALSE
Status of entry (SNMP RowStatus): Active
Threshold (milliseconds): 5000 (not considered if react RTT is configured) Distribution Statistics:
Number of statistic hours kept: 2
Number of statistic distribution buckets kept: 1
Statistic distribution interval (milliseconds): 20
History Statistics:
Number of history Lives kept: 0 Number of history Buckets kept: 15 History Filter Type: None
Enhanced History:
The output lists the details of the configuration of operation 11. The operation is an ICMP echo to 209.165.201.30, with a frequency of 10 seconds, and it has already started (the start time has already passed).

C] Issue the show ip sla statistics command to display the number of successes, failures, and results of the latest operations.

Note:
With Cisco IOS Release 12.4(4)T, 12.2(33)SB, and 12.2(33)SXI, the show ip sla statistics
command has replaced the show ip sla monitor statistics command.
R1# show ip sla statistics
IPSLAs Latest Operation Statistics
IPSLA operation id: 11
Latest operation start time: *21:22:29.707 UTC Fri Apr 2 2010 Latest operation return code: OK

D] Although not actually required because IP SLA session 11 alone could provide the desired fault tolerance, create a second probe, 22, to test connectivity to the second DNS server located on router ISP2. You can copy and paste the following commands on R1
ip sla 22
icmp-echo 209.165.202.158
frequency 10 exit
ip sla schedule 22 life forever start-time now

E] Verify the new probe using the show ip sla configuration and show ip sla statistics commands.
R1# show ip sla configuration 22 IP SLAs, Infrastructure Engine II. Entry number: 22

Step 4: Configure tracking options.
Although PBR could be used, you will configure a floating static route that appears or disappears depending on the success or failure of the IP SLA.

A] Remove the current default route on R1, and replace it with a floating static route having an administrative distance of 5.
R1(config)# no ip route 0.0.0.0 0.0.0.0 209.165.201.1

R1(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1 5
R1(config)# exit

B] Verify the routing table.

R1# show ip route
*Apr 2 20:00:37.367: %SYS-5-CONFIG_I: Configured from console by console Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1,
L2 - IS-IS level-2 ia - IS-IS inter area, * - candidate default, U - per-user static
Route
o - ODR, P - periodic downloaded static route
Gateway of last resort is 209.165.201.1 to network 0.0.0.0
209.165.201.0/30 is subnetted, 1 subnets
C 209.165.201.0 is directly connected, Serial0/0/0 209.165.202.0/30 is subnetted, 1 subnets
C 209.165.202.128 is directly connected, Serial0/0/1C 192.168.1.0/24 is directly connected, FastEthernet0/0 S* 0.0.0.0/0 [5/0] via 209.165.201.1
Notice that the default static route is now using the route with the administrative distance of 5. The first tracking object is tied to IP SLA object 11.

C] Use the track 1 ip sla 11 reachability command to enter the config-track subconfiguration mode.

Note: With Cisco IOS Release 12.4(20)T, 12.2(33)SXI1, and 12.2(33)SRE and Cisco IOS XE Release 2.4, the track ip sla command has replaced the track rtr command.

R1(config)# track 1 ip sla 11 reachability
R1(config-track)#

D] Specify the level of sensitivity to changes of tracked objects to 10 seconds of down delay and 1 second of up delay using the delay down 10 up 1 command. The delay helps to alleviate the effect of flapping objects—objects that are going down and up rapidly. In this situation, if the DNS server fails momentarily and comes back up within 10 seconds, there is no impact.

R1(config-track)# delay down 10 up 1

R1(config-track)# exit
R1(config)#

E] Configure the floating static route that will be implemented when tracking object 1 is active. To view routing table changes as they happen, first enable the debug ip routing command. Next, use the ip route 0.0.0.0 0.0.0.0 209.165.201.1 2 track 1 command to create a floating static default route via 209.165.201.1 (ISP1). Notice that this command references the tracking object number 1, which in turn references IP SLA operation number 11.

R1# debug ip routing
IP routing debugging is on R1#
*Apr 2 21:26:46.171: RT: NET-RED 0.0.0.0/0
R1# conf t
Enter configuration commands, one per line. End with CNTL/Z. R1(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1 2 track 1 R1(config)#
*Apr 2 21:27:02.851: RT: closer admin distance for 0.0.0.0, flushing 1
routes
*Apr 2 21:27:02.851: RT: NET-RED 0.0.0.0/0
*Apr 2 21:27:02.851: RT: add 0.0.0.0/0 via 209.165.201.1, static metric

[2/0]
*Apr 2 21:27:02.851: RT: NET-RED 0.0.0.0/0
*Apr 2 21:27:02.851: RT: default path is now 0.0.0.0 via 209.165.201.1
*Apr 2 21:27:02.855: RT: new default network 0.0.0.0
*Apr 2 21:27:02.855: RT: NET-RED 0.0.0.0/0
*Apr 2 21:27:07.851: RT: NET-RED 0.0.0.0/0
Notice that the default route with an administrative distance of 5 has been immediately flushed because of a route with a better admin distance. It then adds the new default route with the admin distance of 2.

F] Repeat the steps for operation 22, track number 2, and assign the static route an admin distance higher than track 1 and lower than 5. On R1, copy the following configuration, which sets an admin distance of 3.

track 2 ip sla 22 reachability
delay down 10 up 1 exit
ip route 0.0.0.0 0.0.0.0 209.165.202.129 3 track 2

G] Verify the routing table again.

R1# show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2i - IS-IS,
su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2ia –
IS-IS inter area, * - candidate default, U - per-user stati
o - ODR, P - periodic downloaded static route Gateway of last resort is 209.165.201.1 to network 0.0.0.0
209.165.201.0/30 is subnetted, 1 subnets
C 209.165.201.0 is directly connected, Serial0/0/0 209.165.202.0/30 is subnetted, 1 subnets
C 209.165.202.128 is directly connected, Serial0/0/1C192.168.1.0/24 is directly connected, FastEthernet0/0 S* 0.0.0.0/0 [2/0] via 209.165.201.1
Although a new default route was entered, its administrative distance is not better than 2. Therefore, itdoes not replace the previously entered default route.

Step 5: Verify IP SLA operation.

In this step you observe and verify the dynamic operations and routing changes when tracked objects fail.The following summarizes the process:

- Disable the DNS loopback interface on ISP1 (R2).
- Observe the output of the debug command on R1.
- Verify the static route entries in the routing table and the IP SLA statistics of R1.
- Re-enable the loopback interface on ISP1 (R2) and again observe the operation of the IP SLAtracking feature.

ISP1(config)# interface loopback 1
ISP1(config-if)# shutdown ISP1(config-if)#
*Apr 2 15:53:14.307: %LINK-5-CHANGED: Interface Loopback1, changed state to administratively down
*Apr 2 15:53:15.307: %LINEPROTO-5-UPDOWN: Line protocol on InterfaceLoopback1, changed state to down

A] Shortly after the loopback interface is administratively down, observe the debug output being generatedon R1.

R1#
*Apr 2 21:32:33.323: %TRACKING-5-STATE: 1 ip sla 11 reachability Up->Down
*Apr 2 21:32:33.323: RT: del 0.0.0.0 via 209.165.201.1, static metric [2/0]

```
*Apr  2 21:32:33.323:    RT:  delete network route to 0.0.0.0
*Apr  2 21:32:33.323:    RT:  NET-RED 0.0.0.0/0
*Apr  2 21:32:33.323:    RT:  NET-RED 0.0.0.0/0
*Apr  2 21:32:33.323:    RT:  add 0.0.0.0/0 via 209.165.202.129, static metric
[3/0]
*Apr  2 21:32:33.323:    RT:  NET-RED 0.0.0.0/0
*Apr  2 21:32:33.323:    RT:  default path is now 0.0.0.0 via 209.165.202.129
*Apr  2 21:32:33.323:    RT:  new default network 0.0.0.0
*Apr  2 21:32:33.327:    RT:  NET-RED 0.0.0.0/0
*Apr  2 21:32:46.171:    RT:  NET-RED 0.0.0.0/0
```

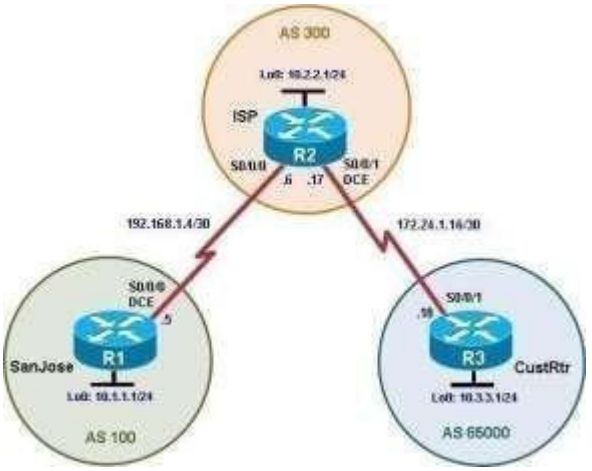
route

```
*Apr 2 21:35:34.327: %TRACKING-5-STATE: 1 ip sla 11 reachability Down->Up
*Apr 2 21:35:34.327: RT: closer admin distance for 0.0.0.0, flushing 1 routes
*Apr 2 21:35:34.327: RT: NET-RED 0.0.0.0/0
*Apr 2 21:35:34.327: RT: add 0.0.0.0/0 via 209.165.201.1, static metric [2/0]
*Apr 2 21:35:34.327: RT: NET-RED 0.0.0.0/0
*Apr 2 21:35:34.327: RT: default path is now 0.0.0.0 via 209.165.201.1
*Apr 2 21:35:34.327: RT: new default network 0.0.0.0
*Apr 2 21:35:34.327: RT: NET-RED 0.0.0.0/0
*Apr 2 21:35:39.327: RT: NET-RED 0.0.0.0/0
*Apr 2 21:35:46.171: RT: NET-RED 0.0.0.0/0
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2ia - IS
IS inter area, * - candidate default, U - per- user static
o - ODR, P - periodic downloaded static route
Gateway of last resort is 209.165.201.1 to network 0.0.0.0
209.165.201.0/30 is subnetted, 1 subnets
C 209.165.201.0 is directly connected, Serial0/0/0
209.165.202.0/30 is subnetted, 1 subnets
C       209.165.202.128 is directly connected, Serial0/0/1C
192.168.1.0/24 is directly connected, FastEthernet0/0
S* 0.0.0.0/0 [2/0] via 209.165.201.
```

PRACTICAL NO: 02

AIM: Using the AS_PATH Attribute

Topology



Objectives

Use BGP commands to prevent private AS numbers from being advertised to the outside world.

Use the AS_PATH attribute to filter BGP routes based on their source AS numbers.

Background

The International Travel Agency's ISP has been assigned an AS number of 300. This provider uses BGP to exchange routing information with several customer networks. Each customer network is assigned an AS number from the private range, such as AS 65000. Configure the ISP router to remove the private AS numbers from the AS Path information of CustRtr.

Required Resources

3 routers (Cisco 1841 with Cisco IOS Release 12.4(24)T1 Advanced IP Services or comparable)

Serial and console cables

Step 1: Prepare the routers for the lab.

Cable the network as shown in the topology diagram. Erase the startup configuration and reload each router to clear previous configurations.

Step 2: Configure the hostname and interface addresses.

A] You can copy and paste the following configurations into your routers to begin.

Router R1 (hostname SanJose)

```
hostname SanJose
!
interface Loopback0
ip address 10.1.1.1 255.255.255.0
!
interface Serial0/0/0
ip address 192.168.1.5 255.255.255.252
clock rate 128000 no shutdown
```

Router R2 (hostname ISP)

```
hostname ISP
!
interface Loopback0
ip address 10.2.2.1 255.255.255.0
!
```

interface Serial0/0/0
ip address 192.168.1.6 255.255.255.252
no shutdown
!
interface Serial0/0/1
ip address 172.24.1.17 255.255.255.252
clock rate 128000 no shutdown

Router R3 (hostname CustRtr)

hostname CustRtr
!
interface Loopback0
ip address 10.3.3.1 255.255.255.0
!
interface Serial0/0/1
ip address 172.24.1.18 255.255.255.252
no shutdown

B] Use ping to test the connectivity between the directly connected routers.

Note: SanJose will not be able to reach either ISP's loopback (10.2.2.1) or CustRtr's loopback (10.3.3.1), nor will it be able to reach either end of the link joining ISP to CustRtr (172.24.1.17 and 172.24.1.18).

Step 3: Configure BGP.

A] Configure BGP for normal operation. Enter the appropriate BGP commands on each router so that they identify their BGP neighbors and advertise their loopback networks.

SanJose(config)# router bgp 100
SanJose(config-router)# neighbor 192.168.1.6 remote-as 300
SanJose(config-router)# network 10.1.1.0 mask 255.255.255.0
ISP(config)# router bgp 300
ISP(config-router)# neighbor 192.168.1.5 remote-as 100
ISP(config-router)# neighbor 172.24.1.18 remote-as 65000
ISP(config-router)# network 10.2.2.0 mask 255.255.255.0
CustRtr(config)# router bgp 65000
CustRtr(config-router)# neighbor 172.24.1.17 remote-as 300
CustRtr(config-router)# network 10.3.3.0 mask 255.255.255.0

B] Verify that these routers have established the appropriate neighbor relationships by issuing the show ipbgp neighbors command on each router.

ISP# show ip bgp neighbors
BGP neighbor is 172.24.1.18, remote AS 65000, external link BGP version 4, remote router ID 10.3.3.1
BGP state = Established, up for 00:02:05
<output omitted>
BGP neighbor is 192.168.1.5, remote AS 100, external link BGP version 4, remote router ID 10.1.1.1
BGP state = Established, up for 00:04:19
<output omitted>

Step 4: Remove the private AS.

A] Display the SanJose routing table using the show ip route command. SanJose should have a route to both 10.2.2.0 and 10.3.3.0. Troubleshoot if necessary.

SanJose# show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

B] Ping the 10.3.3.1 address from SanJose.Why does this fail?
C] Ping again, this time as an extended ping, sourcing from the Loopback0 interface address.
SanJose# ping
Protocol [ip]:
Target IP address: 10.3.3.1
Repeat count [5]:
Datagram size [100]: Timeout in seconds [2]: Extended commands [n]: y
Source address or interface: 10.1.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/64/68 ms
Note: You can bypass extended ping mode and specify a source address using one of these commands:
SanJose# ping 10.3.3.1 source 10.1.1.1
or
SanJose# ping 10.3.3.1 source Lo0
D] Check the BGP table from SanJose by using the show ip bgp command. Note the AS path for the 10.3.3.0 network. The AS 65000 should be listed in the path to 10.3.3.0.
SanJose# show ip bgp
BGP table version is 5, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i – internal Origin codes: i - IGP, e - EGP, ? - incomplete

Network	Next Hop	Metric LocPrf Weight Path
*> 10.1.1.0	0.0.0.0	32768 i
*> 10.2.2.0	192.168.1.6	0 300 i
*> 10.3.3.0	192.168.1.6	0 300 65000 i

E] Configure ISP to strip the private AS numbers from BGP routes exchanged with SanJose using the following commands.
ISP(config)# router bgp 300
ISP(config-router)# neighbor 192.168.1.5 remove-private-as
F] After issuing these commands, use the clear ip bgp * command on ISP to reestablish the BGP relationship between the three routers. Wait several seconds and then return to SanJose to check its routing table.
Note: The clear ip bgp * soft command can also be used to force each router to resend its BGP table. Does SanJose still have a route to 10.3.3.0?
SanJose should be able to ping 10.3.3.1 using its loopback 0 interface as the source of the ping.
SanJose# ping 10.3.3.1 source lo0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.3.1, timeout is 2 seconds:
Packet sent with a source address of 10.1.1.1 !!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
Now check the BGP table on SanJose. The AS_ PATH to the 10.3.3.0 network should be AS 300. It no longer has the private AS in the path.

Network	Next Hop	Metric LocPrf Weight Path
*> 10.1.1.0	0.0.0.0	32768 i
*> 10.2.2.0	192.168.1.6	0 300 i
*> 10.3.3.0	192.168.1.6	0 300 i

Step 5: Use the AS_PATH attribute to filter routes.

As a final configuration, use the AS_PATH attribute to filter routes based on their origin. In a complex environment, you can use this attribute to enforce routing policy. In this case, the provider router, ISP, must be configured so that it does not propagate routes that originate from AS 100 to the customer router CustRtr.

AS-path access lists are read like regular access lists. The statements are read sequentially, and there is an implicit deny at the end. Rather than matching an address in each statement like a conventional access list, AS path access lists match on something called a regular expression. Regular expressions are a way of matching text patterns and have many uses. In this case, you will be using them in the AS path access list to match text patterns in AS paths.

A] Configure a special kind of access list to match BGP routes with an AS_PATH attribute that both begins and ends with the number 100. Enter the following commands on ISP.

For more details on configuring regular expressions on Cisco routers, see:
http://www.cisco.com/en/US/docs/ios/12_2/termserv/configuration/guide/tcfaapre_ps1835_TSD_Products
_Configuration_Guide_Chapter.html

B] Apply the configured access list using the neighbor command with the filter-list option.

ISP(config)# router bgp 300
ISP(config-router)# neighbor 172.24.1.18 filter-list 1 out

The out keyword specifies that the list is applied to routing information sent to this neighbor. Use the clear ip bgp * command to reset the routing information. Wait several seconds and then check the routing table for ISP. The route to 10.1.1.0 should be in the routing table.

Note: To force the local router to resend its BGP table, a less disruptive option is to use the clear ip bgp

C] Run the following Tcl script on all routers to verify whether there is connectivity. All pings from ISP should be successful. SanJose should not be able to ping the CustRtr loopback 10.3.3.1 or the WAN link

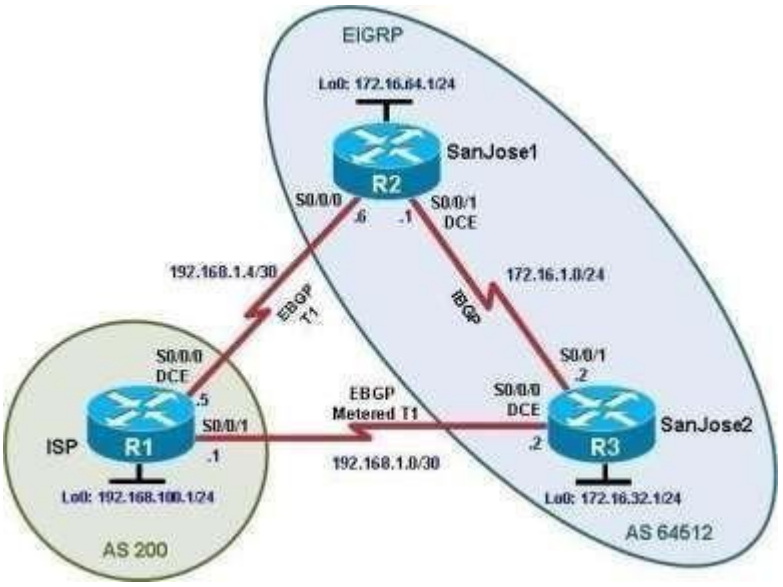
172.24.1.16/30. CustRtr should not be able to ping the SanJose loopback 10.1.1.1 or the WAN link 192.168.1.4/30.

```
ISP# tclsh
foreach address { 10.1.1.1
10.2.2.1
10.3.3.1
192.168.1.5
192.168.1.6
172.24.1.17
172.24.1.18
} {
ping $address }
```

PRACTICAL NO: 03

AIM: Configuring IBGP and EBGP Sessions, Local Preference, and MED

Topology



Objectives

- For IBGP peers to correctly exchange routing information, use the next-hop-self command with the Local- Preference and MED attributes.
- Ensure that the flat-rate, unlimited-use T1 link is used for sending and receiving data to and from the AS 200 on ISP and that the metered T1 only be used in the event that the primary T1 link has failed.

Background

The International Travel Agency runs BGP on its SanJose1 and SanJose2 routers externally with the ISP router in AS 200. IBGP is run internally between SanJose1 and SanJose2. Your job is to configure both EBGP and IBGP for this internetwork to allow for redundancy. The metered T1 should only be used in the event that the primary T1 link has failed. Traffic sent across the metered T1 link offers the same bandwidth of the primary link but at a huge expense. Ensure that this link is not used unnecessarily.

Note: This lab uses Cisco 1941 routers with Cisco IOS Release 15.4 with IP Base. The switches are Cisco WS-C2960-24TT-L with Fast Ethernet interfaces, therefore the router will use routing metrics associated with a 100 Mb/s interface. Depending on the router or switch model and Cisco IOS Software version, the commands available and output produced might vary from what is shown in this lab.

Required Resources

- 3 routers (Cisco IOS Release 15.2 or comparable)
- Serial and Ethernet cables

Step 0: Suggested starting configurations.

A] Apply the following configuration to each router along with the appropriate hostname. The exec-timeout 0 0 command should only be used in a lab environment.

Router(config)# no ip domain-lookup

Router(config)# line con 0 Router(config-line)# logging synchronous Router(config-line)# exec-timeout 0 0

Step 1: Configure interface addresses.

A] Using the addressing scheme in the diagram, create the loopback interfaces and apply IPv4 addresses to these and the serial interfaces on ISP (R1), SanJose1 (R2), and SanJose2 (R3).

Router R1 (hostname ISP)

ISP(config)# interface Loopback0
ISP(config-if)# ip address 192.168.100.1 255.255.255.0
ISP(config-if)# exit
ISP(config)# interface Serial0/0/0
ISP(config-if)# ip address 192.168.1.5 255.255.255.252
ISP(config-if)# clock rate 128000 ISP(config-if)# no shutdown ISP(config-if)# exit
ISP(config)# interface Serial0/0/1
ISP(config-if)# ip address 192.168.1.1 255.255.255.252
ISP(config-if)# no shutdown
ISP(config-if)# end
ISP#

Router R2 (hostname SanJose1)

SanJose1(config)# interface Loopback0
SanJose1(config-if)# ip address 172.16.64.1 255.255.255.0
SanJose1(config-if)# exit
SanJose1(config)# interface Serial0/0/0
SanJose1(config-if)# ip address 192.168.1.6 255.255.255.252
SanJose1(config-if)# no shutdown SanJose1(config-if)# exit SanJose1(config)# interface Serial0/0/1
SanJose1(config-if)# ip address 172.16.1.1 255.255.255.0
SanJose1(config-if)# clock rate 128000 SanJose1(config-if)# no shutdown SanJose1(config-if)# end
SanJose1#

Router R3 (hostname SanJose2)

SanJose2(config)# interface Loopback0
SanJose2(config-if)# ip address 172.16.32.1 255.255.255.0
SanJose2(config-if)# exit
SanJose2(config)# interface Serial0/0/0
SanJose2(config-if)# ip address 192.168.1.2 255.255.255.252
SanJose2(config-if)# clock rate 128000 SanJose2(config-if)# no shutdown SanJose2(config-if)# exit SanJose2(config)# interface Serial0/0/1
SanJose2(config-if)# ip address 172.16.1.2 255.255.255.0
SanJose2(config-if)# no shutdown
SanJose2(config-if)# end
SanJose2#

B] Use ping to test the connectivity between the directly connected routers. Both SanJose routers should be able to ping each other and their local ISP serial link IP address. The ISP router cannot reach the segment between SanJose1 and SanJose2.

Step 2: Configure EIGRP.

Configure EIGRP between the SanJose1 and SanJose2 routers. (Note: If using an IOS prior to 15.0, use the no auto-summary router configuration command to disable automatic summarization. This command is the default beginning with IOS 15.)

SanJose1(config)# router eigrp 1
SanJose1(config-router)# network 172.16.0.0
SanJose2(config)# router eigrp 1
SanJose2(config-router)# network 172.16.0.0

Step 3: Configure IBGP and verify BGP neighbors.

A] Configure IBGP between the SanJose1 and SanJose2 routers. On the SanJose1 router, enter the following configuration.

SanJose1(config)# router bgp 64512
SanJose1(config-router)# neighbor 172.16.32.1 remote-as 64512
SanJose1(config-router)# neighbor 172.16.32.1 update-source lo0

If multiple pathways to the BGP neighbor exist, the router can use multiple IP interfaces to communicate with the neighbor. The source IP address therefore depends on the outgoing interface. The update-source lo0 command instructs the router to use the IP address of the interface Loopback0 as the source IP address for all BGP messages sent to that neighbor.

B] Complete the IBGP configuration on SanJose2 using the following commands.

SanJose2(config)# router bgp 64512
SanJose2(config-router)# neighbor 172.16.64.1 remote-as 64512
SanJose2(config-router)# neighbor 172.16.64.1 update-source lo0

C] Verify that SanJose1 and SanJose2 become BGP neighbors by issuing the show ip bgp neighbors command on SanJose1. View the following partial output. If the BGP state is not established, troubleshoot the connection.

SanJose2# show ip bgp neighbors
BGP neighbor is 172.16.64.1, remote AS 64512, internal link BGP version 4, remote router ID 172.16.64.1

Step 4: Configure EBGP and verify BGP neighbors.

A] Configure ISP to run EBGP with SanJose1 and SanJose2. Enter the following commands on ISP.

ISP(config)# router bgp 200
ISP(config-router)# neighbor 192.168.1.6 remote-as 64512
ISP(config-router)# neighbor 192.168.1.2 remote-as 64512
ISP(config-router)# network 192.168.100.0

B] Configure a discard static route for the 172.16.0.0/16 network. Any packets that do not have a more specific match (longer match) for a 172.16.0.0 subnet will be dropped instead of sent to the ISP. Later in this lab we will configure a default route to the ISP.

SanJose1(config)# ip route 172.16.0.0 255.255.0.0 null0

C] Configure SanJose1 as an EBGP peer to ISP.

SanJose1(config)# router bgp 64512
SanJose1(config-router)# neighbor 192.168.1.5 remote-as 200
SanJose1(config-router)# network 172.16.0.0

D] Use the show ip bgp neighbors command to verify that SanJose1 and ISP have reached the established state. Troubleshoot if necessary.

SanJose1# show ip bgp neighbors
BGP neighbor is 172.16.32.1, remote AS 64512, internal link BGP version 4, remote router ID 172.16.32.1
BGP state = Established, up for 00:12:43
<output omitted>
BGP neighbor is 192.168.1.5, remote AS 200, external link BGP version 4, remote router ID 192.168.100.1
BGP state = Established, up for 00:06:49
.
*Sep 8 21:09:59.699: %BGP-5-ADJCHANGE: neighbor 192.168.1.5 Up

E] Configure a discard static route for 172.16.0.0/16 on SanJose2 and as an EBGP peer to ISP.

SanJose2(config)# ip route 172.16.0.0 255.255.0.0 null0

SanJose2(config)# router bgp 64512
SanJose2(config-router)# neighbor 192.168.1.1 remote-as 200
SanJose2(config-router)# network 172.16.0.0

Step 5: View BGP summary output.

In Step 4, the show ip bgp neighbors command was used to verify that SanJose1 and ISP had reached the established state. A useful alternative command is show ip bgp summary. The output should be similar to the following.

SanJose2# show ip bgp summary
BGP router identifier 172.16.32.1, local AS number 64512 BGP table version is 6, main routing table version 6
2 network entries using 288 bytes of memory
4 path entries using 320 bytes of memory
4/2 BGP path/bestpath attribute entries using 640 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory BGP using 1272 total bytes of memory
GP activity 2/0 prefixes, 4/0 paths, scan interval 60 secs
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal, r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter, x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

Network	Next Hop	Metric	LocPrf	Weight Path
* i 172.16.0.0	172.16.32.1	0	100	0 i
*>	0.0.0.0	0		32768 i
*> 192.168.100.0	192.168.1.5	0	150	0 200 i

SanJose1#
SanJose2# show ip bgp
BGP table version is 7, local router ID is 172.16.32.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal, r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter, x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

Network	Next Hop	Metric	LocPrf	Weight Path
* i	172.16.0.0	172.16.64.1	0	100	0 i
*>		0.0.0.0	0		32768 i
*>i	192.168.100.0	172.16.64.1	0	150	0 200 i
*		192.168.1.1	0	125	0 200 i

SanJose2#
This now indicates that routing to the loopback segment for ISP 192.168.100.0 /24 can be reached only through the link common to SanJose1 and ISP. SanJose2's next hop to 192.168.100.0/24 is SanJose1 because both routers have been configured using the next-hop-self command.
Step 9: Set BGP MED.
a. In the previous step we saw that SanJose1 and SanJose2 will route traffic for 192.168.100.0/24 using the link between SanJose1 and ISP. Examine what the return path ISP

takes to reach AS 64512. Notice that the return path is different from the original path. This is known as asymmetric routing and is not necessarily an unwanted trait.

```
ISP# show ip bgp
BGP table version is 22, local router ID is 192.168.100.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal, r RIB-failure, S
Stale, m multipath, b backup-path, f RT-Filter, x best-external, a additional-path, c RIB-
compressed
Set DF bit in IP header? [no]:
(0.0.0.0)
(0.0.0.0)
Reply to request 0 (28 ms). Received packet has options Total option bytes= 40, padded
length=40
Record route: (172.16.1.2)
(192.168.1.6)
(192.168.100.1)
(192.168.1.5)
(172.16.1.1)
(172.16.32.1) <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
Reply to request 1 (28 ms). Received packet has options Total option bytes= 40, padded
length=40
Record route: (172.16.1.2)
(192.168.1.5)
(172.16.1.1)
(172.16.32.1) <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
Reply to request 2 (28 ms). Received packet has options Total option bytes= 40, padded
length=40
Record route: (172.16.1.2)
(192.168.1.6)
(192.168.100.1)
(192.168.1.5)
(172.16.1.1)
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
Reply to request 3 (28 ms). Received packet has options Total option bytes= 40, padded
length=40
Record route: (172.16.1.2)
(192.168.1.6)
(192.168.100.1)
(192.168.1.5)
(172.16.1.1)
```

(172.16.32.1) <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
Reply to request 4 (28 ms). Received packet has options Total option bytes= 40, padded length=40
Record route: (172.16.1.2)
(192.168.1.6)
(192.168.100.1)
(192.168.1.5)
(172.16.1.1)
(172.16.32.1) <*>
(0.0.0.0)
(0.0.0.0)
(0.0.0.0)
End of list
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/28 ms SanJose2#

Step 10: Establish a default route. The final step is to establish a default route that uses a policy statement that adjusts to changes in the network.

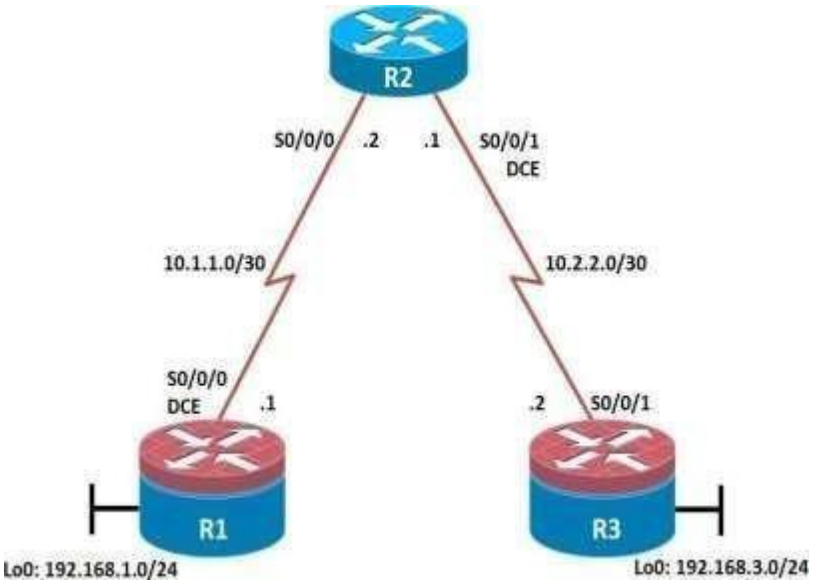
Configure ISP to inject a default route to both SanJose1 and SanJose2 using BGP using the default- originate command. This command does not require the presence of 0.0.0.0 in the ISP router. Configure the 10.0.0.0/8 network which will not be advertised using BGP. This network will be used to test the default route on SanJose1 and SanJose2.

ISP(config)# router bgp 200
ISP(config-router)# neighbor 192.168.1.6 default-originate ISP(config-router)# neighbor 192.168.1.2 default-originate ISP(config-router)# exit
ISP(config)# interface loopback 10
ISP(config-if)# ip address 10.0.0.1 255.255.255.0
ISP(config-if)#
a - application route
+ - replicated route, % - next hop override Gateway of last resort is 192.168.1.5 to network 0.0.0.0
B* 0.0.0.0/0 [20/0] via 192.168.1.5, 00:00:36
EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2 E1 - OSPF external type 1, E2 - OSPF external type 2
B* 0.0.0.0/0 [200/0] via 172.16.32.1, 00:00:06
172.16.0.0/16 is variably subnetted, 6 subnets, 3 masks S 172.16.0.0/16 is directly connected, Null0
172.16.1.0/24 is directly connected, Serial0/0/1 L 172.16.1.1/32 is directly connected, Serial0/0/1
172.16.32.0/24 [90/2297856] via 172.16.1.2, 05:49:25, Serial0/0/1
C 172.16.64.0/24 is directly connected, Loopback0 L 172.16.64.1/32 is directly connected, Loopback0 B 192.168.100.0/24 [200/0] via 172.16.32.1, 00:00:06
SanJose1#
SanJose2# show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

PRACTICAL NO: 04

AIM: Secure the Management Plane

Topology



Objectives

- Secure management access.
- Configure enhanced username password security.
- Enable AAA RADIUS authentication.
- Enable secure remote management.

Background

The management plane of any infrastructure device should be protected as much as possible. Controlling access to routers and enabling reporting on routers are critical to network security and should be part of a comprehensive security policy.

In this lab, you build a multi-router network and secure the management plane of routers R1 and R3.

Note: This lab uses Cisco 1941 routers with Cisco IOS Release 15.2 with IP Base. Depending on the router or switch model and Cisco IOS Software version, the commands available and output produced might vary from what is shown in this lab.

Required Resources

- 3 routers (Cisco IOS Release 15.2 or comparable)
- Serial and Ethernet cables

Step 1: Configure loopbacks and assign addresses.

Cable the network as shown in the topology diagram. Erase the startup configuration and reload each router to clear previous configurations. Using the addressing scheme in the diagram, apply the IP addresses to the interfaces on the R1, R2, and R3 routers.

You can copy and paste the following configurations into your routers to begin.

Note: Depending on the router model, interfaces might be numbered differently than those listed. You might need to alter the designations accordingly.

R1

hostname R1
interface Loopback 0 description R1 LAN
ip address 192.168.1.1 255.255.255.0
exit

!
interface Serial0/0/0 description R1 --> R2
ip address 10.1.1.1 255.255.255.252
clock rate 128000 no shutdown
exit
!
end
R2
hostname R2
!
interface Serial0/0/0 description R2 --> R1
ip address 10.1.1.2 255.255.255.252
no shutdown exit
interface Serial0/0/1 description R2 --> R3
ip address 10.2.2.1 255.255.255.252
clock rate 128000 no shutdown
exit
!
end
R3
hostname R3
!
interface Loopback0 description R3 LAN
ip address 192.168.3.1 255.255.255.0
exit
Step 2: Configure static routes.
On R1, configure a default static route to ISP.
R1(config)# ip route 0.0.0.0 0.0.0.0 10.1.1.2
On R3, configure a default static route to ISP.
R3(config)# ip route 0.0.0.0 0.0.0.0 10.2.2.1
On R2, configure two static routes.

i. R2(config)# ip	route	192.168.1.0 255.255.255.0	10.1.1.1
ii. R2(config)# ip	route	192.168.3.0 255.255.255.0	10.2.2.2

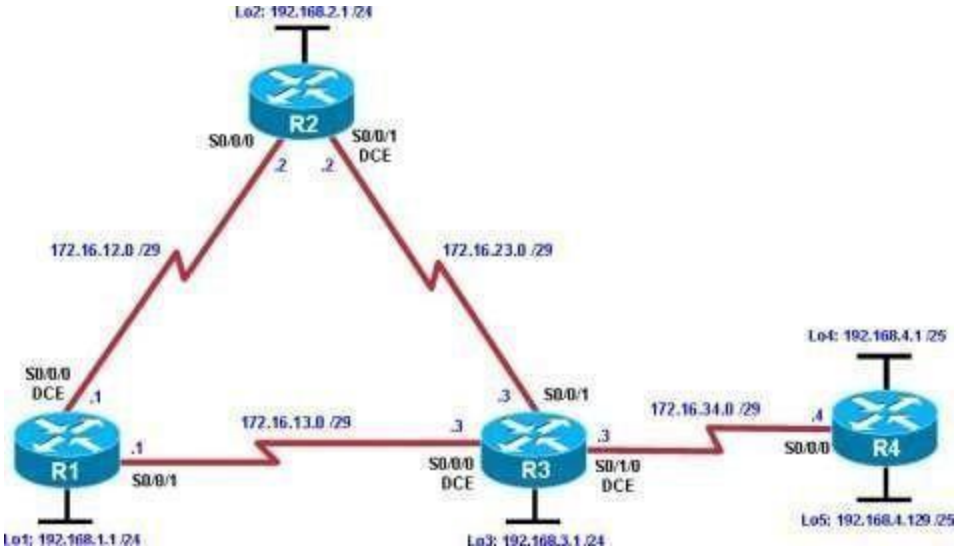
From the R1 router, run the following Tcl script to verify connectivity.
foreach address { 192.168.1.1
10.1.1.1
10.1.1.2
10.2.2.1
10.2.2.2
192.168.3.1
} { ping $address }
R1# tclsh
R1(tcl)#foreach address {
+>(tcl)#192.168.1.1
+>(tcl)#10.1.1.1
+>(tcl)#10.1.1.2
+>(tcl)#10.2.2.1
+>(tcl)#10.2.2.2
+>(tcl)#192.168.3.1
+>(tcl)#} { ping $address }

Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.2.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/14/16 ms Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/15/16 ms R1(tcl)#
Are the pings now successful?
Configure a console password and enable login for routers. For additional security, the exec-timeout command causes the line to log out after 5 minutes of inactivity. The logging synchronous command prevents console messages from interrupting command entry.
Note: To avoid repetitive logins during this lab, the exec-timeout command can be set to 0 0, R1(config-line)# password ciscovtypass R1(config-line)# exec-timeout 5 0 R1(config-line)#
To increase the encryption level of console and VTY lines, it is recommended to enable authentication using the local database. The local database consists of usernames and password combinations that are created locally on each device. The local and VTY lines are configured to refer to the local database when authenticating a user.
To create local database entry encrypted to level 4 (SHA256), use the username name secret password
global configuration command. In global configuration mode, enter the following command:
R1(config)# username JR-ADMIN secret class12345
R1(config)# username ADMIN secret class54321
Set the console line to use the locally defined login accounts.
R1(config)# line console 0 R1(config-line)# login local R1(config-line)# exit R1(config)#
Set the vty lines to use the locally defined login accounts.
R1(config)# line vty 0 4 R1(config-line)# login local R1(config-line)# end R1(config)#
Repeat the steps 4a to 4c on R3. i.
To verify the configuration, telnet to R3 from R1 and login using the ADMIN local database account.
R1# telnet 10.2.2.2
Trying 10.2.2.2 ... Open
Username: ADMIN
Password:
R3>

PRACTICAL NO: 05

AIM: Configure and Verify Path Control

Topology



Objectives

- Configure and verify policy-based routing.
- Select the required tools and commands to configure policy-based routing operations.
- Verify the configuration and operation by using the proper show and debug commands.

Background

You want to experiment with policy-based routing (PBR) to see how it is implemented and to study how it could be of value to your organization. To this end, you have interconnected and configured a test network with four routers. All routers are exchanging routing information using EIGRP.

Note: This lab uses Cisco 1841 routers with Cisco IOS Release 12.4(24)T1, and the Advanced IP Services image c1841-advipservicesk9-mz.124-24.T1.bin. You can use other routers (such as 2801 or 2811) and Cisco IOS Software versions if they have comparable capabilities and features. Depending on the router and software version, the commands available and output produced might vary from what is shown in this lab.

Required Resources

- 4 routers (Cisco 1841 with Cisco IOS Release 12.4(24)T1 Advanced IP Services or comparable)
- Serial and console cables

Step 1: Prepare the routers for the lab.

Cable the network as shown in the topology diagram. Erase the startup configuration, and reload each router to clear previous configurations.

Step 2: Configure router hostname and interface addresses.

Using the addressing scheme in the diagram, create the loopback interfaces and apply IP addresses to these and the serial interfaces on R1, R2, R3, and R4. On the serial interfaces connecting R1 to R3 and R3 to R4, specify the bandwidth as 64 Kb/s and set a clock rate on the DCE using the clock rate 64000 command. On the serial interfaces connecting R1 to R2 and R2 to R3, specify the bandwidth as 128 Kb/s and set a clock rate on the DCE using the clock rate 128000 command.

You can copy and paste the following configurations into your routers to begin.

Note: Depending on the router model, interfaces might be numbered differently than those listed. You might need to alter them accordingly.

```
Router R1
hostname R1
!
interface Lo1 description R1 LAN
ip address 192.168.1.1 255.255.255.0
!
interface Serial0/0/0 description R1 --> R2
ip address 172.16.12.1 255.255.255.248
clock rate 128000 bandwidth 128no shutdown
!
interface Serial0/0/1 description R1 --> R3
ip address 172.16.13.1 255.255.255.248
bandwidth 64no shutdown
!
end
Router R2
hostname R2
!
interface Lo2 description R2 LAN
ip address 192.168.2.1 255.255.255.0
!
interface Serial0/0/0 description R2 --> R1
ip address 172.16.12.2 255.255.255.248
bandwidth 128no shutdown
interface Serial0/0/1 description R2 --> R3
ip address 172.16.23.2 255.255.255.248
clock rate 128000
Router R2
router eigrp 1
network 192.168.2.0
network 172.16.12.0 0.0.0.7
network 172.16.23.0 0.0.0.7
no auto-summary
Router R3
router eigrp 1
network 192.168.3.0
network 172.16.13.0 0.0.0.7
network 172.16.23.0 0.0.0.7
network 172.16.34.0 0.0.0.7
no auto-summary
Router R4
router eigrp 1
network 192.168.4.0
network 172.16.34.0 0.0.0.7
no auto-summary
```

You should see EIGRP neighbor relationship messages being generated.

```
foreach address { 172.16.12.1
172.16.12.2
```

172.16.13.1
172.16.13.3
172.16.23.2
172.16.23.3
172.16.34.3
172.16.34.4
192.168.1.1
192.168.2.1
192.168.3.1
192.168.4.1
192.168.4.129
} { ping $address }
You should get ICMP echo replies for every address pinged. Make sure to run the Tcl script on each router.
Step 5: Verify the current path.
Before you configure PBR, verify the routing table on R1.
On R1, use the show ip route command. Notice the next-hop IP address for all networks discovered by EIGRP.
R1# show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter areaN1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2ia - IS-IS inter area, * - candidate default, U - per-user static
route
o - ODR, P - periodic downloaded static route
Gateway of last resort is not set 172.16.0.0/29 is subnetted, 4 subnets
D 172.16.34.0 [90/41024000] via 172.16.13.3, 00:05:18, Serial0/0/1D
172.16.23.0
[90/21024000] via 172.16.12.2, 00:05:18, Serial0/0/0
172.16.12.0 is directly connected, Serial0/0/0C 172.16.13.0 is directly connected, Serial0/0/1
192.168.4.0/25 is subnetted, 2 subnets
192.168.4.0 [90/41152000] via 172.16.13.3, 00:05:06, Serial0/0/1 D 192.168.4.128
[90/41152000] via 172.16.13.3, 00:05:06, Serial0/0/1
192.168.1.0/24 is directly connected, Loopback1
192.168.2.0/24 [90/20640000] via 172.16.12.2, 00:05:18, Serial0/0/0D 192.168.3.0/24
[90/21152000] via 172.16.12.2, 00:05:18, Serial0/0/0
*Feb 23 07:07:46.467: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1 (Se
rial0/0/0), len 28, policy routed
*Feb 23 07:07:46.467: IP: Serial0/1/0 to Serial0/0/0 172.16.13.1
*Feb 23 07:07:55.471: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, le
n 28, policy match
*Feb 23 07:07:55.471: IP: route map R3-to-R1, item 10, permit
*Feb 23 07:07:55.471: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1 (Se
rial0/0/0), len 28, policy routed
*Feb 23 07:07:55.471: IP: Serial0/1/0 to Serial0/0/0 172.16.13.1
*Feb 23 07:07:55.471: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, le
n 28, policy match
*Feb 23 07:07:55.471: IP: route map R3-to-R1, item 10, permit

*Feb 23 07:07:55.475: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1 (Se
rial0/0/0), len 28, policy routed
*Feb 23 07:07:55.475: IP: Serial0/1/0 to Serial0/0/0 172.16.13.1
*Feb 23 07:07:55.475: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, le
n 28, FIB policy match
*Feb 23 07:07:55.475: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, g=
172.16.13.1, len 28, FIB policy routed
*Feb 23 07:08:04.483: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, le
n 28, FIB policy match
*Feb 23 07:08:04.483: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, g=
172.16.13.1, len 28, FIB policy routed
*Feb 23 07:08:04.491: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, le
n 28, FIB policy match
*Feb 23 07:08:04.491: IP: s=192.168.4.129 (Serial0/1/0), d=192.168.1.1, g=
172.16.13.1, len 28, FIB policy routed

On R3, display the policy and matches using the show route-map command.

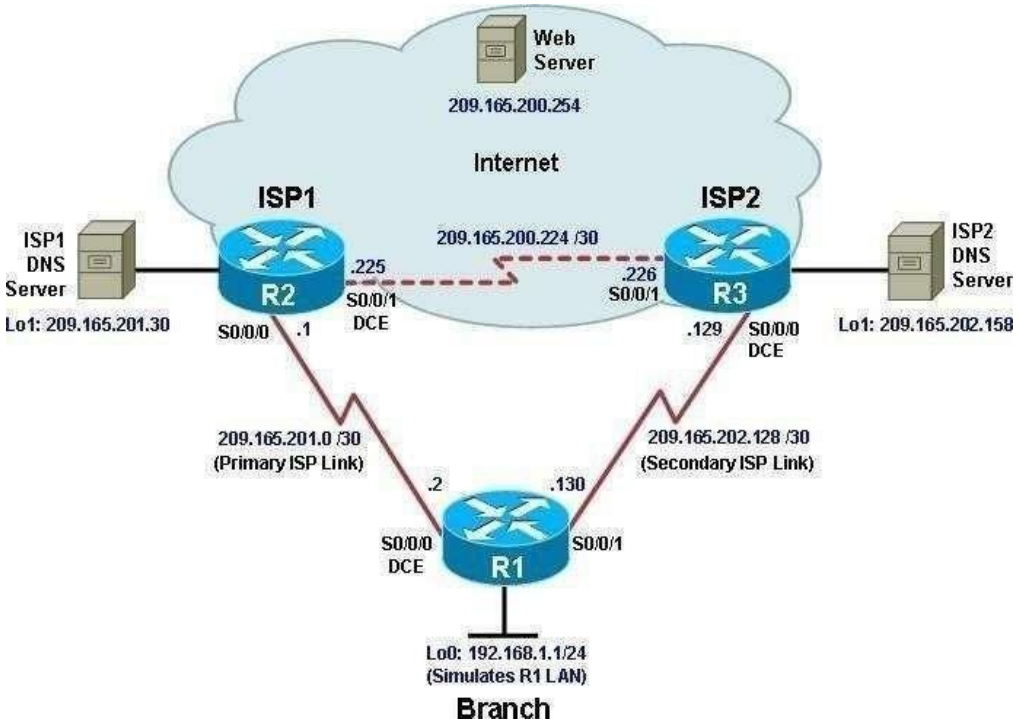
R3# show route-map
route-map R3-to-R1, permit, sequence 10 Match clauses:
ip address (access-lists): PBR- ACLSet clauses:
ip next-hop 172.16.13.1
Policy routing matches: 12 packets, 384 bytes

Note: There are now matches to the

PRACTICAL NO: 06

AIM: Configure IP SLA Tracking and Path Control with gateway

Topology



Objectives

- Configure and verify the IP SLA feature.
- Test the IP SLA tracking feature.
- Verify the configuration and operation using show and debug commands.

Background

You want to experiment with the Cisco IP Service Level Agreement (SLA) feature to study how it could be of value to your organization.

At times, a link to an ISP could be operational, yet users cannot connect to any other outside Internet resources. The problem might be with the ISP or downstream from them. Although policy-based routing (PBR) can be implemented to alter path control, you will implement the Cisco IOS SLA feature to monitor this behavior and intervene by injecting another default route to a backup ISP.

Required Resources

- 3 routers (Cisco IOS Release 15.2 or comparable)
- Serial and Ethernet cables

Step 1: Configure loopbacks and assign addresses.

Cable the network as shown in the topology diagram. Erase the startup configuration and reload each router to clear the previous configurations. Using the addressing scheme in the diagram, create the loopback interfaces and apply IP addresses to them as well as the serial interfaces on R1, ISP1, and ISP2.

You can copy and paste the following configurations into your routers to begin.

Note: Depending on the router model, interfaces might be numbered differently than those listed. You might need to alter them accordingly.

Router R1

hostname R1

interface Loopback 0 description R1 LAN

ip address 192.168.1.1 255.255.255.0
interface Serial0/0/0 description R1 --> ISP1
ip address 209.165.201.2 255.255.255.252
clock rate 128000
bandwidth 128 no shutdown
interface Serial0/0/1 description R1 --> ISP2
ip address 209.165.202.130 255.255.255.252
bandwidth 128 no shutdown

Router ISP1 (R2)

hostname ISP1
interface Loopback0
description Simulated Internet Web Server ip address 209.165.200.254 255.255.255.255
interface Loopback1 description ISP1 DNS Server
ip address 209.165.201.30 255.255.255.255
interface Serial0/0/0 description ISP1 --> R1
ip address 209.165.201.1 255.255.255.252
bandwidth 128 no shutdown
interface Serial0/0/1 description ISP1 --> ISP2
ip address 209.165.200.225 255.255.255.252
clock rate 128000

Router ISP2 (R3)

hostname ISP2 interface Loopback0
description Simulated Internet Web Server ip address 209.165.200.254 255.255.255.255
interface Loopback1 description ISP2 DNS Server
ip address 209.165.202.158 255.255.255.255
interface Serial0/0/0 description ISP2 --> R1
ip address 209.165.202.129 255.255.255.252
clock rate 128000
bandwidth 128 no shutdown
interface Serial0/0/1 description ISP2 --> ISP1
ip address 209.165.200.226 255.255.255.252
bandwidth 128 no shutdown

Verify the configuration by using the show interfaces description command. The output from router R1 is shown here as an example.

R1# show interfaces description | include up

Se0/0/0	up	up	R1 --> ISP1
Se0/0/1	up	up	R1 --> ISP2
Lo0	up	up	R1 LAN

R1#

All three interfaces should be active. Troubleshoot if necessary.

Step 2: Configure static routing.

The current routing policy in the topology is as follows:
Router R1 establishes connectivity to the Internet through ISP1 using a default static route.
ISP1 and ISP2 have dynamic routing enabled between them, advertising their respective public address pools.
ISP1 and ISP2 both have static routes back to the ISP LAN.
Note: For the purpose of this lab, the ISPs have a static route to an RFC 1918 private network address on the branch router R1. In an actual branch implementation, Network Address Translation (NAT) would be configured for all traffic exiting the branch LAN. Therefore, the

static routes on the ISP routers would be pointing to the provided public pool of the branch office.
Implement the routing policies on the respective routers. You can copy and paste the following configurations.
Router R1
R1(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1
R1(config)#
Router ISP1 (R2)
ISP1(config)# router eigrp 1
ISP1(config-router)# network 209.165.200.224 0.0.0.3
ISP1(config-router)# network 209.165.201.0 0.0.0.31
ISP1(config-router)# no auto-summary ISP1(config-router)# exit ISP1(config)#
ISP1(config-router)# ip route 192.168.1.0 255.255.255.0 209.165.201.2
ISP1(config)#
Router ISP2 (R3)
ISP2(config)# router eigrp 1
ISP2(config-router)# network 209.165.200.224 0.0.0.3
ISP2(config-router)# network 209.165.202.128 0.0.0.31
ISP2(config-router)# no auto-summary ISP2(config-router)# exit ISP2(config)#
ISP2(config)# ip route 192.168.1.0 255.255.255.0 209.165.202.130
ISP2(config)#
EIGRP neighbor relationship messages on ISP1 and ISP2 should be generated. Troubleshoot if necessary.
The Cisco IOS IP SLA feature enables an administrator to monitor network performance between Cisco devices (switches or routers) or from a Cisco device to a remote IP device. IP SLA probes continuously check the reachability of a specific destination, such as a provider edge router interface, the DNS server of the ISP, or any other specific destination, and can conditionally announce a default route only if the connectivity is verified.
Before implementing the Cisco IOS SLA feature, you must verify reachability to the Internet servers. From router R1, ping the web server, ISP1 DNS server, and ISP2 DNS server to verify connectivity. You can copy the following Tcl script and paste it into R1.
foreach address { 209.165.200.254
209.165.201.30
209.165.202.158 } {
ping $address source 192.168.1.1 }
All pings should be successful. Troubleshoot if necessary.
Trace the path taken to the web server, ISP1 DNS server, and ISP2 DNS server. You can copy the following Tcl script and paste it into R1.
foreach address { 209.165.200.254
209.165.201.30
209.165.202.158 } {
trace $address source 192.168.1.1 }
Through which ISP is traffic flowing?
R1# show ip sla configuration 22 IP SLAs, Infrastructure Engine-II. Entry number: 22
Owner:
Tag:
Type of operation to perform: icmp-echo
Target address/Source address: 209.165.201.158/0.0.0.0 Type Of Service parameter: 0x0
Request size (ARR data portion): 28 Operation timeout (milliseconds): 5000 Verify data: No
Vrf Name:

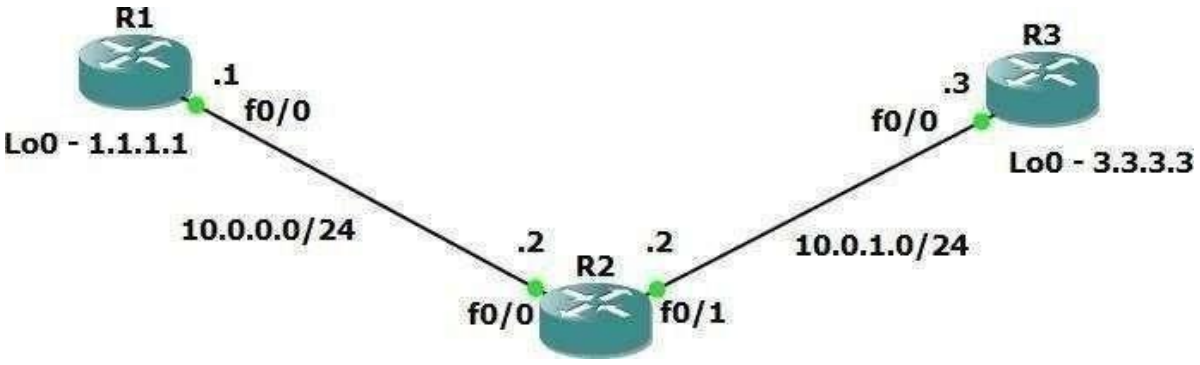
Schedule:
Operation frequency (seconds): 10 (not considered if randomly scheduled)
Next Scheduled Start Time: Start Time already passed Group Scheduled : FALSE
Randomly Scheduled : FALSE Life (seconds): Forever Entry Ageout (seconds): never
Recurring (Starting Everyday): FALSE Status of entry (SNMP RowStatus): Active
Threshold (milliseconds): 5000 (not considered if react RTT is configured)
Distribution Statistics:
Number of statistic hours kept: 2
Number of statistic distribution buckets kept: 1 Statistic distribution interval (milliseconds): 20
History Statistics:
Number of history Lives kept: 0 Number of history Buckets kept: 15 History Filter Type: None
Enhanced History:
R1#
R1# show ip sla statistics 22
IPSLAs Latest Operation Statistics
IPSLA operation id: 22
Latest RTT: 16 milliseconds
Latest operation start time: 10:38:29 UTC Sat Jan 10 2015 Latest operation return code: OK
Number of successes: 82 Number of failures: 0
Operation time to live: Forever
Specify the level of sensitivity to changes of tracked objects to 10 seconds of down delay and 1 second of up delay using the delay down 10 up 1 command. The delay helps to alleviate the effect of flapping objects—objects that are going down and up rapidly. In this situation, if the DNS server fails momentarily and comes back up within 10 seconds, there is no impact.
R1(config-track)# delay down 10 up 1
R1(config-track)# exit
R1(config)#
To view routing table changes as they happen, first enable the debug ip routing command.
R1# debug ip routing
IP routing debugging is on R1#
Configure the floating static route that will be implemented when tracking object 1 is active. Use the ip route 0.0.0.0 0.0.0.0 209.165.201.1 2 track 1 command to create a floating static default route via 209.165.201.1 (ISP1). Notice that this command references the tracking object number 1, which in turn references IP SLA operation number 11.
R1(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1 2 track 1
R1(config)#
Jan 10 10:45:39.119: RT: updating static 0.0.0.0/0 (0x0) : via 209.165.201.1 0 1048578
C 192.168.1.0/24 is directly connected, Loopback0 L 192.168.1.1/32 is directly connected, Loopback0
209.165.201.0/24 is variably subnetted, 2 subnets, 2 masks C 209.165.201.0/30 is directly connected, Serial0/0/0
L 209.165.201.2/32 is directly connected, Serial0/0/0 209.165.202.0/24 is variably subnetted, 2 subnets, 2 masks
C 209.165.202.128/30 is directly connected, Serial0/0/1 L 209.165.202.130/32 is directly connected, Serial0/0/1 R1#

PRACTICAL NO: 07

AIM: Configuring Basic MPLS Using OSPF

Step 1 – IP addressing of MPLS Core and OSPF

First bring 3 routers into your topology R1, R2, R3 position them as below. We are going to address the routers and configure ospf to ensure loopbackto loopback connectivity between R1 and R3




```
R1

hostname
R1int lo0

ip add 1.1.1.1 255.255.255.255

ip ospf 1 area 0



int f0/0

ip add 10.0.0.1 255.255.255.0

no shut

ip ospf 1 area 0



R2

hostname

r2

int lo0
```

```
ip add 2.2.2.2 255.255.255.255

ip ospf 1 area 0


int f0/0

ip add 10.0.0.2 255.255.255.0

no shut

ip ospf 1 area 0


int f0/1

ip add 10.0.1.2 255.255.255.0

no shut

ip ospf 1 area 0



R3

hostname R3int

lo0

ip add 3.3.3.3 255.255.255.255

ip ospf 1 area 0


int f0/0

ip add 10.0.1.3 255.255.255.0

no shut

ip ospf 1 area 0
```

You should now have full ip connectivity between R1, R2, R3 to verify thiswe need to see if we can ping between the loopbacks of R1 and R3

```
R1#ping 3.3.3.3 source lo0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 3.3.3.3, timeout is 2seconds:
Packet sent with a source address of 1.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-tripmin/avg/max =
40/52/64 ms
R1#
```

You could show the routing table here, but the fact that you can ping between the loopbacks is verification enough and it is safe to move on.

Step 2 – Configure LDP on all the interfaces in the MPLS Core

In order to run MPLS you need to enable it, there are two ways to do this.

At each interface enter the mpls ip command

Under the ospf process use the mpls ldp autoconfig command

For this tutorial we will be using the second option, so go int the ospf process and enter mpls ldp autoconfig – this will enable mpls label distribution protocol on every interface running ospf under that specificprocess.

```
R1

router ospf 1

mpls ldp autoconfig


R2

router ospf 1

mpls ldp autoconfig


R3

router ospf 1

mpls ldp autoconfig
```

You should see log messages coming up showing the LDP neighbors are up

```
R2#

*Mar 1 00:31:53.643: %SYS-5-CONFIG_I: Configured fromconsole

*Mar   1 00:31:54.423: %LDP-5-NBRCHG: LDP Neighbor
1.1.1.1:0 (1) is UPR2#

*Mar   1 00:36:09.951: %LDP-5-NBRCHG: LDP Neighbor
3.3.3.3:0 (2) is UP
```

To verify the mpls interfaces the command is very simple – sh mpls interface

This is done on R2 and you can see that both interfaces are running mplsand using LDP

```
R2#sh mpls interface
Interface              IP           Tunnel
Operationa
l
FastEthernet0/0        Yes (ldp)    No       Yes
FastEthernet0/1        Yes (ldp)    No       Yes
```

You can also verify the LDP neighbors with the sh mpls ldp neighbors command.

We now need to repeat this process for R3 & R6

Router 6 will peer OSPF using process number 2 to a VRF configured onR3. It will use the local site addressing of 192.168.2.0/24.

```
R6
int lo0
ip add 6.6.6.6 255.255.255.255
ip ospf 2 area 2int
f0/0
ip add 192.168.2.6 255.255.255.0
ip ospf 2 area 2


no shut


R3
int f0/1
no shut
ip add 192.168.2.3 255.255.255.0
```

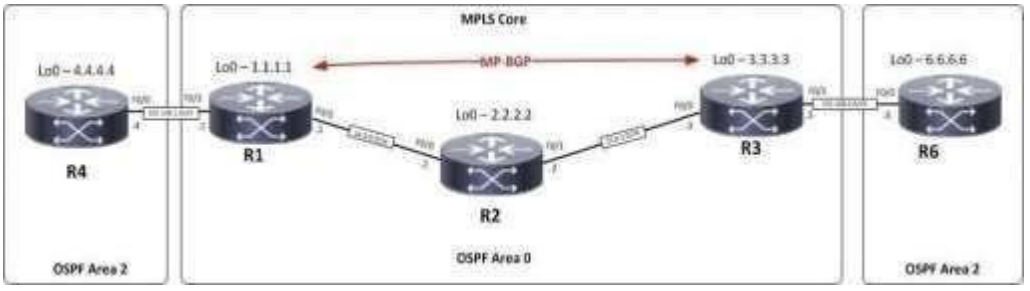
We also need to configure a VRF onto R3 as well.
So now we have configured the VRF on R3 we need to move the interface F0/1 into that VRF
R3
int f0/1
ip vrf forwarding RED
Now notice what happens when you do that – the IP address is removed

```
R3(config-if)#ip vrf forwarding RED

% Interface FastEthernet0/1 IP address 192.168.2.1removed due to
enabling VRF RED
```

You just need to re-apply it

```
R3
int f0/1
ip address 192.168.2.1 255.255.255.0
```

Now if we view the config on R3 int f0/1 you can see the VRF configured.



R1,R2,R3 form the MPLS Core and are running OSPF with all loopbacks running a /32 address and all have full connectivity. R1 and R3 are peeringwith MP-BGP. LDP is enabled on all the internal interfaces. The external interfaces of the MPLS core have been placed into

a VRF called RED and then a site router has been joined to that VRF on each side of the MPLS core – (These represent a small office)
The final step to get full connectivity across the MPLS core is to redistributethe routes in OSPF on R1 and R3 into MP-BGP and MP-BGP into OSPF, this is what we are going to do now.
We need to redistribute the OSPF routes from R4 into BGP in the VRF on R1, the OSPF routes from R6 into MP-BGP in the VRF
Before we start lets do some verifications
Check the routes on R4

```
R4#sh ip route

4.0.0.0/32 is subnetted, 1 subnets

C 4.4.4.4 is directly connected, Loopback0
```

on R3 and then theroutes in MP-BGP in R1 and R3 back out to OSPF

```
Type escape sequence to

abort.Tracing the route to

6.6.6.6



1 192.168.1.1 20 msec 8 msec 8 msec

  2 10.0.0.2 [MPLS: Labels 17/20 Exp 0] 36 msec 40 msec
36 msec

  3 192.168.2.1 [MPLS: Label 20 Exp 0] 16 msec 40 msec 16msec

  4 192.168.2.6 44 msec 40 msec 56 msecR4#
```

UNIVERSITY OF MUMBAI

Practical Journal of

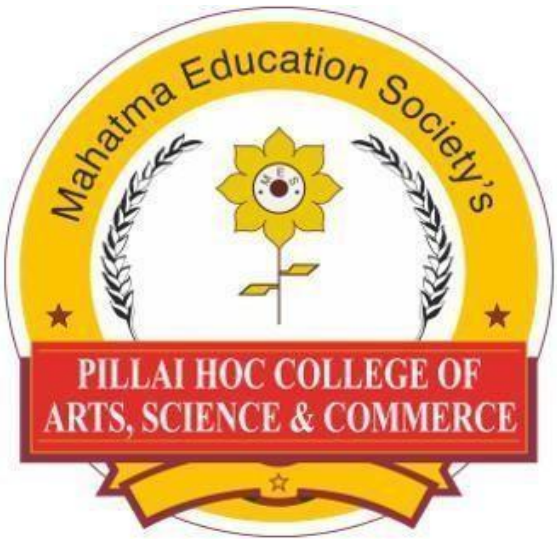
Big Data Analytics, Modern Networking & Computer Vision

M.Sc. (Information Technology) Part-I

Submitted by

PATIL PAVAN MAHESH

Seat No: 26MSCIT203010



DEPARTMENT OF INFORMATION TECHNOLOGY

PILLAI HOC COLLEGE OF ARTS,SCIENCE & COMMERCE(AUTONOMOUS),

RASAYANI, 410207 MAHARASTRA

(Affiliated to Mumbai University)

RASAYANI-410207, MAHARASHTRA

NAAC Accredited with 'A+' Grade in Cycle

2025-2026

MAHATMA EDUCATION SOCIETY'S
Pillai HOC College of Arts, Science & Commerce (Autonomous),
Rasayani 410207
(Affiliated to Mumbai University)
NAAC Accredited with 'A+' Grade in Cycle

DEPARTMENT OF

INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the experiment work entered in this journal is as per the syllabus in **M.Sc. (Information Technology) Part-I, Semester-II**; class prescribed by University of Mumbai for **Computer Vision**, of Mahatma Education Society's **Pillai HOC College of Arts, Science & Commerce(Autonomous), Rasayani** by **Patil Pavan Mahesh** during Academic year **2025-2026.**
Seat No: **26MSCIT203010**

In-Charge **Co-Ordinator**

External Examiner **Principal**

Date **College Seal**

COMPUTER VISION

INDEX

PRACTICAL NO: 01

AIM: A] Perform Geometric transformation.

Theory:

Geometric Transformations:

Geometric transformation is a fundamental technique used in image processing that involves manipulating the spatial arrangement of pixels in an image. It is used to modify the geometric properties of an image, such as its size, shape, position, and orientation.

OpenCV provides two transformation functions, cv2.warpAffine and cv2.warpPerspective, with which you can have all kinds of transformations. cv2.warpAffine takes a 2x3 transformation matrix while cv2.warpPerspective takes a 3x3 transformation matrix as input.

Translation

Translation is the shifting of object‘s location. If you know the shift in (x,y) direction, let it be (t_x, t_y), you can create the transformation matrix M as follows:

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

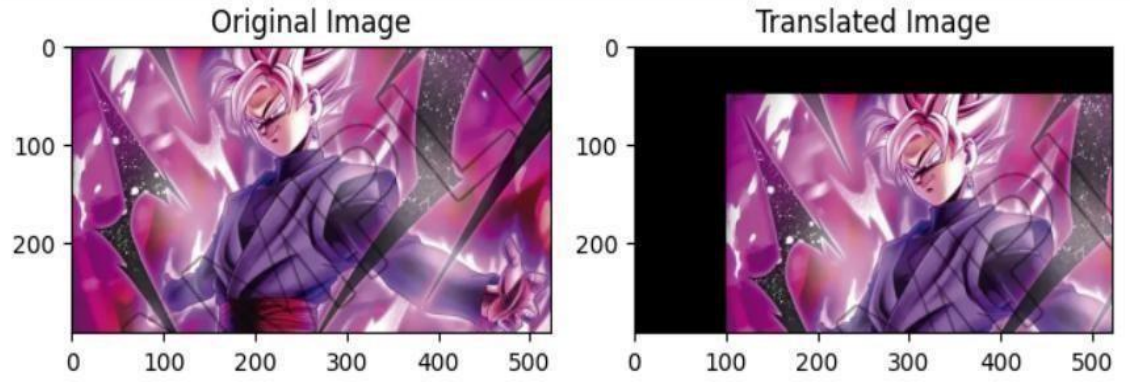
You can take make it into a NumPy array of type np.float32 and pass it into cv2.warpAffine() function. See below example for a shift of (100,50):

Warning:

Third argument of the cv2.warpAffine() function is the size of the output image, which should
be in the form of (width, height). Remember width = number of columns, and height = number of rows.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
img = cv2.imread("/content/goku black.jpeg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
rows, cols, channels = img_rgb.shape
M = np.float32([[1, 0, 100], [0, 1, 50]])
dst = cv2.warpAffine(img_rgb, M, (cols, rows))
fig, axs = plt.subplots(1, 2, figsize=(7, 4))
axs[0].imshow(img_rgb)
axs[0].set_title('Original Image')
axs[1].imshow(dst)
axs[1].set_title('Translated Image')
plt.tight_layout()
plt.show()
```

Output:



AIM: A] Image Scaling

Theory:

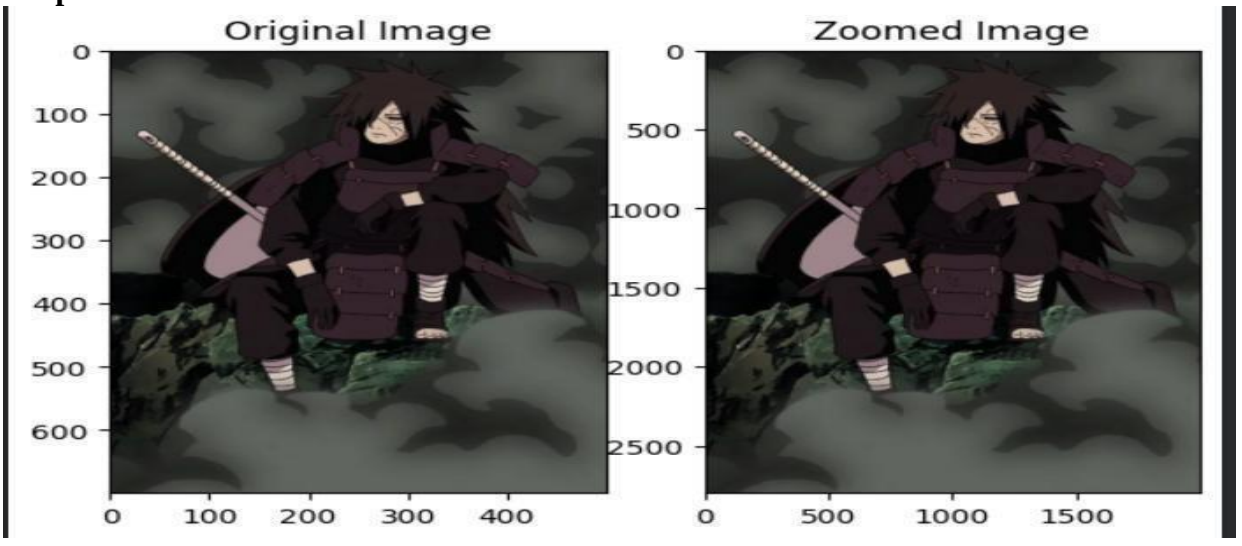
Scaling:

Scaling is just resizing of the image. OpenCV comes with a function cv2.resize() for this purpose. The size of the image can be specified manually, or you can specify the scaling factor.

Different interpolation methods are used. Preferable interpolation methods are cv2.INTER_AREA for shrinking and cv2.INTER_CUBIC (slow) & cv2.INTER_LINEAR for zooming. By default, interpolation method used is cv2.INTER_LINEAR for all resizing purposes. You can resize an input image either of following methods:

```
import cv2

import matplotlib.pyplot as plt

import numpy as np

img = cv2.imread("/content/photo.jpeg")

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

rows, cols, channels = img_rgb.shape

resize_img = cv2.resize(img_rgb, (0, 0), fx=4, fy=4, interpolation=cv2.INTER_CUBIC)

plt.subplot(121), plt.imshow(img_rgb), plt.title('Original Image')

plt.subplot(122), plt.imshow(resize_img), plt.title('Zoomed Image')

plt.show()
```

Output:



AIM: B]Image Shrinking

Theory:

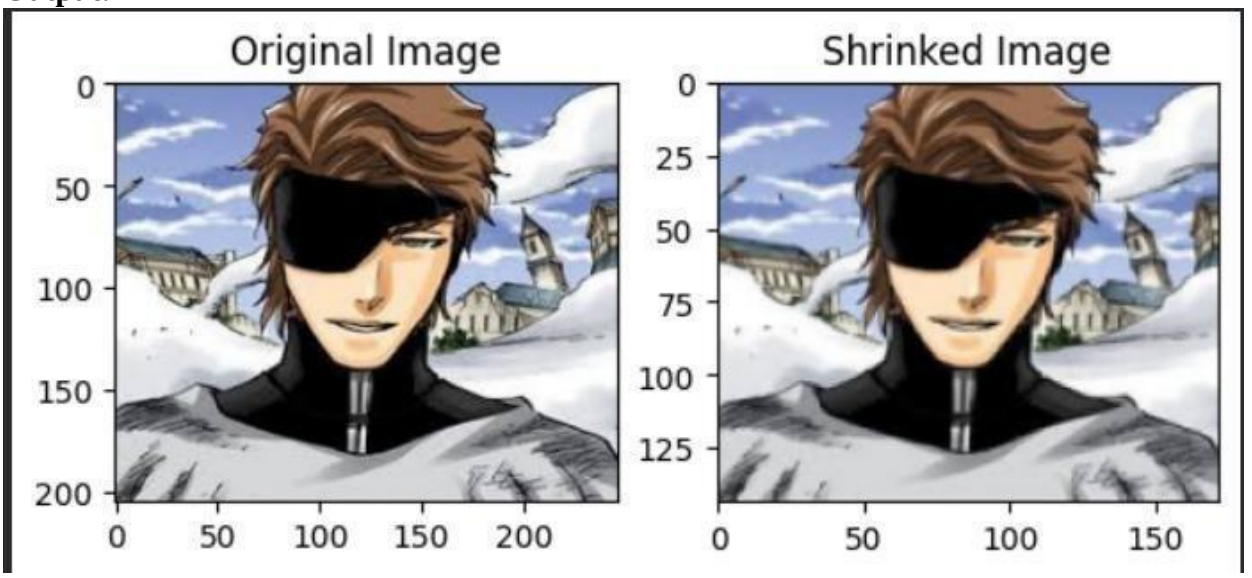
Shrinking:

Image shrinking in Python involves reducing the dimensions of an image, effectively making it smaller while maintaining its aspect ratio. This process is commonly done using libraries like PIL (Python Imaging Library) or its fork, Pillow. By resizing the image to smaller dimensions, either by specifying new dimensions or a scaling factor, you can achieve the desired reduction in size.

This is useful for tasks like image compression, thumbnail generation, or preparing images for web display, where smaller file sizes are advantageous. The process typically involves opening the image, calculating the new dimensions, resizing the image accordingly, and then saving the resized image to a new file.

```
import cv2

import matplotlib.pyplot as plt

import numpy as np

img = cv2.imread("/content/images.jpg")

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

rows, cols, channels = img_rgb.shape

resize_img = cv2.resize(img_rgb, (0, 0), fx=0.7, fy=0.7,
interpolation=cv2.INTER_AREA)

plt.subplot(121), plt.imshow(img_rgb), plt.title('Original Image')

plt.subplot(122), plt.imshow(resize_img), plt.title('Shrinked Image')

plt.show()
```

Output:



AIM: C] Image Rotation Theory: Rotation:

Rotation of an image for an angle θ is achieved by the transformation matrix of the form

$$M = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

But OpenCV provides scaled rotation with adjustable center of rotation so that you can rotate at any location you prefer. Modified transformation matrix is given by

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot center.x - \beta \cdot center.y \\ -\beta & \alpha & \beta \cdot center.x + (1-\alpha)\cdot center.y \end{bmatrix}$$

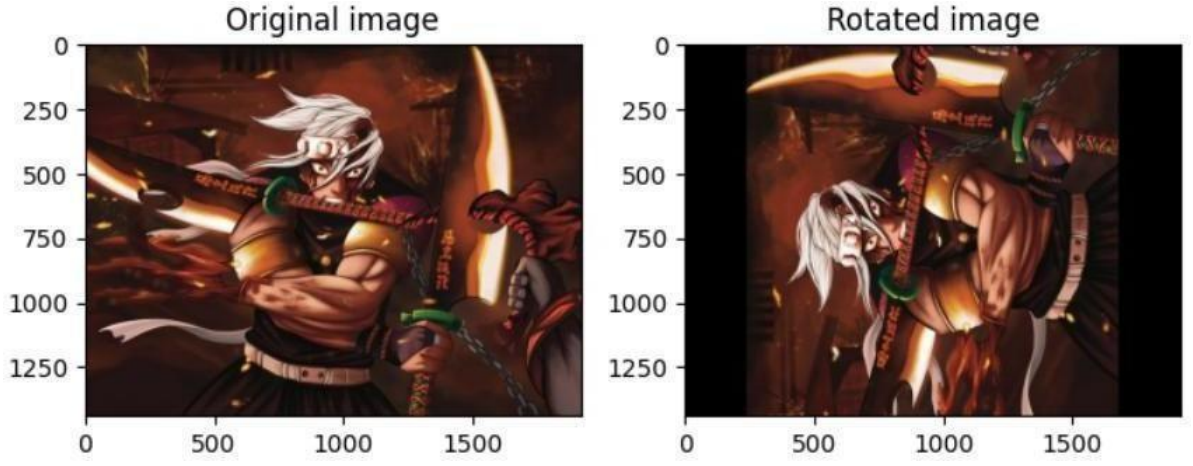
where:

$$\alpha = scale \cdot \cos\theta,$$
$$\beta = scale \cdot \sin\theta$$

To find this transformation matrix, OpenCV provides a function, cv2.getRotationMatrix2D. Check below example which rotates the image by 90 degree with respect to center without any scaling.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
img = cv2.imread("/content/tengen.jpg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
rows, cols, channels = img_rgb.shape
center = (cols // 2, rows // 2)
angle = 90
scale = 1
rotation_matrix = cv2.getRotationMatrix2D(center, angle, scale)
rotated_image = cv2.warpAffine(img_rgb, rotation_matrix, (cols, rows))
fig, axs = plt.subplots(1, 2, figsize=(7, 4))
axs[0].imshow(img_rgb)
axs[0].set_title("Original image")
axs[1].imshow(rotated_image)
axs[1].set_title("Rotated image")
plt.tight_layout()
plt.show()
```

Output:

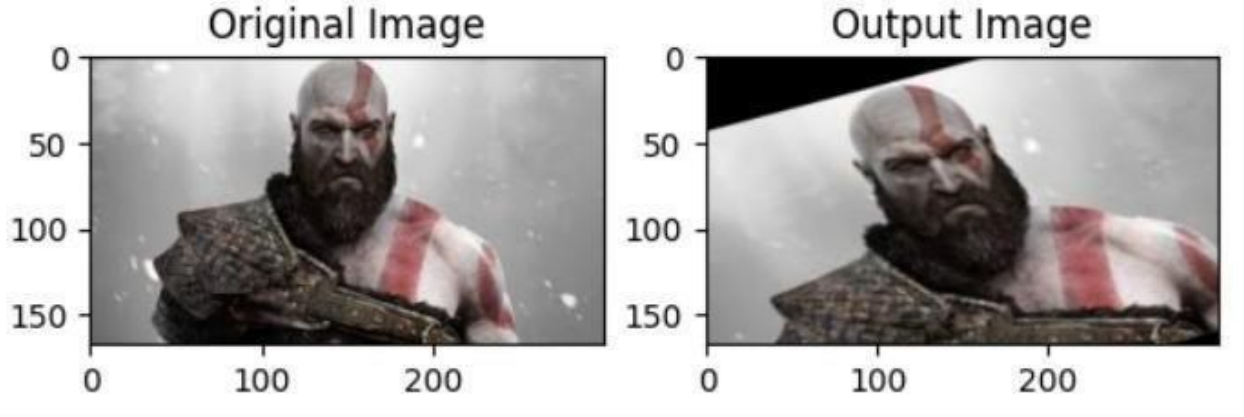


AIM: D] Affine Transformation

Theory:
In affine transformation, all parallel lines in the original image will still be parallel in the output image. To find the transformation matrix, we need three points from input image and their corresponding locations in output image. Then cv2.getAffineTransform will create a 2x3 matrix which is to be passed to cv2.warpAffine.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
img = cv2.imread("/content/kratos.jpg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
rows, cols, channels = img_rgb.shape
pts1 = np.float32([[50, 50], [200, 50], [50, 200]])
pts2 = np.float32([[10, 100], [200, 50], [100, 250]])
M = cv2.getAffineTransform(pts1, pts2)
dst = cv2.warpAffine(img_rgb, M, (cols, rows))
plt.subplot(121), plt.imshow(img_rgb), plt.title('Original Image')
plt.subplot(122), plt.imshow(dst), plt.title('Output Image')
plt.show()
```

Output:



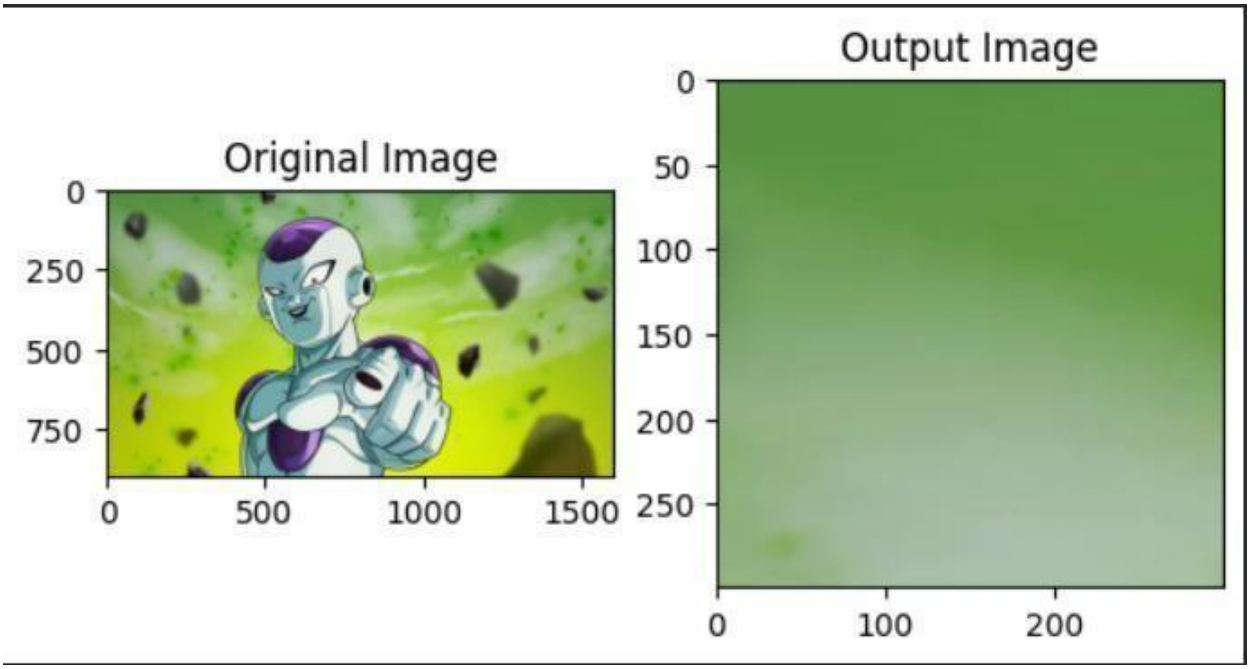
AIM: E] Perspective Transformation.

Theory:

For perspective transformation, you need a 3x3 transformation matrix. Straight lines will remain straight even after the transformation. To find this transformation matrix, you need 4 points on the input image and corresponding points on the output image. Among these 4 points, 3 of them should not be collinear. Then transformation matrix can be found by the function cv2.getPerspectiveTransform. Then apply cv2.warpPerspective with this 3x3 transformation matrix.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
img = cv2.imread("/content/frieza.jpeg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
rows, cols, channels = img_rgb.shape
pts1 = np.float32([[133, 34], [226, 16], [133, 206], [226, 219]])
pts2 = np.float32([[0, 0], [300, 0], [0, 300], [300, 300]])
M = cv2.getPerspectiveTransform(pts1, pts2)
dst = cv2.warpPerspective(img_rgb, M, (300, 300))
plt.subplot(121), plt.imshow(img_rgb), plt.title('Original Image')
plt.subplot(122), plt.imshow(dst), plt.title('Output Image')
plt.show()
```

Output:



AIM: F] Shearing X-axis.

Theory:

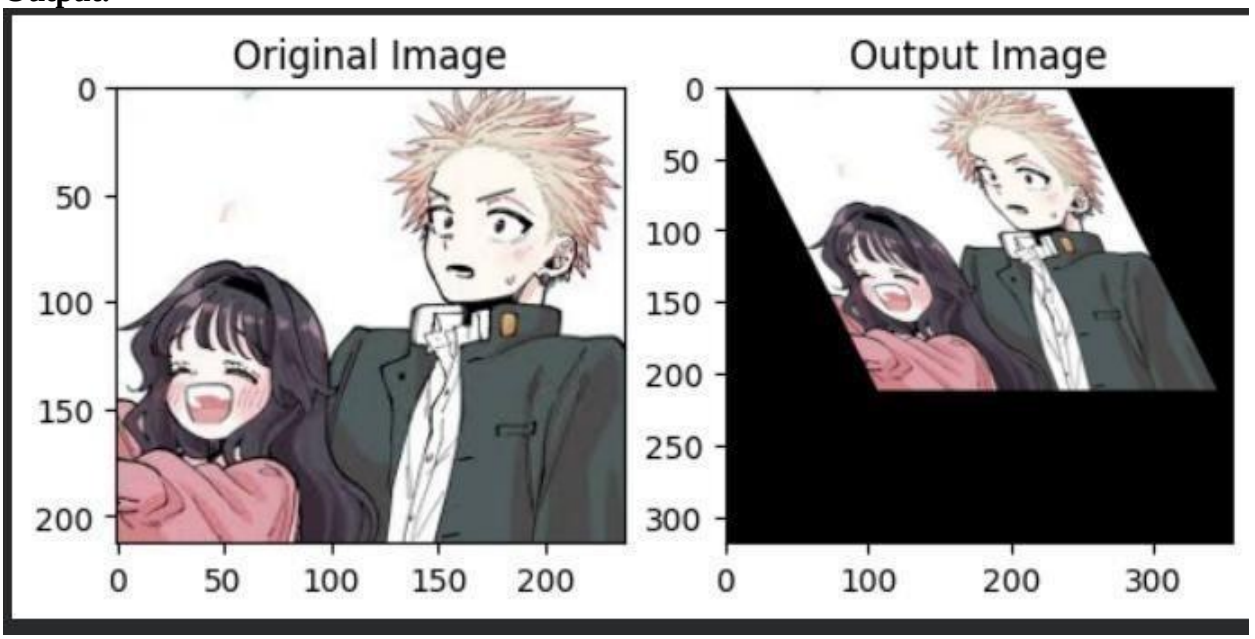
Shearing deals with changing the shape and size of the 2D object along x-axis and y- axis. It is similar to sliding the layers in one direction to change the shape of the 2D object. It is an ideal technique to change the shape of an existing object in a two dimensional plane. In a two- dimensional plane, the object size can be changed along X direction as well as Y direction.

Shear:

In x shear, the y co-ordinates remain the same but the x co-ordinates changes.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
img = cv2.imread("/content/waguri.jpg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
rows, cols, channels = img_rgb.shape
M = np.float32([[1, 0.5, 0], [0, 1, 0], [0, 0, 1]])
dst = cv2.warpPerspective(img_rgb, M, (int(cols*1.5), int(rows*1.5)))
plt.subplot(121), plt.imshow(img_rgb), plt.title('Original Image')
plt.subplot(122), plt.imshow(dst), plt.title('Output Image')
plt.show()
```

Output:



AIM: G] Shearing Y-axis.

Theory:

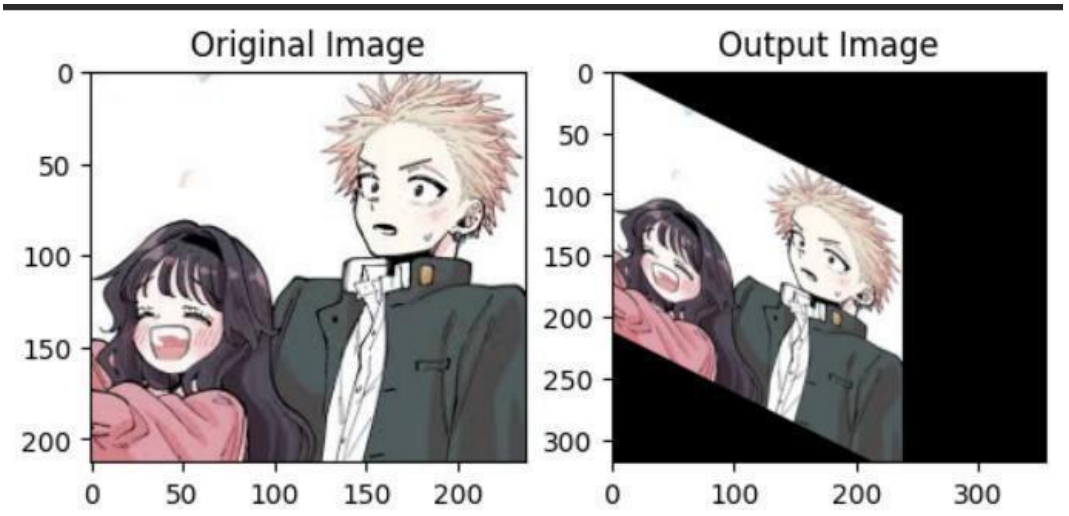
Shearing deals with changing the shape and size of the 2D object along x-axis and y- axis. It is similar to sliding the layers in one direction to change the shape of the 2D object.It is an ideal technique to change the shape of an existing object in a two dimensional plane. In a two- dimensional plane, the object size can be changed along X direction as well as Y direction.

Y-Shear:

In y shear, the x co-ordinates remain the same but the y co-ordinates changes.

```
import cv2

import matplotlib.pyplot as plt

import numpy as np

img = cv2.imread("/content/waguri.jpg")

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

rows, cols, channels = img_rgb.shape

M = np.float32([[1, 0, 0], [0.5, 1, 0], [0, 0, 1]])

dst = cv2.warpPerspective(img_rgb, M, (int(cols*1.5), int(rows*1.5)))

plt.subplot(121), plt.imshow(img_rgb), plt.title('Original Image')

plt.subplot(122), plt.imshow(dst), plt.title('Output Image')

plt.show()
```

Output:



AIM: H] Reflected Image.

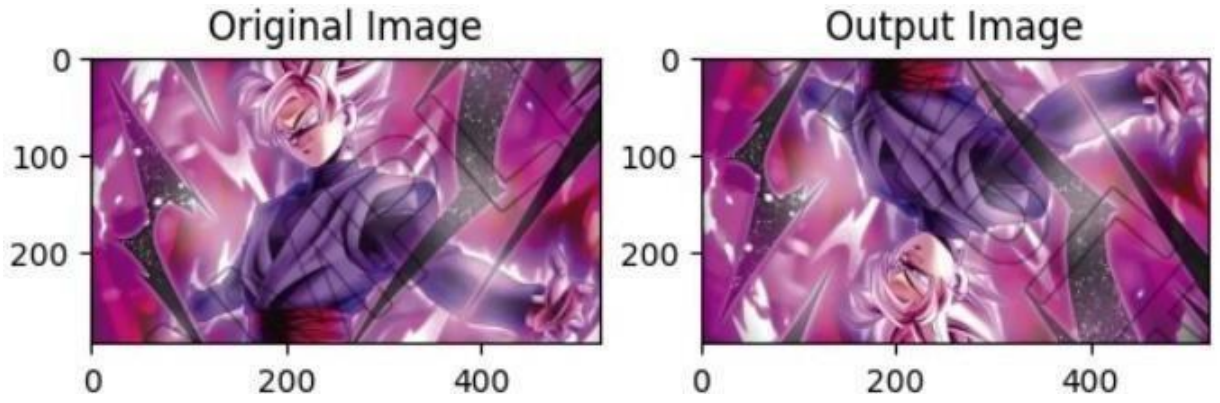
Theory

Image Reflection

Image reflection is used to flip the image vertically or horizontally. For reflection along the x-axis, we set the value of Sy to -1, Sx to 1, and vice-versa for the y-axis reflection.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
img = cv2.imread("/content/goku black.jpeg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
rows, cols, channels = img_rgb.shape
M = np.float32([[1, 0, 0], [0, -1, rows], [0, 0, 1]])
dst = cv2.warpPerspective(img_rgb, M, (cols, rows))
plt.subplot(121), plt.imshow(img_rgb), plt.title('Original Image')
plt.subplot(122), plt.imshow(dst), plt.title('Output Image')
plt.show()
```

Output:



AIM: I] Cropped Image.

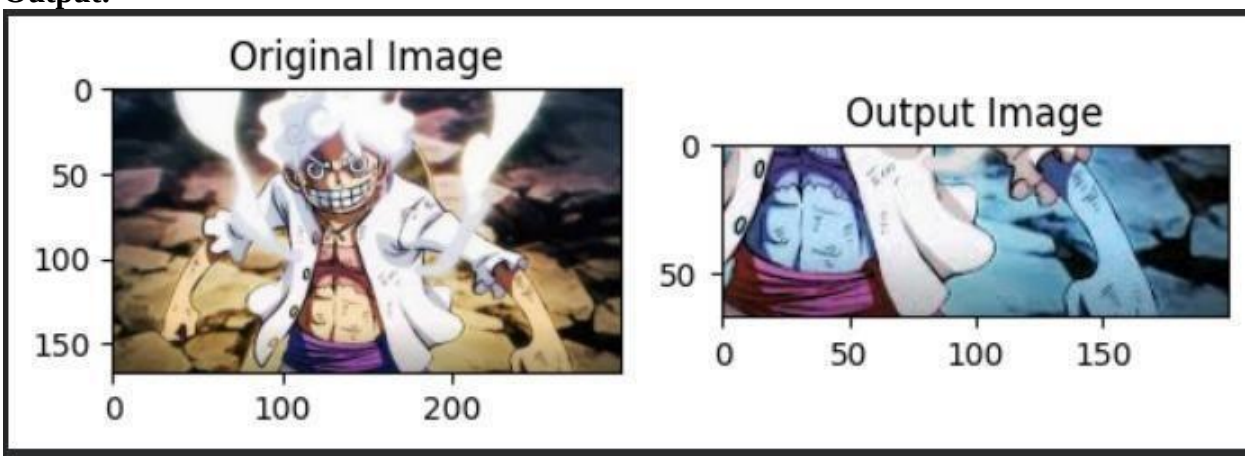
Theory:

Image Cropping

Cropping is the removal of unwanted outer areas from an image.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
img = cv2.imread("/content/luffy.jpg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
rows, cols, channels = img_rgb.shape
dst = img[100:300, 100:300]
plt.subplot(121), plt.imshow(img_rgb), plt.title('Original Image')
plt.subplot(122), plt.imshow(dst), plt.title('Output Image')
plt.show()
```

Output:



PRACTICAL NO: 02

AIM: Perform Image Stitching.

Theory:

Image stitching is the process of combining multiple overlapping images to create a seamless, high-resolution output image. This technique is commonly used to create panoramic images, virtual tours, and even some medical imaging applications.

Image stitching involves several steps:

Feature detection: Identifying and extracting unique features (e.g., corners, edges) from each input image. Compute the SIFT-key points and descriptors for both the images.

Feature matching: Finding correspondences between features in the overlapping regions of the input images. Compute distances between every descriptor in one image and every descriptor in the other image. Select the top _m' matches for each descriptor of an image.

Homography estimation: Estimating the transformation (e.g., rotation, scaling, translation) that aligns the input images. Run RANSAC to estimate homography

Warping: Applying the estimated transformation to the input images. Warp to align for stitching

Blending: Combining the warped images into a single seamless output image. Now stitch them together

Explanation of Code:

Firstly, we have to find out the features matching in both the images. These best matched features act as the basis for stitching. We extract the key points and sift descriptors for both the images as follows:

sift = cv2.SIFT_create()

find the keypoints and descriptors with SIFT kp1, des1 = sift.detectAndCompute(img1,None) kp2, des2 = sift.detectAndCompute(img2,None)

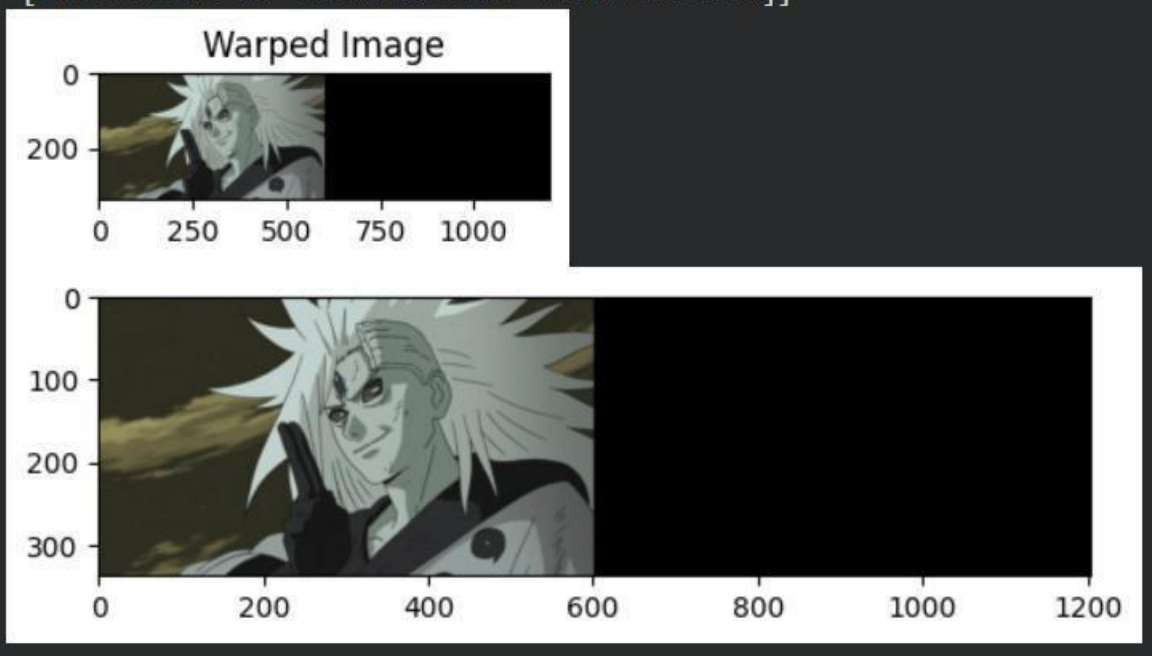
kp1 and kp2 are keypoints, des1 and des2 are the descriptors of the respective images.Now, the obtained descriptors in one image are to be recognized in the image too. We do that as follows:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
# Load and convert the first image to grayscale
img = cv2.imread("/content/jyubi.jpg")
img1_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Load and convert the second image to grayscale
img2 = cv2.imread("/content/jyubi.jpg")
img2_gray = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
# Create SIFT detector
sift = cv2.SIFT_create()
# Detect and compute keypoints and descriptors
kp1, des1 = sift.detectAndCompute(img1_gray, None)
kp2, des2 = sift.detectAndCompute(img2_gray, None)
# Use BFMatcher to find matches
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1, des2, k=2)
# Apply ratio test
good = []
# Try with a more relaxed ratio
for m, n in matches:
    if m.distance < 0.75 * n.distance:
```

```
            good.append(m)
        print("Good matches found:", len(good))
        if len(good) >= 4:
            src = np.float32([kp1[m.queryIdx].pt for m in good]).reshape(-1, 1, 2)
            dst = np.float32([kp2[m.trainIdx].pt for m in good]).reshape(-1, 1, 2)
            H, masked = cv2.findHomography(src, dst, cv2.RANSAC, 5.0)
            print(H)
            # Warp the first image onto the second image plane
            dst_img = cv2.warpPerspective(img, H, (img.shape[1] + img2.shape[1], img.shape[0]))
            plt.subplot(122)
            plt.imshow(dst_img)
            plt.title("Warped Image")
            plt.show()
            plt.figure()
            dst_img[0:img.shape[0], 0:img.shape[1]] = img
            cv2.imwrite("resultant_stitched_panorama.jpg", dst_img)
            plt.imshow(dst_img)
            plt.show()
        else:
            print("Not enough matches found.")
            img_matches = cv2.drawMatches(img, kp1, img2, kp2, good, None, flags=2)
            plt.imshow(cv2.cvtColor(img_matches, cv2.COLOR_BGR2RGB))
            plt.title("Matched Keypoints")
            plt.axis('off')
    plt.show()
```

Output:

```
Good matches found: 456
[[ 1.00000000e+00  5.58294032e-16 -3.19529216e-13]
 [ 4.45396607e-16  1.00000000e+00 -1.07040970e-13]
 [ 2.50759348e-18  1.63004613e-18  1.00000000e+00]]
```



PRACTICAL NO: 03

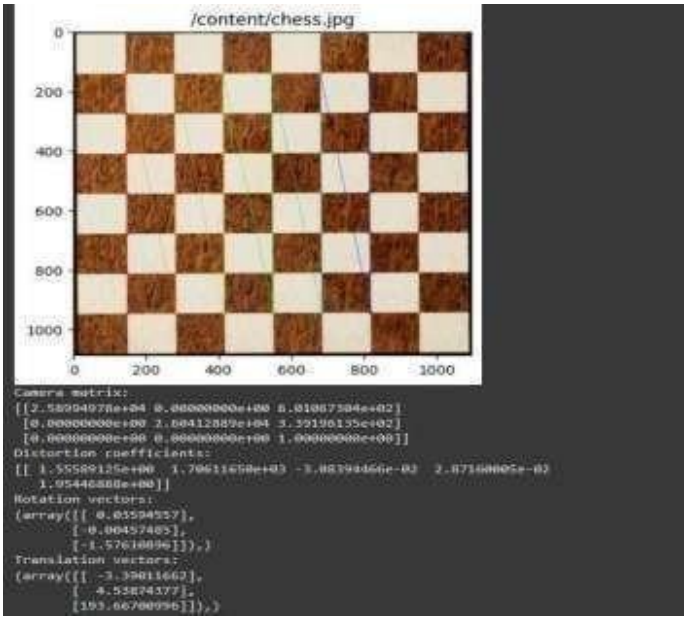
AIM: Perform Camera Calibration.

Theory:

A camera is an integral part of several domains like robotics, space exploration, etc camera is playing a major role. It helps to capture each and every moment and helpful for many analyses. In order to use the camera as a visual sensor, we should know the parameters of the camera. Camera Calibration is nothing but estimating the parameters of a camera, parametersabout the camera are required to determine an accurate relationship between a3D point in the real world and its corresponding 2D projection (pixel) in the image captured by that calibrated camera.

```
import numpy as np
import cv2 as cv import glob
import matplotlib.pyplot as plt
criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 30, 0.001)
objp = np.zeros((6*7, 3), np.float32)
objp[:, :2] = np.mgrid[0:7, 0:6].T.reshape(-1, 2)
objpoints = [] imgpoints = []
file images = glob.glob('/content/chess.jpg') for fname in images:
img = cv.imread(fname) if img is None:
corners2 = cv.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)
imgpoints.append(corners2)
cv.drawChessboardCorners(img, (7, 6), corners2, ret)
# Display the image with chessboard corners using matplotlib plt.imshow(cv.cvtColor(img,
cv.COLOR_BGR2RGB)) plt.title(fname) plt.show()else:
print(f"Chessboard corners not found in image {fname}")
empty if len(objpoints) > 0 and len(imgpoints)0:
ret, mtx, dist, rvecs, tvecs = cv.calibrateCamera(objpoints, imgpoints, gray.shape[::-1], None,
None)
print("Camera matrix: ")
print(mtx) print("Distortion
```

Output:



PRACTICAL NO: 04

AIM: A] Perform the following Face detection.

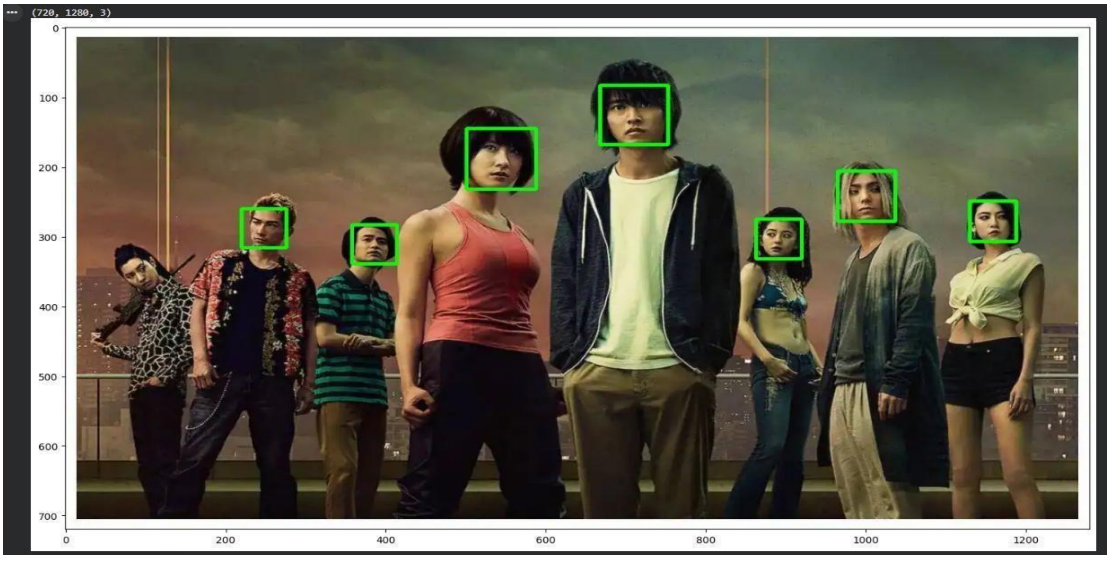
Theory:

Face detection involves identifying a person‘s face in an image or video. Thisis done by analyzing the visual input to determine whether a person‘s facial features are present. Since human faces are so diverse, face detection models typically need to be trained on large amounts of input data for them to be accurate. The training dataset must contain a sufficient representation of people who come from different backgrounds, genders, and cultures.These algorithms also need to be fed many training samples comprising different lighting, angles, and orientations to make correct predictions in real- world scenarios.These nuances make face detection a non-trivial, time- consuming task thatrequires hours of model training and millions of data samples.

face detection

```
import cv2
import matplotlib.pyplot as plt
imagePath = '/content/arisu.jpg'
img = cv2.imread(imagePath)
print(img.shape)
gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
face_classifier = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")
face = face_classifier.detectMultiScale(gray_image, scaleFactor=1.1, minNeighbors=5,
minSize=(40, 40))
for (x, y, w, h) in face:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 4)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.figure(figsize=(20, 10))
plt.imshow(img_rgb)
plt.show()
```

Output:



AIM: B] Perform the following Object detection.

Theory

Object Detection

- Haar cascade:Basically, the Haar cascade technique is an approach based on machine learning where we use a lot of positive and negative images to train the classifier to classify between the images. Haar cascade classifiers are considered as the effective way to do object detection with the OpenCV library.
- Positive images: These are the images that contain the objects which we want to be identified from the classifier.
- Negative Images: These are the images that do not contain any object that we want to be detected by the classifier, and these can be images of everything else
-

```
import cv2
from matplotlib import pyplot as plt
imaging = cv2.imread("/content/Stop.jpg")
imaging_gray = cv2.cvtColor(imaging, cv2.COLOR_BGR2GRAY)
imaging_rgb = cv2.cvtColor(imaging, cv2.COLOR_BGR2RGB)
xml_data = cv2.CascadeClassifier('/content/stop_sign_classifier_2.xml')
detecting = xml_data.detectMultiScale(imaging_gray, minSize=(30, 30))
amountDetecting = len(detecting)
if amountDetecting != 0:
    for (a, b, width, height) in detecting:
        cv2.rectangle(imaging_rgb, (a, b),
                      (a + height, b + width),
                      (0, 275, 0), 9)
plt.imshow(imaging_rgb)
plt.show()
```

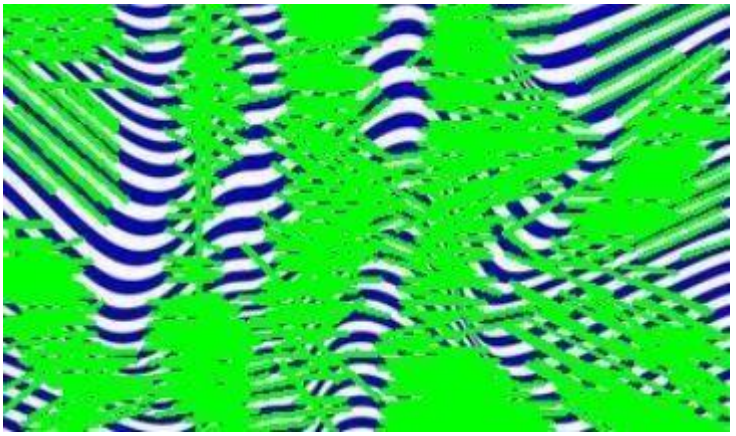
Output:

AIM: C] Perform the following Line detection.

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
image = cv2.imread('/content/Line Image.jpg')
cv2_imshow(image)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 50, 150, apertureSize=3)
line_list = []
lines = cv2.HoughLinesP(
   edges,
   rho = 2,
   theta = np.pi / 180,
   threshold = 100,
   minLineLength = 20,
   maxLineGap = 5
)
for points in lines:
   x1, y1, x2, y2 = points[0]
   cv2.line(image, (x1, y1), (x2, y2), (0, 255, 0), 2)
line_list.append([(x1, y1), (x2, y2)])
```

Input image:

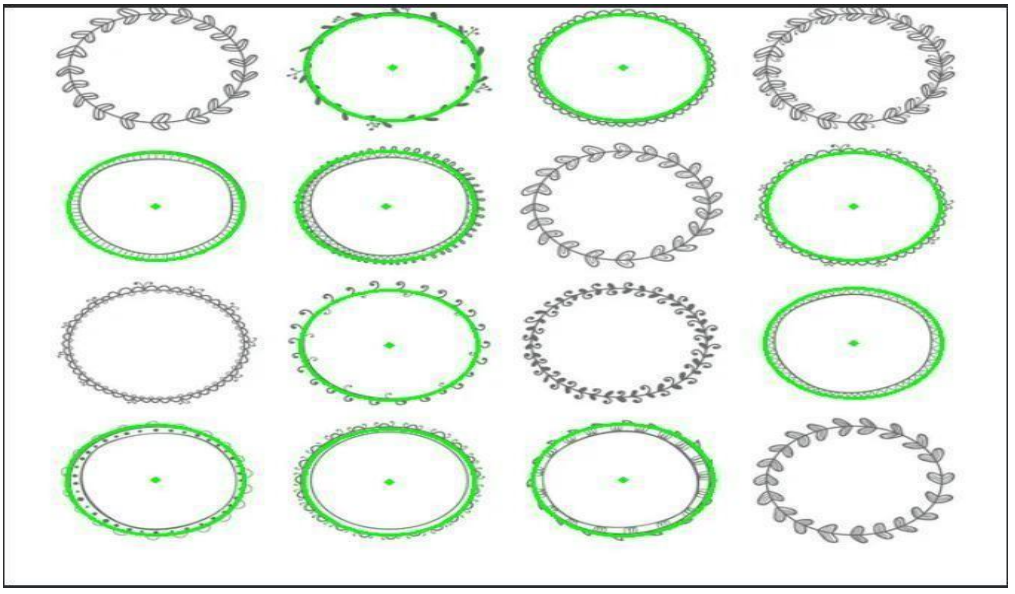
Output:



AIM: D] Perform the following Hough transform.

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
img = cv2.imread('/content/Circle.jpg')
cv2_imshow(img)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray_blurred = cv2.blur(gray, (3,3))
detected_circles = cv2.HoughCircles(gray_blurred,
                                    cv2.HOUGH_GRADIENT,
                                    1, 20,
                                    param1=100,
                                    param2=50,
                                    minRadius=2,
                                    maxRadius=80)
if detected_circles is not None:
    detected_circles = np.uint16(np.around(detected_circles))
    for pt in detected_circles[0, :]:
        a, b, r = pt[0], pt[1], pt[2]
        cv2.circle(img, (a, b), r, (0, 255, 0), 2)
        cv2.circle(img, (a, b), 1, (0, 255, 0), 3)
```

Output:



AIM: E] Perform the following Pedestrian detection.

```
import cv2
import imutils
from google.colab.patches import cv2_imshow
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
image = cv2.imread('/content/Public.jpg')
image = imutils.resize(image, width=min(400, image.shape[1]))
(regions, _) = hog.detectMultiScale(image, winStride=(4, 4), padding=(4, 4), scale=1.05)
for (x, y, w, h) in regions:
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)
cv2_imshow(image)
```

Output:

AIM: F] Perform the following Face Recognition.

```
import cv2
from google.colab.patches import cv2_imshow
img = cv2.imread('/content/pavan.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=8,
minSize=(60, 60))
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 255), 2)
font = cv2.FONT_HERSHEY_SIMPLEX
text = 'I am donquixote doflamingo'
font_scale = 1
font_thickness = 1
text_size, _ = cv2.getTextSize(text, font, font_scale, font_thickness)
text_x = 20
text_y = 30 + text_size[1]
cv2.putText(img, text, (text_x, text_y), font, font_scale, (0, 0, 0), font_thickness,
cv2.LINE_AA)
scale_percent = 20
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
resized_img = cv2.resize(img, dim, interpolation=cv2.INTER_AREA)
cv2_imshow(resized_img)
```

Output:

PRACTICAL NO: 05

AIM: A] Implement object detection and tracking from video

```
import cv2
from google.colab.patches import cv2_imshow from tracker import *
tracker = EuclideanDistTracker()
cap = cv2.VideoCapture("/content/highway_video.mp4")
object_detector = cv2.createBackgroundSubtractorMOG2(history=100, varThreshold=40)
while True:
ret, frame = cap.read()
if not ret:
print("End of video or cannot read the frame.")
break # Exit the loop if the video ends or if there's an issue height, width, channels =
frame.shape # Unpack all three values print(height, width)
roi = frame[340:720, 500:800] mask = object_detector.apply(roi)
_, mask = cv2.threshold(mask, 254, 255, cv2.THRESH_BINARY)
contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
detections = []
for cnt in contours:
area = cv2.contourArea(cnt) if area > 100:
x, y, w, h = cv2.boundingRect(cnt)
cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 2) detections.append([x, y, w, h])
boxes_ids = tracker.update(detections) for box_id in boxes_ids:
x, y, w, h, id = box_id
cv2.putText(roi, str(id), (x, y - 15), cv2.FONT_HERSHEY_PLAIN, 1, (255, 0, 0), 2)
cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3) # Display images using cv2_imshow
cv2_imshow(roi) cv2_imshow(frame) cv2_imshow(mask) key = cv2.waitKey(30) if key ==
27:
break
cap.release() cv2.destroyAllWindows()
```

Output:

PRA

AIM: B] Implement object detection and tracking from video. (Count number of Faces using Python)

```
import cv2
import numpy as np
import dlib
from google.colab.patches import cv2_imshow
cap = cv2.VideoCapture("faces.mp4")
detector = dlib.get_frontal_face_detector()
while True:
    ret, frame = cap.read(1)
    if not ret:
        print("End of video stream.")
        break
    frame = cv2.flip(frame, 1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(gray)
    i = 0
    for face in faces:
        x, y = face.left(), face.top()
        x1, y1 = face.right(), face.bottom()
        cv2.rectangle(frame, (x, y), (x1, y1), (0, 255, 0), 2)
        i = i+1
        cv2.putText(frame, 'face num'+str(i), (x-10, y-10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 3)
        print(face, i)
    frame = cv2.resize(frame, (1000, 600))
```

```
    cv2_imshow(frame)

    # Removed cv2.waitKey and cv2.destroyAllWindows as they are not compatible with
cv2_imshow

    # if cv2.waitKey(1) & 0xFF == ord('q'):

    #     break

cap.release()

# cv2.destroyAllWindows()
```

Output:

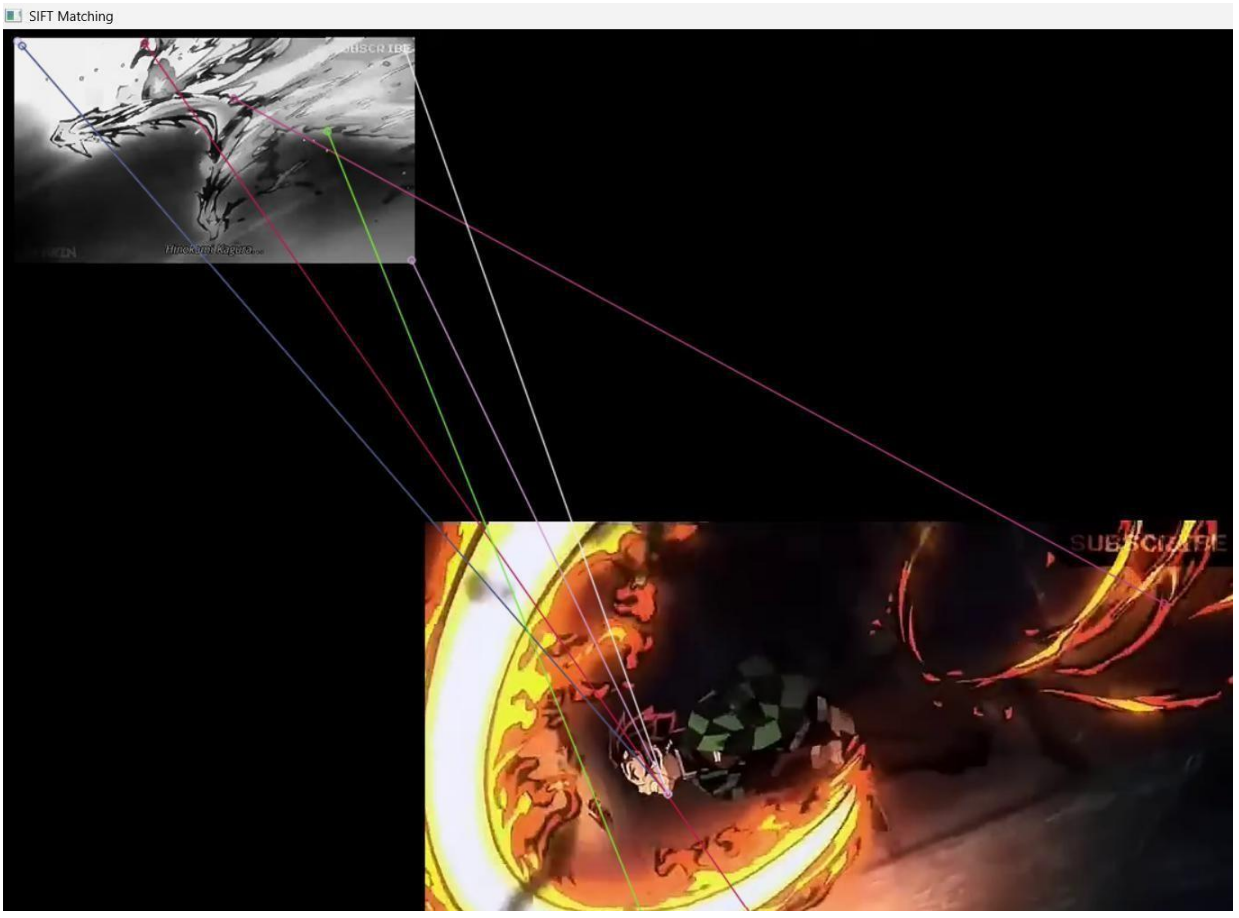
```
[(221, 85) (293, 157)] 1
[(74, 93) (160, 179)] 2
[(405, 45) (477, 117)] 3
```

AIM: C] Implement object detection and tracking from video. (Object Tracking using Homography)

```
import cv2
import numpy as np
# ====== PATHS (CHANGE THESE) ======
img_path = "img.jpg"        # your image file
video_path = "video.mp4"     # your video file
# Read image
img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
if img is None:
    print("+ Image not found!")
    exit()
# Open video
cap = cv2.VideoCapture(video_path)
if not cap.isOpened():
    print("+ Video not found!")
    exit()
# Create SIFT detector
sift = cv2.SIFT_create()
# Detect keypoints & descriptors in image
kp_image, desc_image = sift.detectAndCompute(img, None)
# FLANN matcher
index_params = dict(algorithm=1, trees=5)  # algorithm=1 for KDTree
search_params = dict()
flann = cv2.FlannBasedMatcher(index_params, search_params)
while True:
    ret, frame = cap.read()
    if not ret:
        p r i n t ( "  Video
        finished") break
    grayframe = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    kp_frame, desc_frame = sift.detectAndCompute(grayframe, None)
    if desc_image is not None and desc_frame is not None and len(desc_image) > 0
and len(desc_frame) > 0:

        matches = flann.knnMatch(desc_image, desc_frame, k=2)
        good_points = []
        for m, n in matches:
            if m.distance < 0.75 * n.distance:
                good_points.append(m)
        if len(good_points) > 10:
            query_pts = np.float32([kp_image[m.queryIdx].pt for m in
good_points]).reshape(-1, 1, 2)
            train_pts = np.float32([kp_frame[m.trainIdx].pt for m in
good_points]).reshape(-1, 1, 2)
            matrix, mask = cv2.findHomography(query_pts, train_pts, cv2.RANSAC,
5.0)
            if matrix is not None:
                h, w = img.shape
```

```
                pts = np.float32([[0, 0], [0, h], [w, h], [w, 0]]).reshape(-1, 1, 2)
                dst = cv2.perspectiveTransform(pts, matrix)
                frame = cv2.polylines(frame, [np.int32(dst)], True, (0, 255, 0), 3)
            result = cv2.drawMatches(img, kp_image, frame, kp_frame, good_points, None,
        flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
            cv2.imshow("SIFT Matching", result)
          else:
            cv2.imshow("Frame", frame)
          if cv2.waitKey(1) & 0xFF == 27:  # ESC key to exit
            break
        cap.release()
cv2.destroyAllWindow
```

Output:



PRACTICAL NO: 06

AIM: Perform Colorization.

```
import numpy as np
import cv2
from cv2 import dnn

proto_file = 'colorization_deploy_v2.prototxt'
model_file = 'colorization_release_v2.caffemodel'
hull_pts = 'pts_in_hull.npy'
img_path = 'DBZ1.jpg'

net = dnn.readNetFromCaffe(proto_file,model_file)
kernel = np.load(hull_pts)

img = cv2.imread(img_path)
scaled = img.astype("float32") / 255.0
lab_img = cv2.cvtColor(scaled, cv2.COLOR_BGR2LAB)

class8 = net.getLayerId("class8_ab")
conv8 = net.getLayerId("conv8_313_rh")
pts = kernel.transpose().reshape(2, 313, 1, 1)
net.getLayer(class8).blobs = [pts.astype("float32")]
net.getLayer(conv8).blobs =[np.full([1, 313], 2.606, dtype = "float32")]

resized = cv2.resize(lab_img, (224, 224))

L = cv2.split(resized)[0]

L -= 50

net.setInput(cv2.dnn.blobFromImage(L))
ab_channel = net.forward()[0, :, :, :].transpose((1, 2, 0))
ab_channel = cv2.resize(ab_channel, (img.shape[1], img.shape[0]))
```

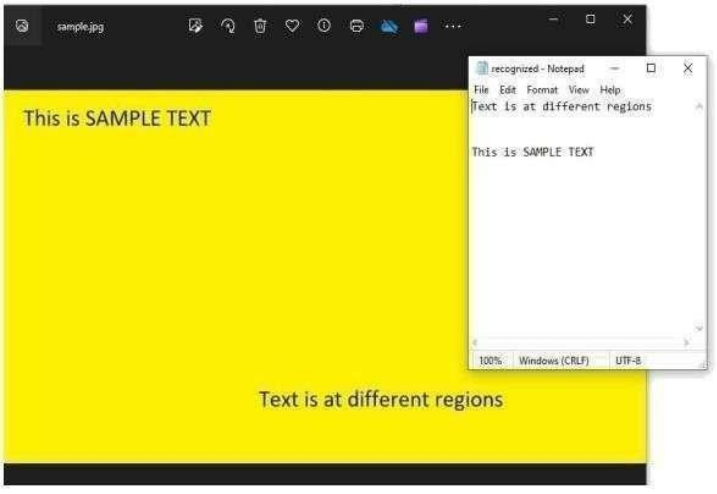
Output:

PRACTICAL NO: 07

AIM: Perform Text Detection and Recognition

```
import cv2
import pytesseract

pytesseract.pytesseract.tesseract_cmd = 'C:\\Program Files\\Tesseract-OCR\\tesseract.exe'

img = cv2.imread('sample.jpg')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

ret, thresh1 = cv2.threshold(gray, 0, 255, cv2.THRESH_OTSU | cv2.THRESH_BINARY_INV)

rect_kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(18,18))

dilation= cv2.dilate(thresh1, rect_kernel, iterations = 1)


contours, hierarchy = cv2.findContours(dilation,cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

im2 = img.copy()

file = open('recognized.txt','w+')
file.write("")
file.close()

for cnt in contours:
    x,y,w,h = cv2.boundingRect(cnt)
    rect = cv2.rectangle(im2, (x,y), (x+w,y+h), (0,255,0),2)
    cropped = im2[y:y+h,x:x+w]

    file = open('recognized.txt','a')
    text = pytesseract.image_to_string(cropped)
    file.write(text)
    file.write("\n")
```

Output

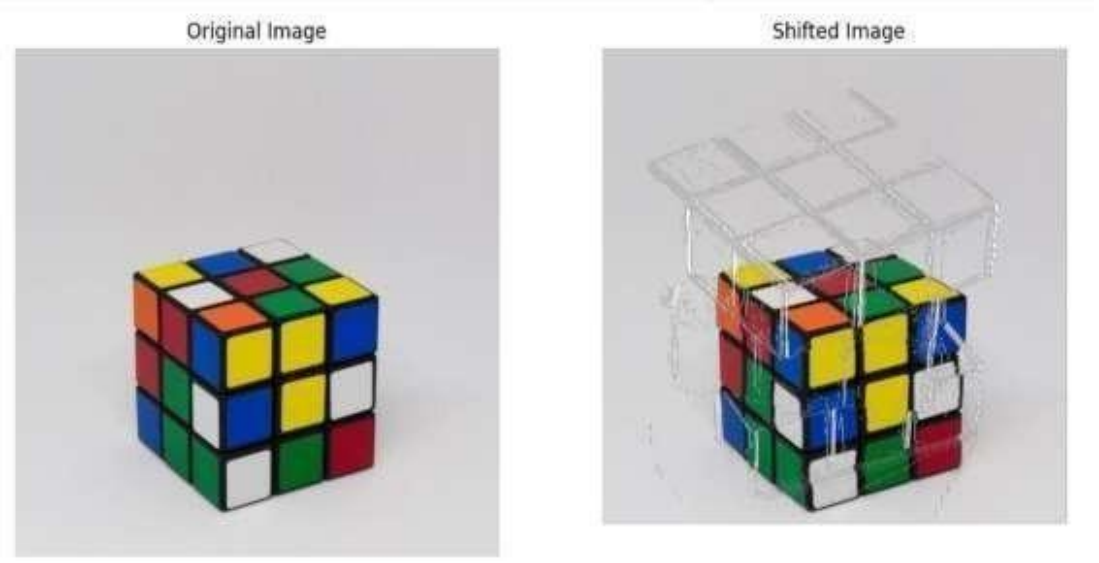


PRACTICAL NO: 08

AIM: Construct 3D model from Images.

```
from PIL import Image import numpy as np
import matplotlib.pyplot as plt

def shift_image(img, depth_img, shift_amount=10): img = img.convert("RGBA")
data = np.array(img)
depth_img = depth_img.convert("L") depth_data = np.array(depth_img)
deltas = ((depth_data / 255.0) * float(shift_amount)).astype(int)
shifted_data = np.zeros_like(data) height, width, _ = data.shape
for y, row in enumerate(deltas):
for x, dx in enumerate(row):
if 0 <= x + dx < width: shifted_data[y, x + dx] = data[y, x]
shifted_image = Image.fromarray(shifted_data.astype(np.uint8))
# Load images
img_path = "/content/sample_data/cube1.jpg" # Replace with your image path
depth_img_path = "/content/sample_data/cube2.jpg" # Replace with your depth image path
# Original Image axs[0].imshow(img) axs[0].set_title('Original Image') axs[0].axis('off')

# Shifted Image axs[1].imshow(shifted_img) axs[1].set_title('Shifted Image')
axs[1].axis('off')
plt.show()
```

Output



PRACTICAL NO: 09

AIM: Perform Feature extraction using RANSAC.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
img1_color = cv2.imread("/content/sample_data/cube1.jpg") img2_color =
cv2.imread("/content/sample_data/cube2.jpg") img1 = cv2.cvtColor(img1_color,
cv2.COLOR_BGR2GRAY) img2 = cv2.cvtColor(img2_color, cv2.COLOR_BGR2GRAY)
height, width = img2.shape
orb_detector = cv2.ORB_create(5000)
kp1, d1 = orb_detector.detectAndCompute(img1, None) kp2, d2 =
orb_detector.detectAndCompute(img2, None)
matcher = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True) matches =
matcher.match(d1, d2)
matches = sorted(matches, key=lambda x: x.distance) matches = matches[:int(len(matches) *
0.9)] no_of_matches = len(matches)
p1 = np.zeros((no_of_matches, 2)) p2 = np.zeros((no_of_matches, 2))
axs[0].set_title('Original Image')
axs[0].axis('off')
axs[1].imshow(cv2.cvtColor(transformed_img, cv2.COLOR_BGR2RGB))
axs[1].set_title('Transformed Image')
```

Output: