Auswärtiges Amt

DAAD Deutscher Akademischer Austauschdienst
German Academic Exchange Service

Geo-IT Online Seminar
Freie Universität Berlin
Institute of Geographical Sciences

Freie Universität Berlin

TRAINING ON GOOGLE EARTH ENGINE

MODULE 4 : IMAGE in Google Earth Engine

Prof. Dr. Eng. Ayman Abdulrahman

Coordinated from

Dr. Wahib Sahwan (Freie Universität Berlin)

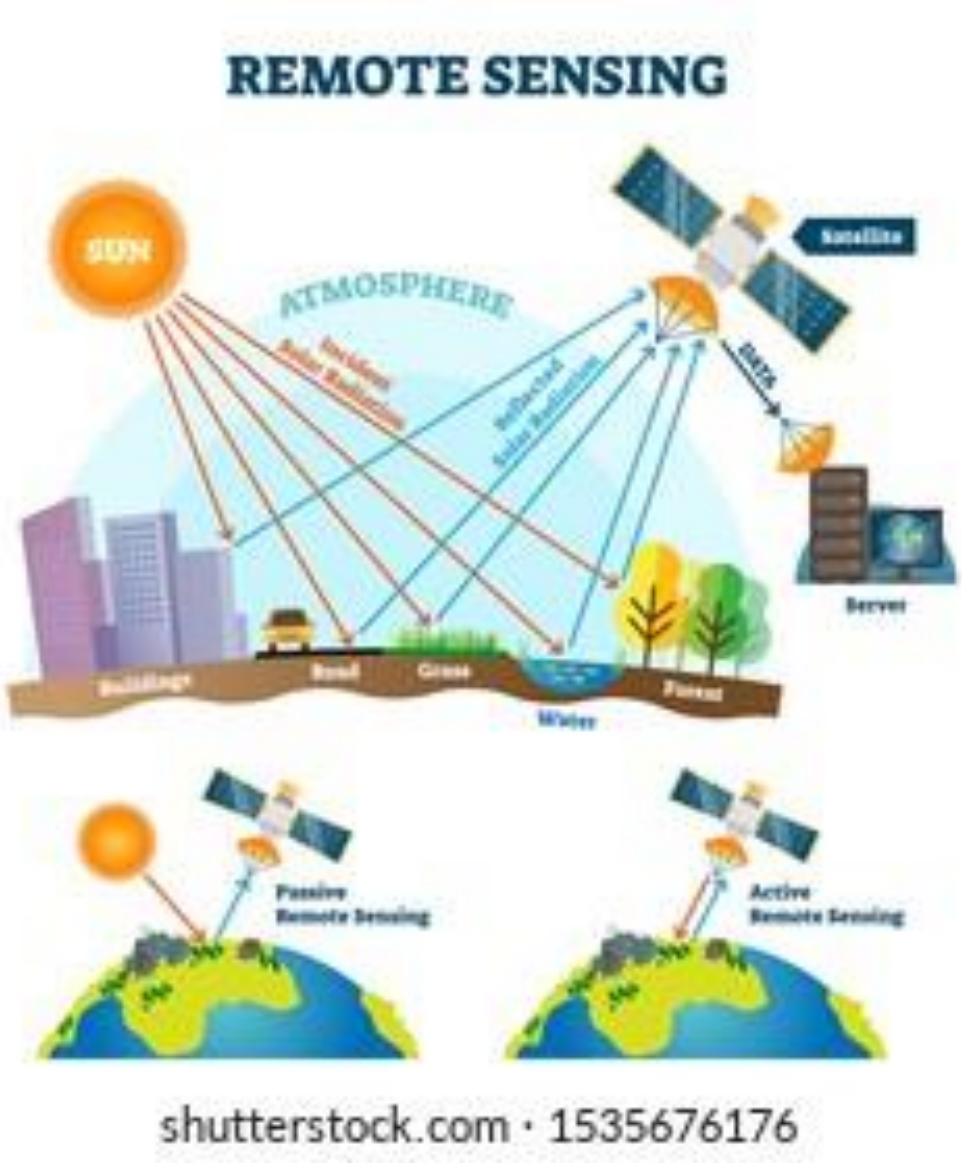
Geo-IT

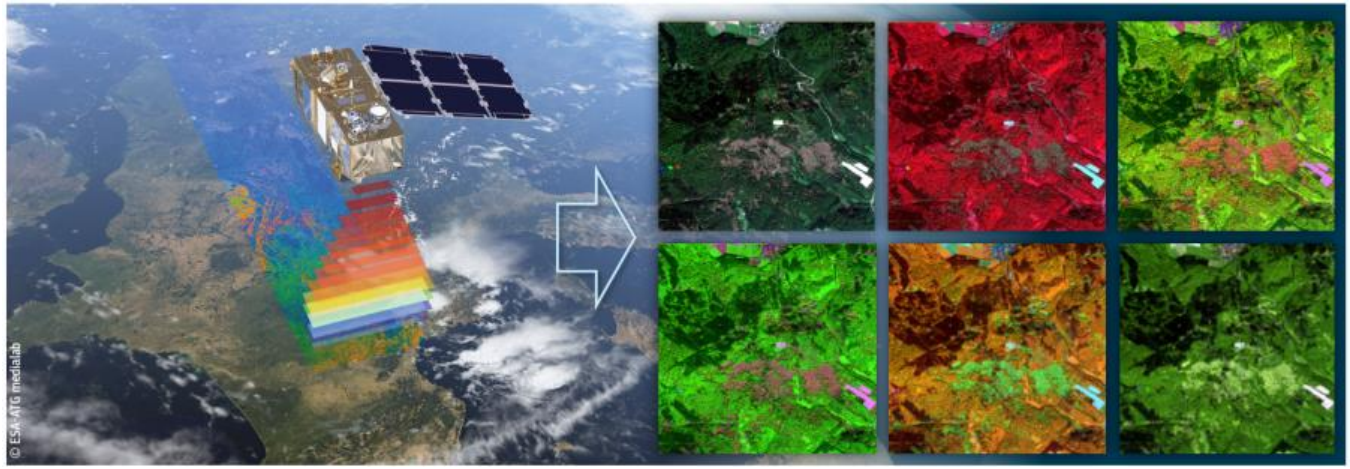
The Technology of Data Acquisition for Sustainable Development and Crisis Management

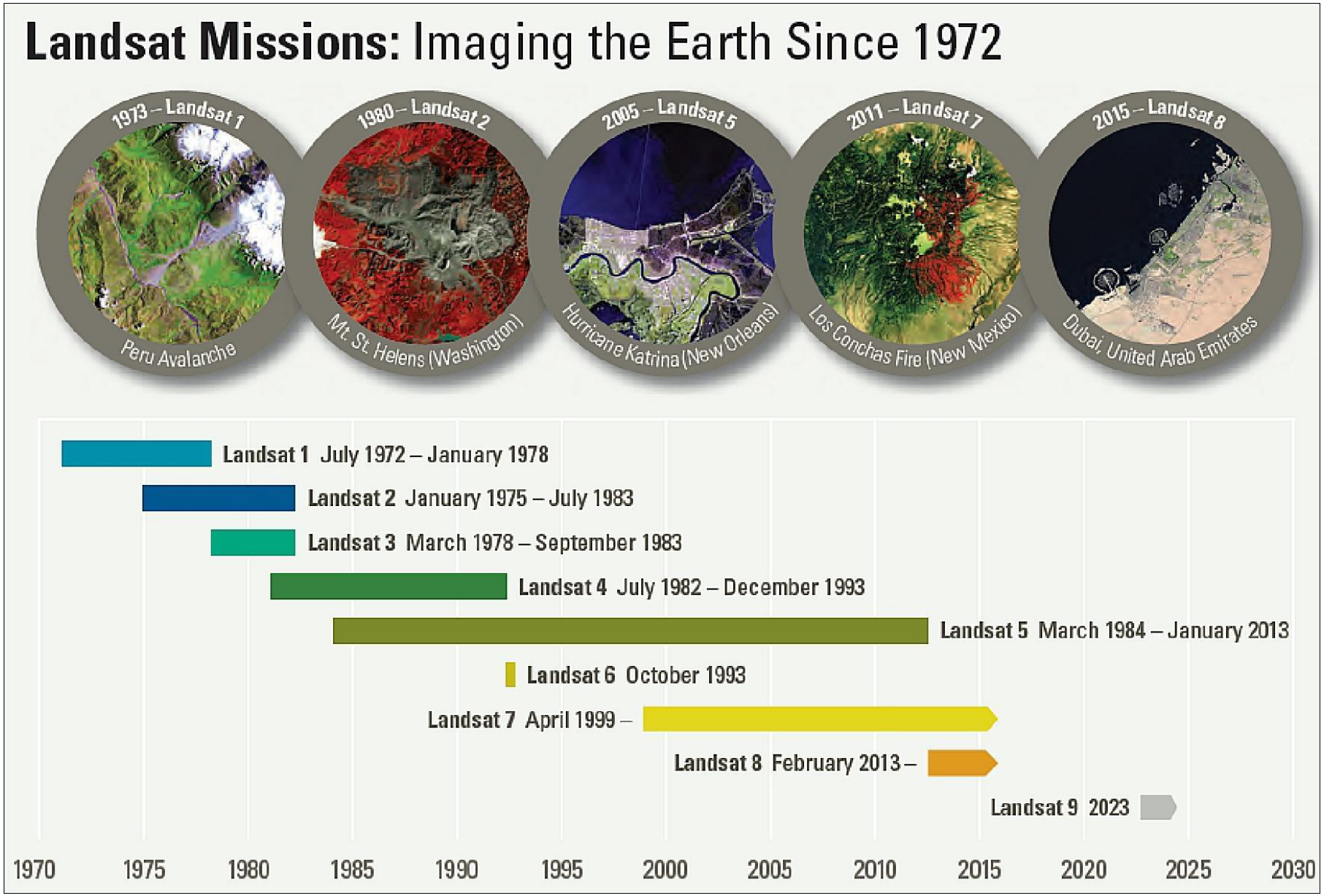
(Germany, Jordan, Lebanon and Syria)

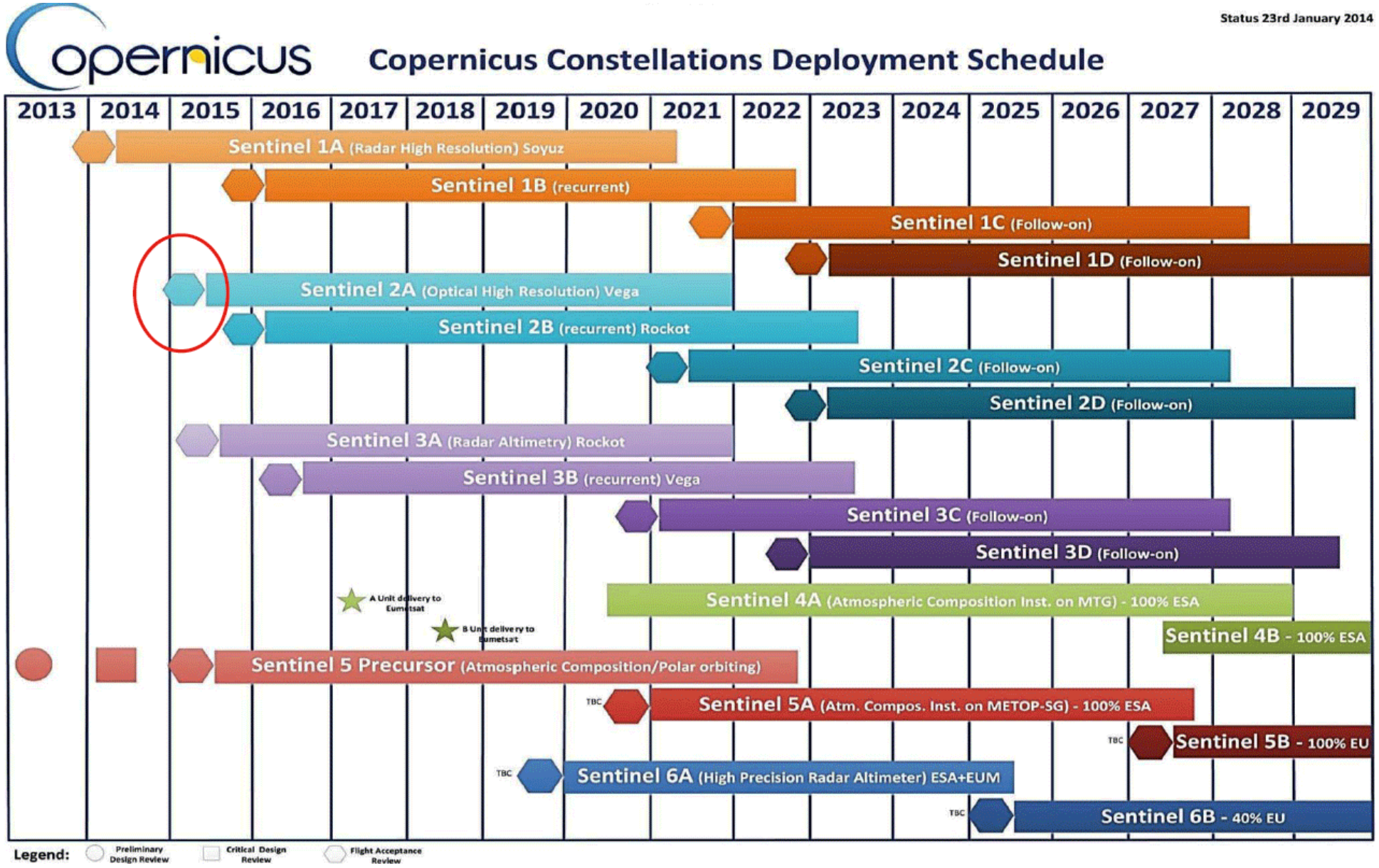
MODULE 4 : IMAGE in Google Earth Engine

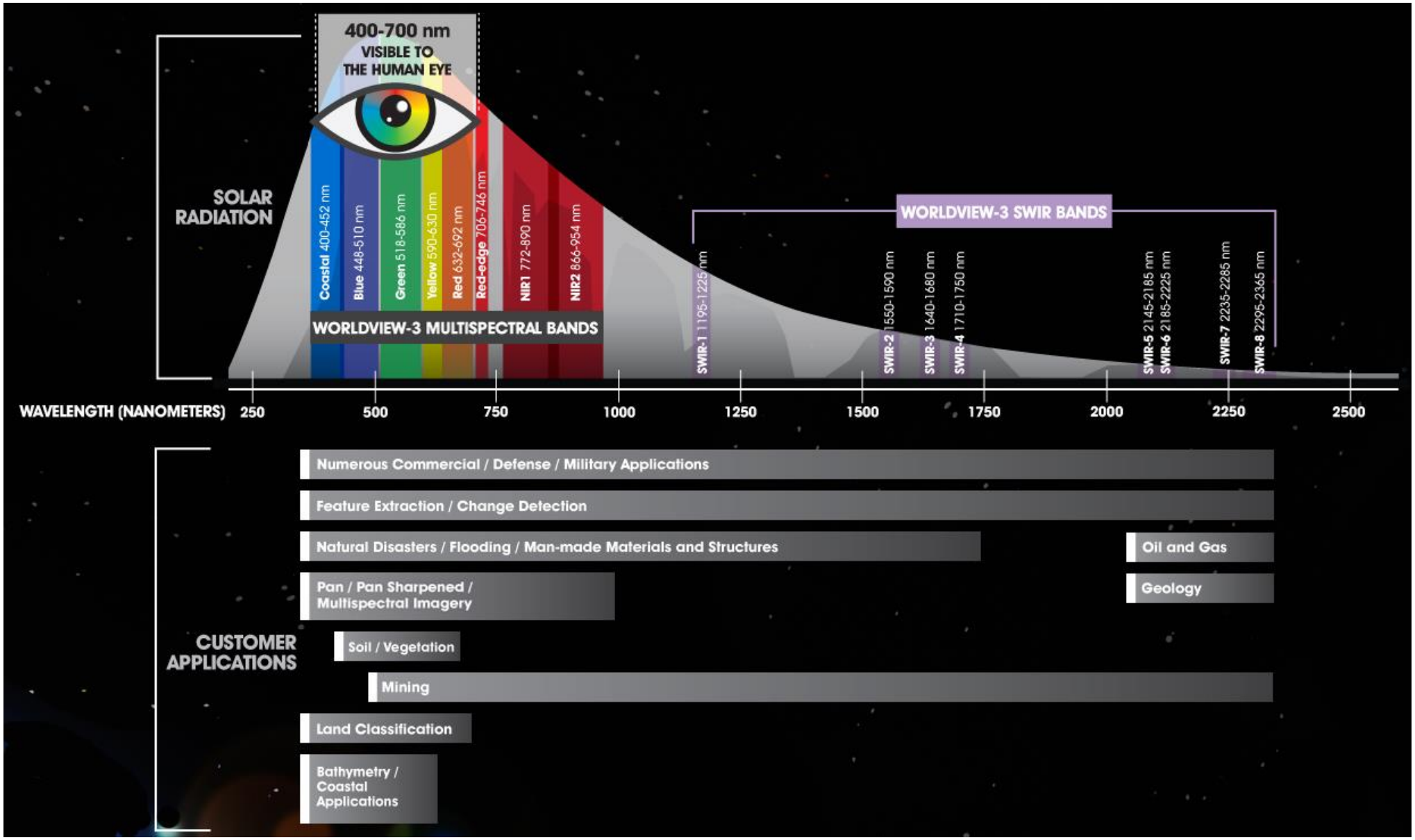
- *Define Image*
- *Import Image*
- *Image Properties*
- *Image Bands (Landsat, Sentinel,...)*
- *Mathematical Operations*
- *Reducing*
- *Masking*
- *Visualisation of Image*
- *Export of Image*
- *Charts*
- *Examples and exercises*

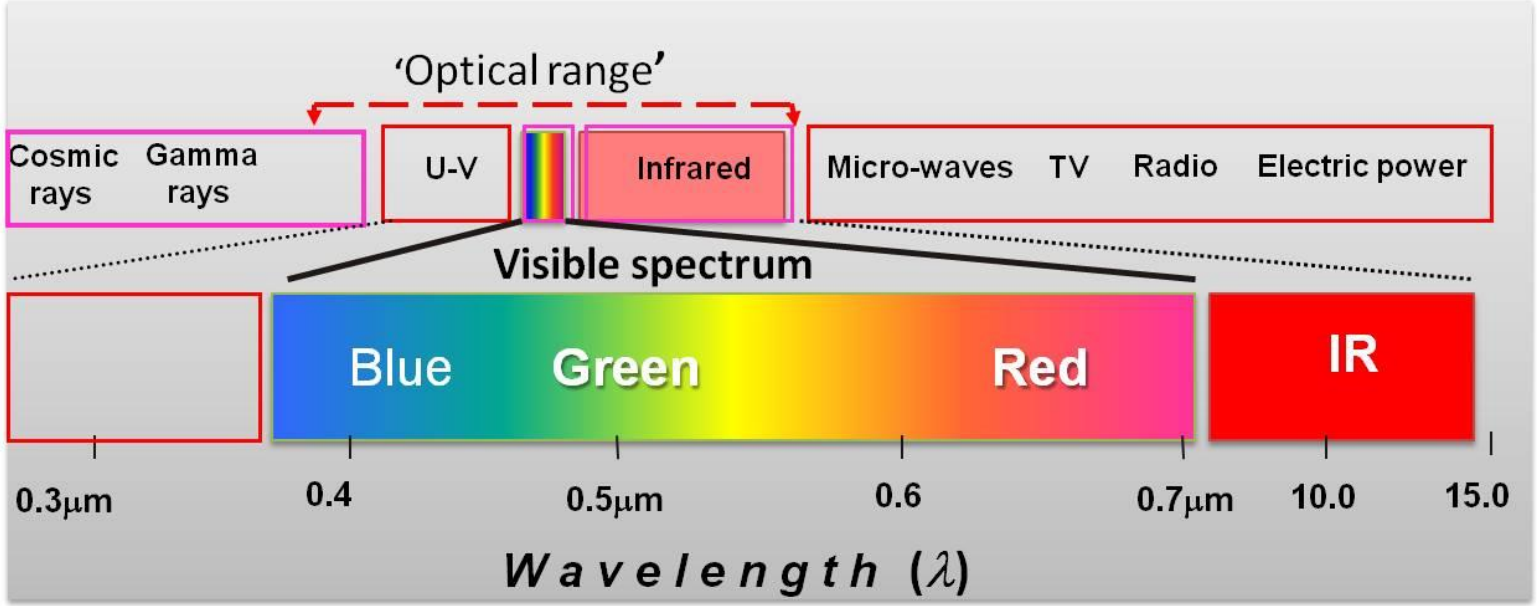

REMOTE SENSING
SUN
ATMOSPHERE
Satellite
DATA
Server
Passive
Remote Sensing
Active
Remote Sensing
shutterstock.com · 1535676176

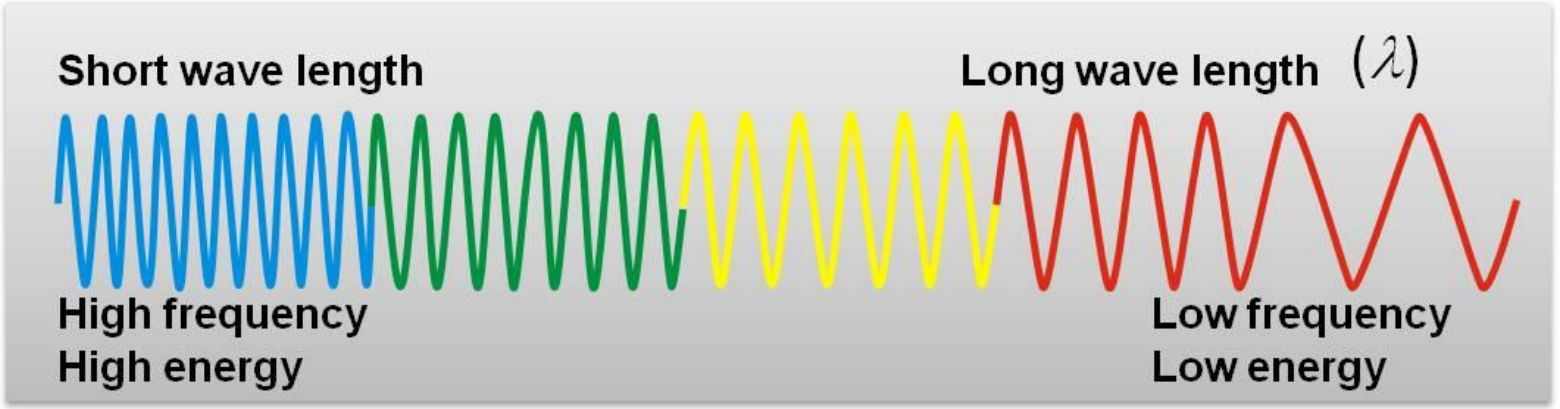

© ESA-ATG medialab

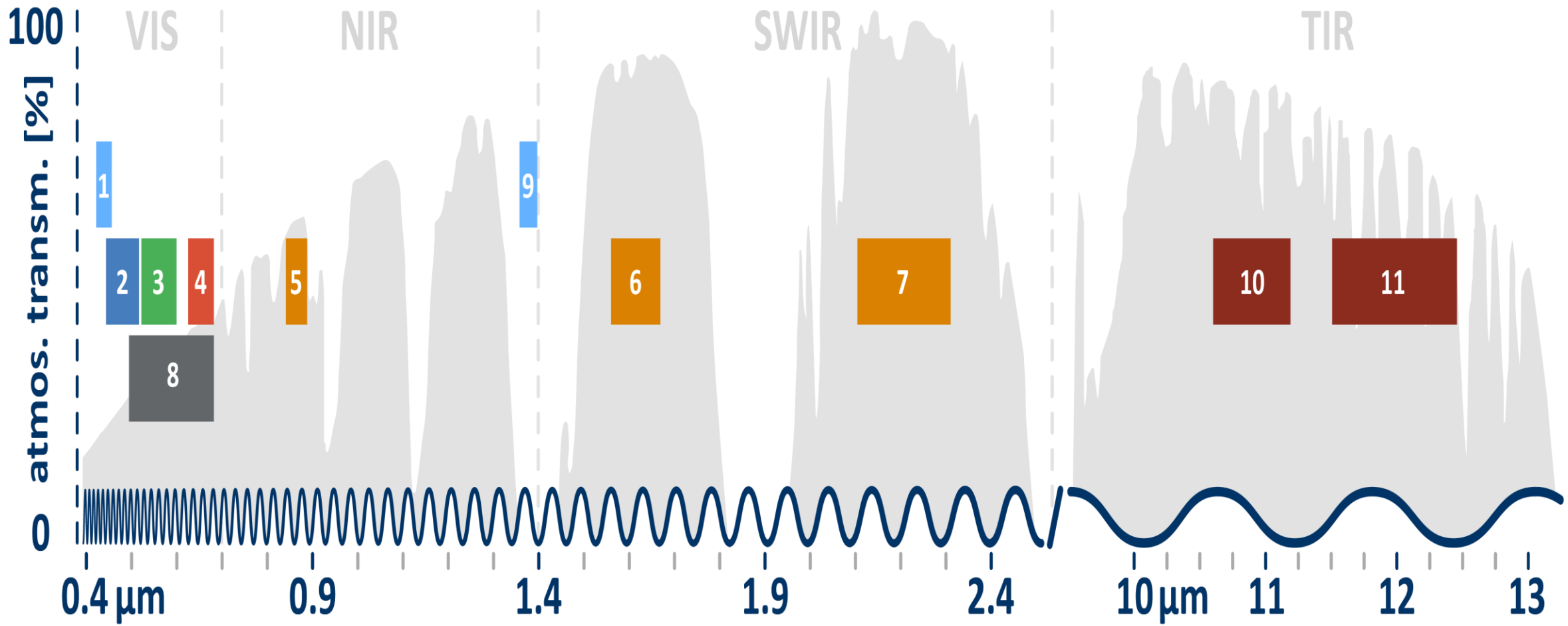

Landsat Missions: Imaging the Earth Since 1972
1973 – Landsat 1
Peru Avalanche
1980 – Landsat 2
Mt. St. Helens (Washington)
2005 – Landsat 5
Hurricane Katrina (New Orleans)
2011 – Landsat 7
Los Conchas Fire (New Mexico)
2015 – Landsat 8
Dubai, United Arab Emirates
Landsat 1 July 1972 – January 1978
Landsat 2 January 1975 – July 1983
Landsat 3 March 1978 – September 1983
Landsat 4 July 1982 – December 1993
Landsat 5 March 1984 – January 2013
Landsat 6 October 1993
Landsat 7 April 1999 –
Landsat 8 February 2013 –
Landsat 9 2023
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025
2030

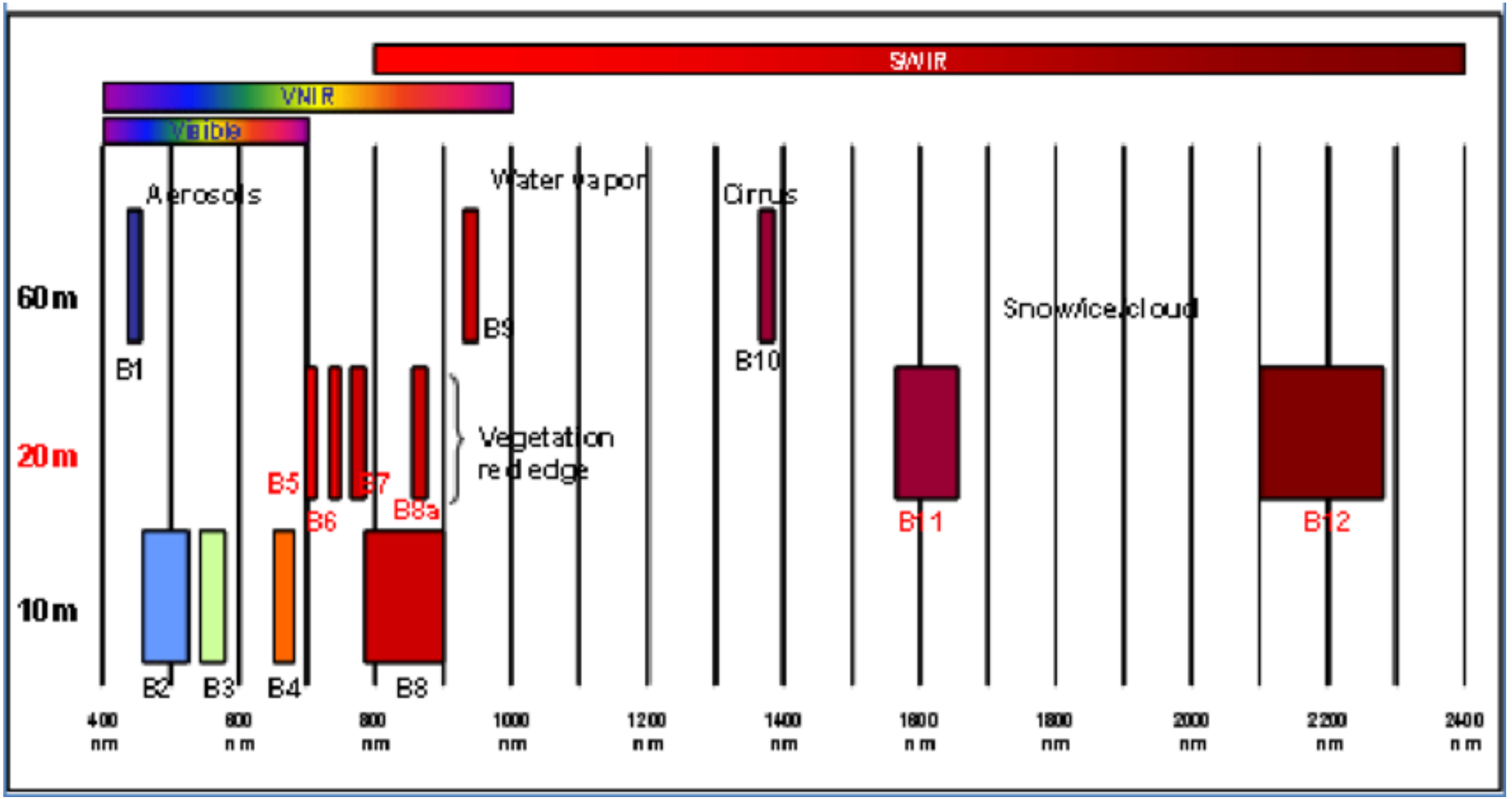

Status 23rd January 2014
Copernicus
Copernicus Constellations Deployment Schedule
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
Sentinel 1A (Radar High Resolution) Soyuz
Sentinel 1B (recurrent)
Sentinel 1C (Follow-on)
Sentinel 1D (Follow-on)
Sentinel 2A (Optical High Resolution) Vega
Sentinel 2B (recurrent) Rockot
Sentinel 2C (Follow-on)
Sentinel 2D (Follow-on)
Sentinel 3A (Radar Altimetry) Rockot
Sentinel 3B (recurrent) Vega
Sentinel 3C (Follow-on)
Sentinel 3D (Follow-on)
A Unit delivery to Eumetsat
B Unit delivery to Eumetsat
Sentinel 4A (Atmospheric Composition Inst. on MTG) - 100% ESA
Sentinel 4B - 100% ESA
Sentinel 5 Precursor (Atmospheric Composition/Polar orbiting)
TBC
Sentinel 5A (Atm. Compos. Inst. on METOP-SG) - 100% ESA
TBC
Sentinel 5B - 100% EU
TBC
Sentinel 6A (High Precision Radar Altimeter) ESA+EUM
TBC
Sentinel 6B - 40% EU
Legend:
Preliminary Design Review
Critical Design Review
Flight Acceptance Review

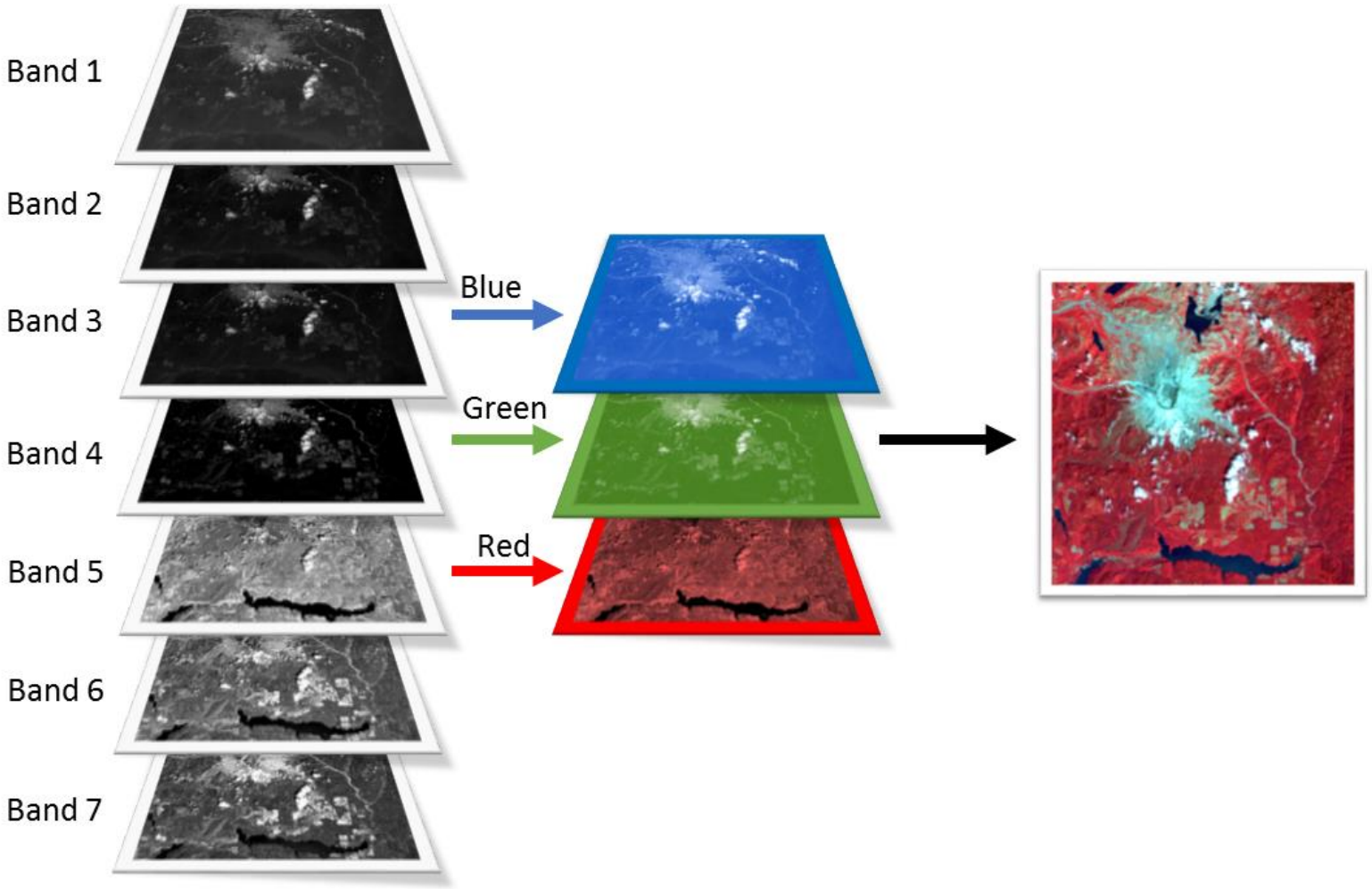

400-700 nm
VISIBLE TO THE HUMAN EYE
SOLAR RADIATION
Coastal 400-452 nm
Blue 448-510 nm
Green 518-586 nm
Yellow 590-630 nm
Red 632-692 nm
Red-edge 706-746 nm
NIR1 772-890 nm
NIR2 866-954 nm
WORLDVIEW-3 MULTISPECTRAL BANDS
WORLDVIEW-3 SWIR BANDS
SWIR-1 1195-1225 nm
SWIR-2 1550-1590 nm
SWIR-3 1640-1680 nm
SWIR-4 1710-1750 nm
SWIR-5 2145-2185 nm
SWIR-6 2185-2225 nm
SWIR-7 2235-2285 nm
SWIR-8 2295-2365 nm
WAVELENGTH (NANOMETERS) 250 500 750 1000 1250 1500 1750 2000 2250 2500
CUSTOMER APPLICATIONS
Numerous Commercial / Defense / Military Applications
Feature Extraction / Change Detection
Natural Disasters / Flooding / Man-made Materials and Structures
Pan / Pan Sharpened / Multispectral Imagery
Soil / Vegetation
Mining
Land Classification
Bathymetry / Coastal Applications
Oil and Gas
Geology


'Optical range'
Cosmic rays
Gamma rays
U-V
Infrared
Micro-waves
TV
Radio
Electric power
Visible spectrum
Blue
Green
Red
IR
0.3μm
0.4
0.5μm
0.6
0.7μm
10.0
15.0
Wavelength (λ)


Short wave length
Long wave length (λ)
High frequency
High energy
Low frequency
Low energy


100
0
atmos. transm. [%]
VIS
NIR
SWIR
TIR
1
2
3
4
5
6
7
8
9
10
11
0.4 µm
0.9
1.4
1.9
2.4
10 µm
11
12
13


SWIR
VNIR
Visible
Aerosols
Water vapor
Cirrus
Snow/ice/cloud
60 m
20 m
10 m
B1
B2
B3
B4
B5
B6
B7
B8
B8a
B9
B10
B11
B12
Vegetation red edge
400 nm
600 nm
800 nm
1000 nm
1200 nm
1400 nm
1600 nm
1800 nm
2000 nm
2200 nm
2400 nm

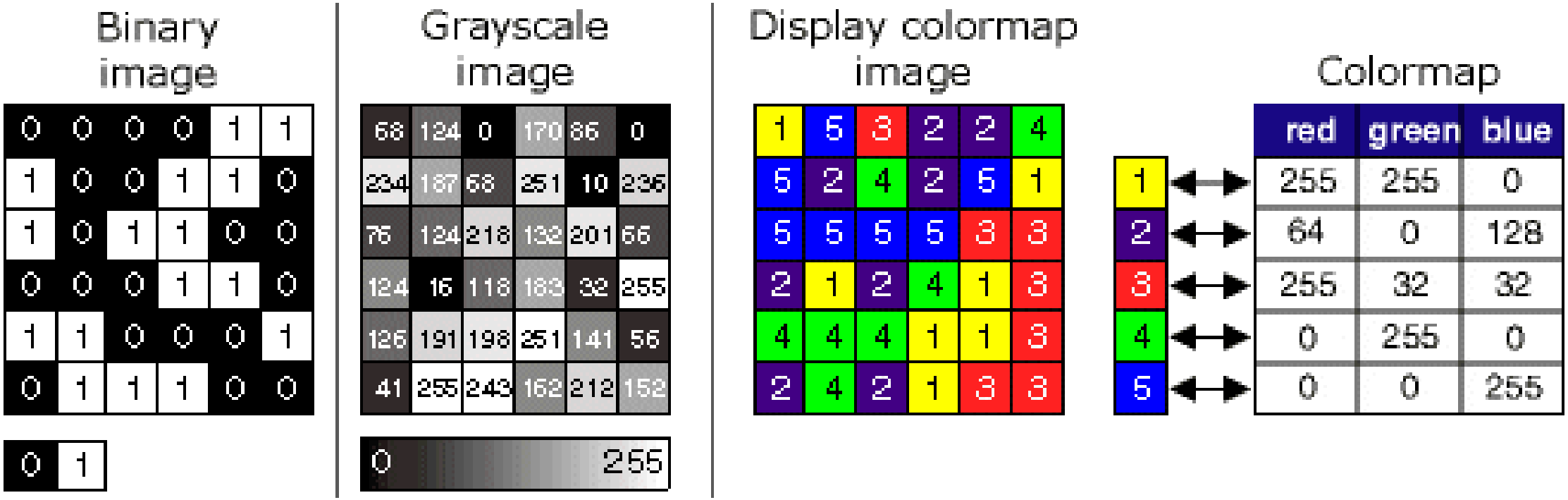

Band 1
Band 2
Band 3
Band 4
Band 5
Band 6
Band 7
Blue
Green
Red

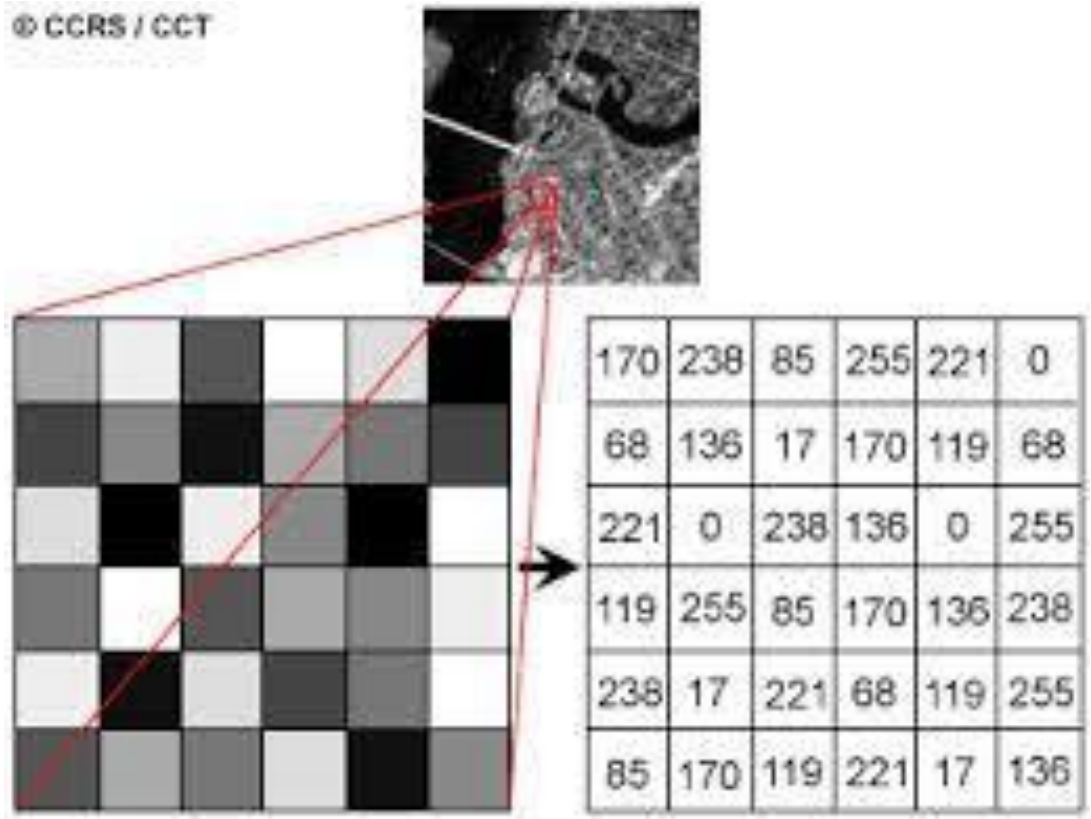
Introduction

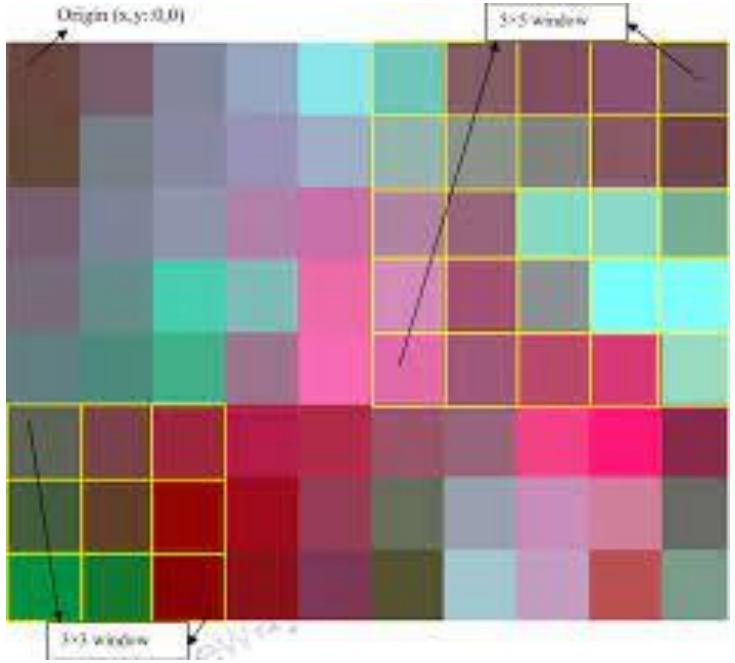
Some rasters have a single band, or layer (a measure of a single characteristic), of data, while others have multiple bands. Basically, a band is represented by a single matrix of cell values, and a raster with multiple bands contains multiple spatially coincident matrices of cell values representing the same spatial area. An example of a single-band raster dataset is a digital elevation model (DEM). Each cell in a DEM contains only one value representing surface elevation. You can also have a single-band orthophoto, which is sometimes called a panchromatic or grayscale image. Most satellite imagery has multiple bands, typically containing values within a range or band of the electromagnetic spectrum.

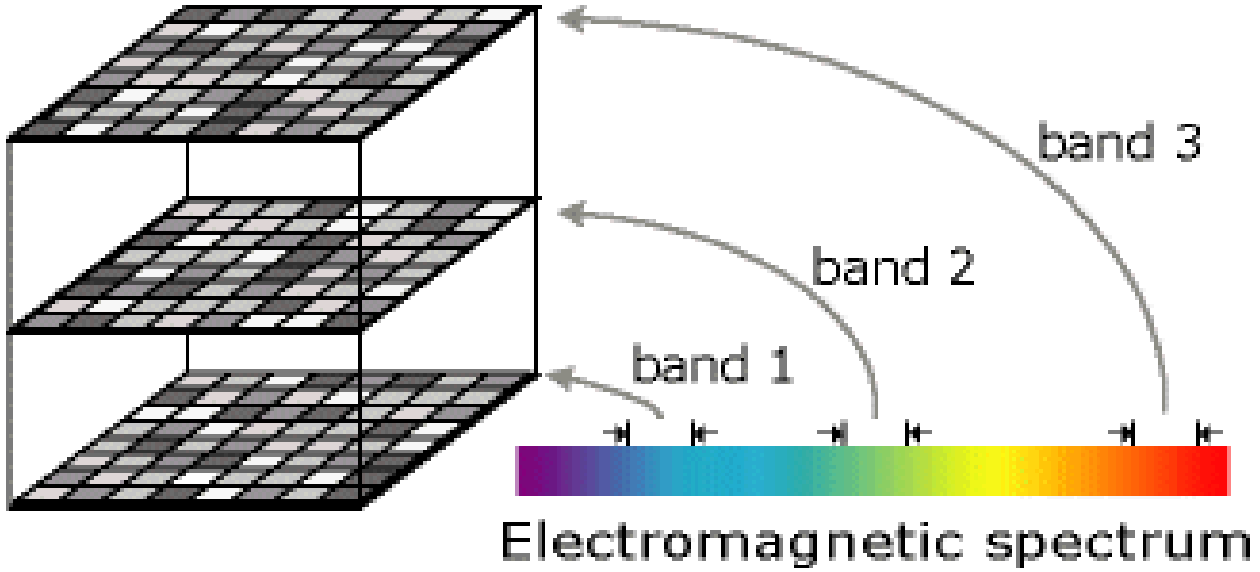
There are three main ways to display (render) single-band raster datasets:
Using two colors—In a binary image, each cell has a value of 0 or 1 and is often displayed using black and white. This type of display is often used for displaying scanned maps with simple line work, such as parcel maps.
Grayscale—In a grayscale image, each cell has a value from 0 to another number, such as 255 or 65535. These are often used for black-and-white aerial photographs.
Color map—One way to represent colors on an image is with a color map. A set of values is coded to match a defined set of red, green, and blue (RGB) values

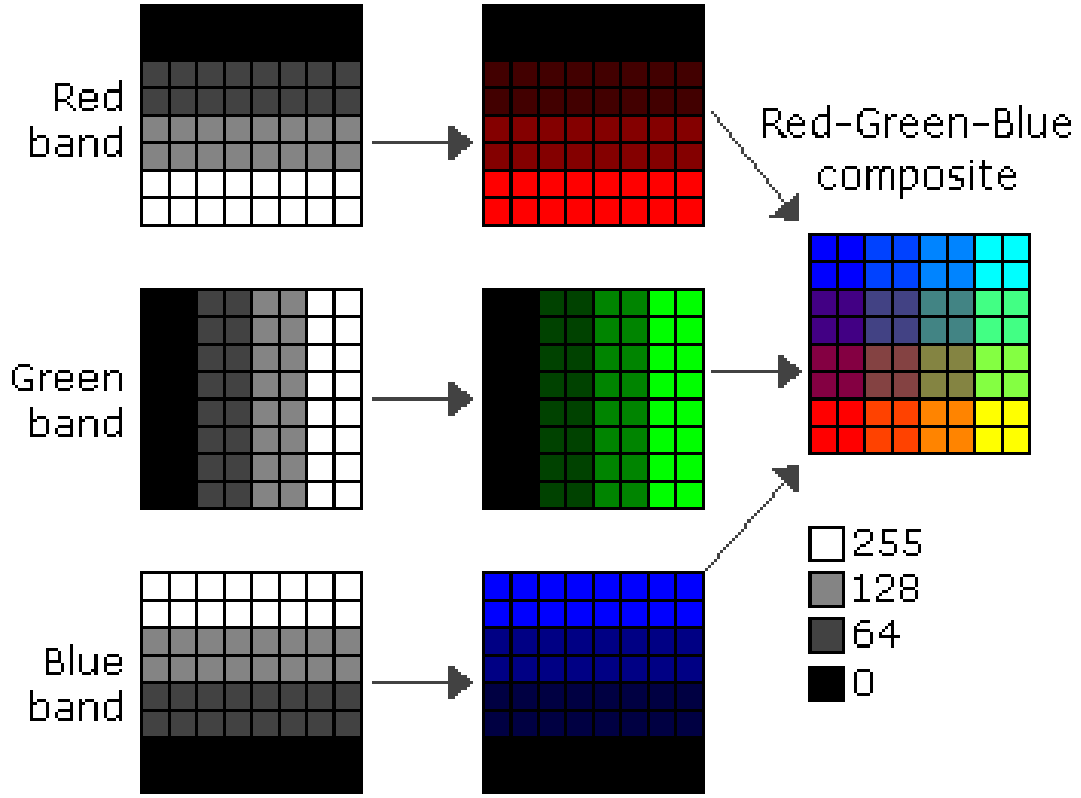
When there are multiple bands, every cell location has more than one value associated with it. With multiple bands, each band usually represents a segment of the electromagnetic spectrum collected by a sensor. Bands can represent any portion of the electromagnetic spectrum, including ranges not visible to the eye, such as the infrared or ultraviolet sections. The term band originated from the reference to the color band on the electromagnetic spectrum.
When you create a map layer from a raster image, you can choose to display a single band of data or form a color composite from multiple bands. A combination of any three of the available bands in a multiband raster dataset can be used to create RGB composites. By displaying bands together as RGB composites, often more information is gleaned from the dataset than if you were to work with just one band.


Binary image
0 0 0 0 1 1
1 0 0 1 1 0
1 0 1 1 0 0
0 0 0 1 1 0
1 1 0 0 0 1
0 1 1 1 0 0
0 1
Grayscale image
68 124 0 170 86 0
234 187 68 251 10 236
76 124 218 132 201 66
124 16 118 183 32 255
126 191 198 251 141 56
41 255 243 162 212 152
0 255
Display colormap image
1 5 3 2 2 4
5 2 4 2 5 1
5 5 5 5 3 3
2 1 2 4 1 3
4 4 4 1 1 3
2 4 2 1 3 3
Colormap
red green blue
1 255 255 0
2 64 0 128
3 255 32 32
4 0 255 0
5 0 0 255

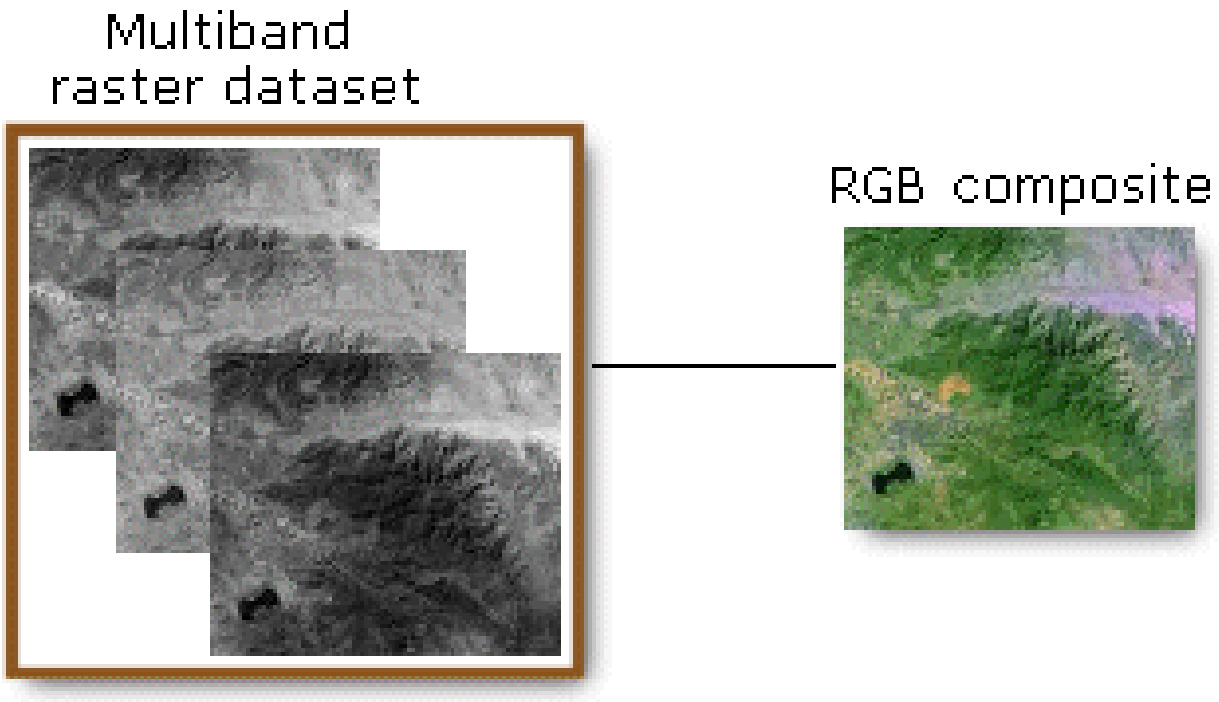

© CCRS / CCT
170 238 85 255 221 0
68 136 17 170 119 68
221 0 238 136 0 255
119 255 85 170 136 238
238 17 221 68 119 255
85 170 119 221 17 136


Origin (x,y: 0,0)
5×5 window
3×3 window


band 3
band 2
band 1
Electromagnetic spectrum


Red band
Green band
Blue band
Red-Green-Blue composite
255
128
64
0

A satellite image, for example, commonly has multiple bands representing different wavelengths from the ultraviolet through the visible and infrared portions of the electromagnetic spectrum. Landsat imagery, for example, is data collected from seven different bands of the electromagnetic spectrum. Bands 1–7, including 6, represent data from the visible, near infrared, and midinfrared regions. Band 6 collects data from the thermal infrared region. Another example of a multiband image is a true color orthophoto in which there are three bands, each representing either red, green, or blue light.



Define Image : ee.Image(args)

An object to represent an Earth Engine image.
This constructor accepts a variety of arguments:
- *A string: an EarthEngine asset id,*
- *A string and a number: an EarthEngine asset id and version,*
- *A number or ee.Array: creates a constant image,*
- *A list: creates an image out of each list element and combines them into a single image,*
- *An ee.Image: returns the argument,*
- *Nothing: results in an empty transparent image.*

Raster information

The Raster information section lists the raster-specific properties, including the following:

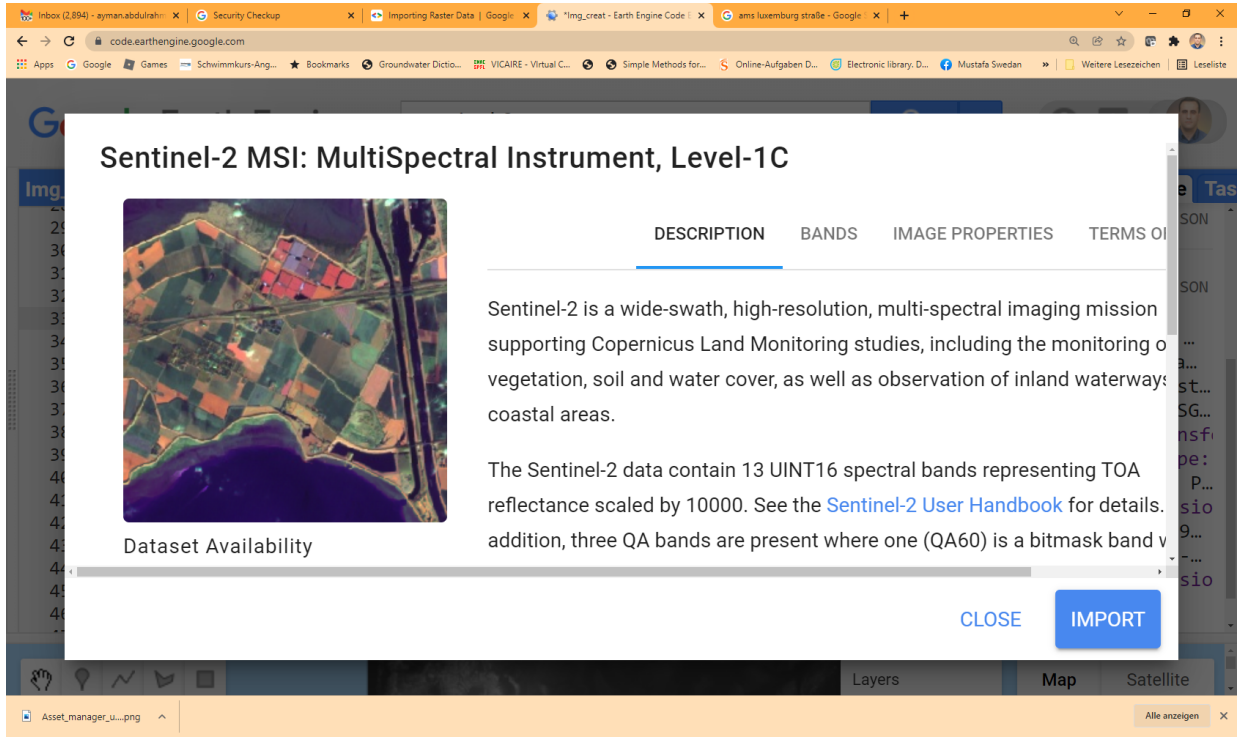
Number of columns and rows (of pixels)
Number of bands
Cell size (x,y)
Uncompressed size
Format
Source type
Pixel type (unsigned/signed, integer/floating point)
Pixel depth/Bit depth (1, 2, 4, 8, 16, 32, 64)
NoData value
Colormap (present/absent)
Pyramids
Compression type
Mensuration Capabilities

The information displayed on the Key Metadata tab (if available) includes the following:

Sensor name
Product name
Acquisition date
Cloud cover
Sun azimuth
Sun elevation
Sensor azimuth
Sensor elevation
Off-nadir angle
Band name (per band)
Minimum wavelength (per band)
Maximum wavelength (per band)
Radiance gain
Radiance bias
Solar irradiance
Reflectance gain
Reflectance bias

- *Import Image*

You can use the Asset Manager or command line interface (CLI) to upload image or other georeferenced raster datasets in GeoTIFF or TFRecord format.



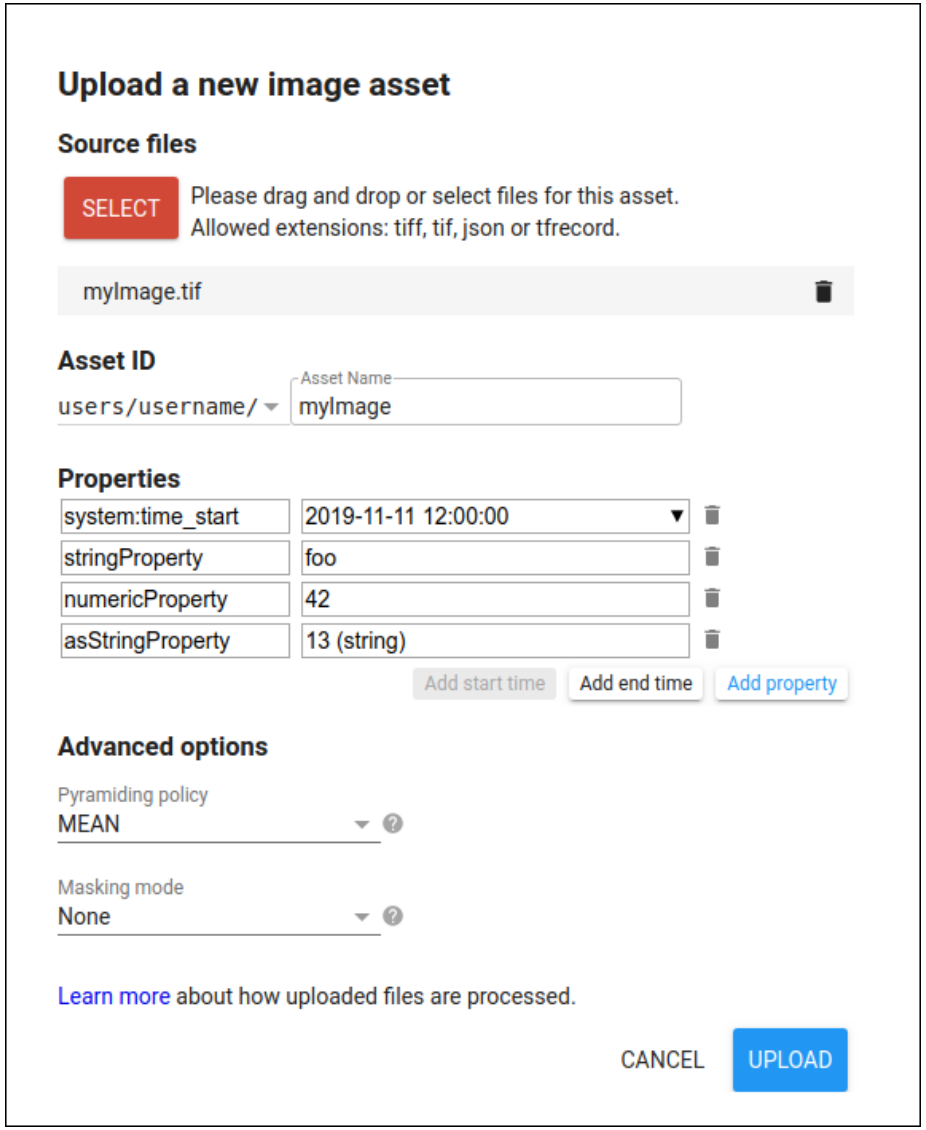


Image Properties

Image.propertyNames()	*Returns the names of properties on this element.*
Image.getInfo(callback)	*returns information about this image*
Image.get(property)	*Extract a property from a feature.*
Image.bandNames()	*Returns a list containing the names of the bands of image.*
Image.bandTypes()	*Returns a dictionary of the image's band types.*

Image.metadata(property, name)
Generates a constant image of type double from a metadata property.

Visualizatio : Map.addLayer

Map.addLayer(eeObject, visParams, name, shown, opacity)

Argument	***Type***	***Details***
eeObject	*Collection|Feature|Image|RawMapId*	*The object to add to the map.*
visParams	*FeatureVisualizationParameters|ImageVisualizationParameters, optional*	*The visualization parameters. For Images and ImageCollection, see ee.data.getMapId for valid parameters. For Features and FeatureCollections, the only supported key is "color", as a CSS 3.0 color string or a hex string in "RRGGBB" format. Ignored when eeObject is a map ID.*
name	*String, optional*	*The name of the layer. Defaults to "Layer N".*
shown	*Boolean, optional*	*A flag indicating whether the layer should be on by default.*

- *Visualisation of Image*

Image.visualize(bands, gain, bias, min, max, gamma, opacity, palette, forceRgbOutput)

Image

Argument	*Type*	*Details*
this: image	*Image*	*The image to visualize.*
bands	*Object, default: null*	*A list of the bands to visualize. If empty, the first 3 are used.*
gain	*Object, default: null*	*The visualization gain(s) to use.*
bias	*Object, default: null*	*The visualization bias(es) to use.*
min	*Object, default: null*	*The value(s) to map to RGB8 value 0.*
max	*Object, default: null*	*The value(s) to map to RGB8 value 255.*
gamma	*Object, default: null*	*The gamma correction factor(s) to use.*
opacity	*Number, default: null*	*The opacity scaling factor to use.*
palette	*Object, default: null*	*The color palette to use. List of CSS color identifiers or hexadecimal color strings (e.g. ['red', '00FF00', 'bluevlolet']).*
forceRgbOutput	*Boolean, default: false*	*Whether to produce RGB output even for single-band inputs.*

Mathematical Operations

Arithmitical Functions

Image.add(image2)
Image.subtract(image2)
Image.multiply(image2)
Image.divide(image2)

logical Functions

- *Image.gt(image2)*
- *Image.gte(image2)*
- *Image.lt(image2)*
- *Image.lte(image2)*
- *Image.eq(image2)*
- *Image.neq(image2)*

- *mage.and(image2)*
- *Image.or(image2)*
- *Image.not()*

Image Functions

- *Image.addBands(srcImg, names, overwrite)*
- *ee.Image.cat(var_args)*
- *Image.clamp(low, high)*
- *Image.clip(geometry)*
- *ee.Image.constant(value)*
- *Image.convolve(kernel)*
- *Image.expression(expression, map)*
- *Image.gradient()*
- *Image.hypot(image2)*
- *Image.mask(mask)*
- *ee.Image.pixelArea()*
- *ee.Image.pixelCoordinates(projection)*
- *ee.Image.pixelLonLat()*

Image Functions

- *Image.projection()*
- *ee.Image.random(seed, distribution)*
- *Image.select(var_args)*
- *Image.set(var_args)*
- *Image.signum()*
- *Image.slice(start, end)*
- *Image.unitScale(low, high)*
- *Image.where(test, value)*

Reducing

Applies a reducer to all of the bands of an image.
The reducer must have a single input and will be called at each pixel to reduce the stack of band values.
The output image will have one band for each reducer output.

Image.reduce(reducer)

Argument	Type	Details
this: image	Image	The image to reduce.
reducer	Reducer	The reducer to apply to the given image.

Reducing

Image.reduceNeighborhood(reducer, kernel, inputWeight, skipMasked, optimization)

Image.reduceRegion(reducer, geometry, scale, crs, crsTransform, bestEffort, maxPixels, tileScale)

Image.reduceRegions(collection, reducer, scale, crs, crsTransform, tileScale)

Image.reduceToVectors(reducer, geometry, scale, geometryType, eightConnected, labelProperty, crs, crsTransform, bestEffort, maxPixels, tileScale, geometryInNativeProjection)

Image.reduceConnectedComponents(reducer, labelBand, maxSize)

Rrducer

ee.Reducer.mean()
ee.Reducer.median(maxBuckets, minBucketWidth, maxRaw)
ee.Reducer.min(numInputs)
ee.Reducer.max(numInputs)
ee.Reducer.minMax()
ee.Reducer.sum()
ee.Reducer.mode(maxBuckets, minBucketWidth, maxRaw)
ee.Reducer.percentile(percentiles, outputNames, maxBuckets, minBucketWidth, maxRaw)

- *Kernel*
  - *ee.Kernel.chebyshev(radius, units, normalize, magnitude)*
  - *ee.Kernel.circle(radius, units, normalize, magnitude)*
  - *ee.Kernel.compass(magnitude, normalize)*
  - *ee.Kernel.cross(radius, units, normalize, magnitude)*
  - *ee.Kernel.diamond(radius, units, normalize, magnitude)*
  - *ee.Kernel.euclidean(radius, units, normalize, magnitude)*
  - *ee.Kernel.fixed(width, height, weights, x, y, normalize)*
  - *ee.Kernel.gaussian(radius, sigma, units, normalize, magnitude)*
  - *ee.Kernel.kirsch(magnitude, normalize)*
  - *ee.Kernel.laplacian4(magnitude, normalize)*
  - *ee.Kernel.laplacian8(magnitude, normalize)*
  - *ee.Kernel.manhattan(radius, units, normalize, magnitude)*
  - *ee.Kernel.octagon(radius, units, normalize, magnitude)*
  - *ee.Kernel.plus(radius, units, normalize, magnitude)*
  - *ee.Kernel.prewitt(magnitude, normalize)*
  - *ee.Kernel.rectangle(xRadius, yRadius, units, normalize, magnitude)*
  - *ee.Kernel.roberts(magnitude, normalize)*
  - *ee.Kernel.sobel(magnitude, normalize)*
  - *ee.Kernel.square(radius, units, normalize, magnitude)*

Masking

- *Masking pixels in an image makes those pixels transparent and excludes them from analysis.*
- *Each pixel in each band of an image has a mask. Those with a mask value of 0 or below will be transparent. Those with a mask of any value above 0 will be rendered.*
- *The mask of an image is set using a call like image1.mask(image2). This call takes the values of image2 and makes them the mask of image1. Any pixels in image2 that have the value 0 will be made transparent in image1.*

- *Export of Image*

You can export images, map tiles, tables and video from Earth Engine.
The exports can be sent :
1. to your Google Drive account,
2.to Google Cloud Storage or
3. to a new Earth Engine asset.

Export.image.toAsset

Creates a batch task to export an Image as a raster to an Earth Engine asset. Tasks can be started from the Tasks tab.
Export.image.toAsset(image, description, assetId, pyramidingPolicy, dimensions, region, scale, crs, crsTransform, maxPixels, shardSize)

Argument	Type	Diletas
image	Image	The image to export.
description	String, optional	A human-readable name of the task. Defaults to "myExportImageTask".
assetId	String, optional	The destination asset ID.
pyramidingPolicy	Object, optional	The pyramiding policy to apply to each band in the image, keyed by band name. Values must be one of: mean, sample, min, max, or mode. Defaults to "mean". A special key, ".default" may be used to change the default for all bands.
dimensions	Number|String, optional	The dimensions to use for the exported image. Takes either a single positive integer as the maximum dimension or"WIDTHxHEIGHT" where WIDTH and HEIGHT are each positive integers.
region	Geometry.LinearRing|Geometry.Polygon|String, optional	A LinearRing, Polygon, or coordinates representing region to export. These may be specified as the Geometry objects or coordinates serialized as a string. If not specified, the region defaults to the viewport at the time of invocation.
scale	Number, optional	Resolution in meters per pixel. Defaults to 1000.
crs	String, optional	CRS to use for the exported image.
crsTransform	List, optional	Affine transform to use for the exported image. Requires "crs" to be defined.
maxPixels	Number, optional	Restrict the number of pixels in the export. By default, you will see an error if the export exceeds 1e8 pixels. Setting this value explicitly allows one to raise or lower this limit.
shardSize	Number, optional	Size in pixels of the tiles in which this image will be computed. Defaults to 256.

Export.image.toCloudStorage

Creates a batch task to export an Image as a raster to Google Cloud Storage. Tasks can be started from the Tasks tab.

Export.image.toCloudStorage(image, description, bucket, fileNamePrefix, dimensions, region, scale, crs, crsTransform, maxPixels, shardSize, fileDimensions, skipEmptyTiles, fileFormat, formatOptions)

Argument	Type	Details
image	Image	The image to export.
description	String, optional	A human-readable name of the task. Defaults to "myExportImageTask".
bucket	String, optional	The Cloud Storage destination bucket.
fileNamePrefix	String, optional	The string used as the output's prefix. A trailing "/" indicates a path. Defaults to the task's description.
dimensions	Number|String, optional	The dimensions to use for the exported image. Takes either a single positive integer as the maximum dimension or "WIDTHxHEIGHT" where WIDTH and HEIGHT are each positive integers.
region	Geometry.LinearRing|Geometry.Polygon|String, optional	A LinearRing, Polygon, or coordinates representing region to export. These may be specified as the Geometry objects or coordinates serialized as a string. If not specified, the region defaults to the viewport at the time of invocation.
scale	Number, optional	Resolution in meters per pixel. Defaults to 1000.
crs	String, optional	CRS to use for the exported image.
crsTransform	List, optional	Affine transform to use for the exported image. Requires "crs" to be defined.
maxPixels	Number, optional	Restrict the number of pixels in the export. By default, you will see an error if the export exceeds 1e8 pixels. Setting this value explicitly allows one to raise or lower this limit.
shardSize	Number, optional	Size in pixels of the tiles in which this image will be computed. Defaults to 256.
fileDimensions	List, optional	The dimensions in pixels of each image file, if the image is too large to fit in a single file. May specify a single number to indicate a square shape, or an array of two dimensions to indicate (width,height). Note that the image will still be clipped to the overall image dimensions. Must be a multiple of shardSize.
skipEmptyTiles	Boolean, optional	If true, skip writing empty (i.e. fully-masked) image tiles. Defaults to false.
fileFormat	String, optional	The string file format to which the image is exported. Currently only 'GeoTIFF' and 'TFRecord' are supported, defaults to 'GeoTIFF'.

Export.image.toDriv

Creates a batch task to export an Image as a raster to Drive. Tasks can be started from the Tasks tab.

Export.image.toDrive(image, description, folder, fileNamePrefix, dimensions, region, scale, crs, crsTransform, maxPixels, shardSize, fileDimensions, skipEmptyTiles, fileFormat, formatOptions)

Argument	Type	Details
image	Image	The image to export.
description	String, optional	A human-readable name of the task. May contain letters, numbers, -, _ (no spaces). Defaults to "myExportImageTask".
folder	String, optional	The Google Drive Folder that the export will reside in. Note: (a) if the folder name exists at any level, the output is written to it, (b) if duplicate folder names exist, output is written to the most recently modified folder, (c) if the folder name does not exist, a new folder will be created at the root, and (d) folder names with separators (e.g. 'path/to/file') are interpreted as literal strings, not system paths. Defaults to Drive root.
fileNamePrefix	String, optional	The filename prefix. May contain letters, numbers, -, _ (no spaces). Defaults to the description.
dimensions	Number|String, optional	The dimensions to use for the exported image. Takes either a single positive integer as the maximum dimension or "WIDTHxHEIGHT" where WIDTH and HEIGHT are each positive integers.
region	Geometry.LinearRing|Geometry.Polygon|String, optional	A LinearRing, Polygon, or coordinates representing region to export. These may be specified as the Geometry objects or coordinates serialized as a string. If not specified, the region defaults to the viewport at the time of invocation.
scale	Number, optional	Resolution in meters per pixel. Defaults to 1000.
crs	String, optional	CRS to use for the exported image.
crsTransform	List, optional	Affine transform to use for the exported image. Requires "crs" to be defined.
maxPixels	Number, optional	Restrict the number of pixels in the export. By default, you will see an error if the export exceeds 1e8 pixels. Setting this value explicitly allows one to raise or lower this limit.
shardSize	Number, optional	Size in pixels of the tiles in which this image will be computed. Defaults to 256.
fileDimensions	List, optional	The dimensions in pixels of each image file, if the image is too large to fit in a single file. May specify a single number to indicate a square shape, or an array of two dimensions to indicate (width,height). Note that the image will still be clipped to the overall image dimensions. Must be a multiple of shardSize.
skipEmptyTiles	Boolean, optional	If true, skip writing empty (i.e. fully-masked) image tiles. Defaults to false.
fileFormat	String, optional	The string file format to which the image is exported. Currently only 'GeoTIFF' and 'TFRecord' are supported, defaults to 'GeoTIFF'.

- ***Charts***

ui.Chart.image.byRegion

Generates a Chart from an image. Extracts and plots band values in one or more regions in the image, with each band in a separate series.

- X-axis = Region labeled by xProperty (default: 'system:index')
- Y-axis = Reducer output.
- Series = Band name.

ui.Chart.image.byRegion(image, regions, reducer, scale, xProperty)

Argument	***Type***	***Details***
İmage	*Image*	*Image to extract band values from.*
regions	*Feature|FeatureCollection|Geometry|List, optional*	*Regions to reduce. Defaults to the image's footprint.*
reducer	*Reducer, optional*	*Reducer that generates the value(s) for the y-axis. Must return a single value per band. Defaults to ee.Reducer.mean().*
scale	*Number, optional*	*Scale to use with the reducer in meters.*
xProperty	*String, optional*	*Property to be used as the label for each Region on the x-axis. Defaults to 'system:index'.*

ui.Chart.image.histogram

Generates a Chart from an image. Computes and plots histograms of the values of the bands in the specified region of the image

- X-axis: Histogram buckets (of band value).
- Y-axis: Frequency (number of pixels with a band value in the bucket).

ui.Chart.image.histogram(image, region, scale, maxBuckets, minBucketWidth, maxRaw, maxPixels)

Argument	Type	Details
image	Image	The image to generate a histogram from.
region	Feature|FeatureCollection|Geometry, optional	The region to reduce. If omitted, uses the entire image.
scale	Number, optional	The pixel scale used when applying the histogram reducer, in meters.
maxBuckets	Number, optional	The maximum number of buckets to use when building a histogram; will be rounded up to a power of 2.
minBucketWidth	Number, optional	The minimum histogram bucket width, or null to allow any power of 2.
maxRaw	Number, optional	The number of values to accumulate before building the initial histogram.
maxPixels	Number, optional	If specified, overrides the maximum number of pixels allowed in the histogram reduction. Defaults to 1e6.

ui.Chart.image.regions

Generates a Chart from an image. Extracts and plots the value of each band in one or more regions.
- X-axis = Band labeled by xProperty (default: band name).
- Y-axis = Reducer output.
- Series = Region labeled by seriesProperty (default: 'system:index').

ui.Chart.image.regions(image, regions, reducer, scale, seriesProperty, xLabels) ui.Chart

Argument	Type	Details
image	Image	Image to extract band values from.
regions	Feature|FeatureCollection|Geometry|List, optional	Regions to reduce. Defaults to the image's footprint.
reducer	Reducer, optional	Reducer that generates the value(s) for the y-axis. Must return a single value per band.
scale	Number, optional	The pixel scale in meters.
seriesProperty	String, optional	Property to be used as the label for each region in the legend. Defaults to 'system:index'.
xLabels	List, optional	A list of labels used for bands on the x-axis. Must have the same number of elements as the image bands. If omitted, bands will be labeled with their names. If the labels are numeric (e.g. wavelengths), x-axis will be continuous.