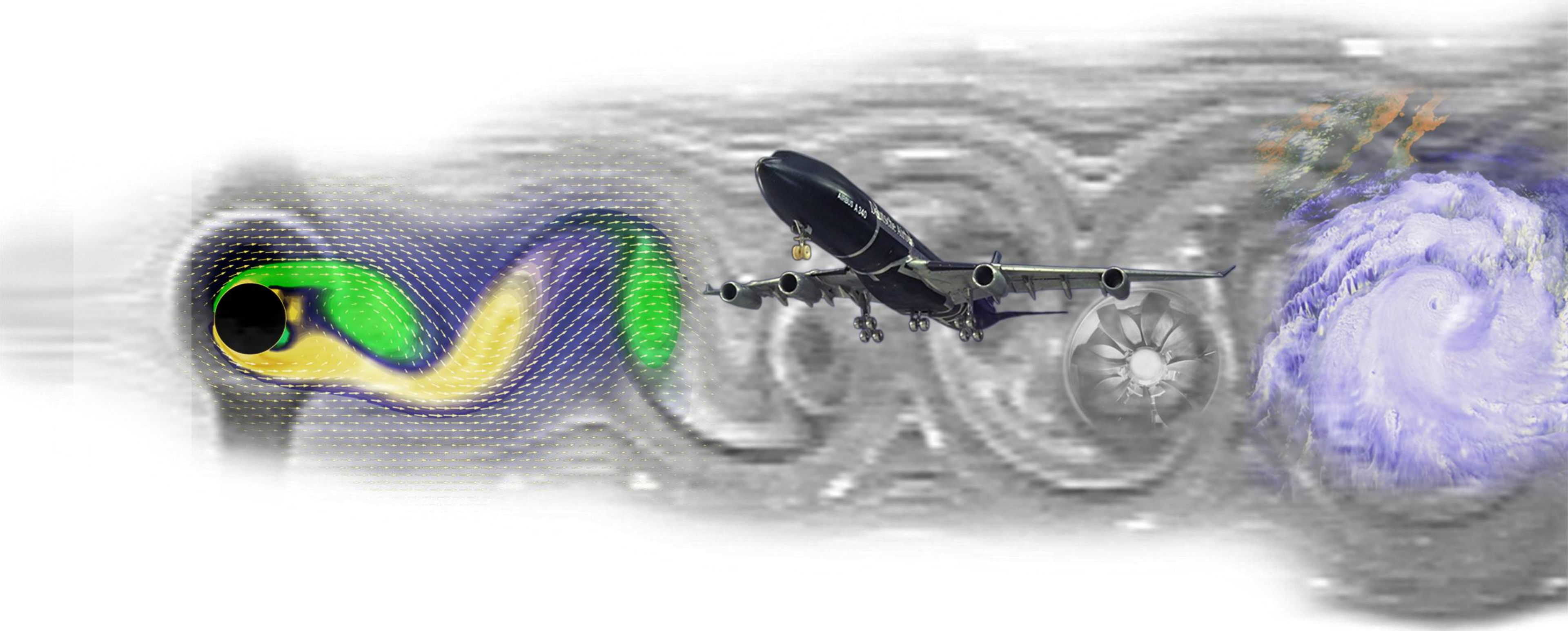
Manual readimx 2.1.8



Overview

- Reading all LaVision file formats: IMG/IMX, VEC/VC7, Set
- Access to all data from the files
- Organized data in a hierarchical structure
- No data conversion/copying

Download

- Download from our webpage: www.lavision.de

- Requirements:

  Matlab 2017a or higher on Windows 10

  Matlab 2017a or higher on OS X 10.11 or higher

Installation

1. Unzip download in folder of your choice

2. Open Matlab and add the path to the search path:

```
>>addpath <MY_READIMX_PATH>
```

or use

```
>>pathtool
```

Buffer data model

The Matlab command

```
>> B=readimx('B00001.im7‘)  % Reads an image from a file
```

loads an image file and creates a **Buffer** structure with two fields:

```
B =

    Frames: {[1x1 struct]}

    Attributes: {19x1 cell}
```

The field *Frames* is a cell array of frame structures and holds all frame data of the file.
The field *Attributes* is a cell array of attribute structures and holds all buffer attributes from the file.

Frame data model

The Matlab command

```
>>F=B.Frames{1}    % Access 1. frame
```

returns a **Frame** structure with the following fields:

```
F=
  Components: {[1x1 struct]}
  Attributes: {13x1 cell}
  Scales: [1x1 struct]
  ComponentNames: {'PIXEL'}
  IsVector: 0
  Grids: [1x1 struct]
```

The field *Components is a cell array of* **Component** structures and holds the image or vector data.
The field *Attributes* is a cell array of **Attribute** structures an hold the frame attributes.
The field *Scales* is a structure holding the X, Y, Z ,I scale information of the image or vector data.
The field *ComponentNames* is a cell array of names, giving the meanings of the components.
The field *IsVector* indicates weather the frame contains vector components or image planes.
The field *Grids* is a structure for grid spaces in X, Y, Z direction.

Component data model

The Matlab command

```
>>C=B.Frames{1}.Components{1} }    % Access 1.  component of 1. frame
```

returns a **Component** structure with the following fields:

```
C=

  Scale: [1x1 struct]

  Planes: {[1152x896 uint16]}

  Name: 'PIXEL‘
```

The field *Scale is a* **Scale** structure for intensity scaling of image or component data.
The field *Planes* is a cell array of **Plane** structures and holds the components data.
The field *Name* is the component name of this component.

Other data models

A **Plane** is a 2-dimensional data array containing data for image (planes) for vector (plane) components.

The **Scale** structure provides data for linear mapping plane data to physical quantities. It has the following fields: *Slope, Offset, Unit, Description.*

The mapping is done by $f(I) = A*I + B$, with slope *A* and offset *B. Unit* and *Description* are simple string.

The **Attribute** structure has two fields: *Name, Value*. The *Name* field is always a string and gives the attribute an identifier. The *Value* field hold the attribute data and has different type: *double, string, array*.

Scripts

File	Description
showimx.m	get a frame, display the data, return compiled data
show2DVec.m	get a frame, display the vectors, return compiled 2D vector data
show3DVec.m	get a frame, display the vectors, return compiled 3D vector data
showPlane.m	get plane and scales, display the image, return compiled 2D image data
showVolume.m	get plane and scales, display slices, return compiled 3D image data
create2DVec.m	get a frame, return compiled 2D vector data
create3DVec.m	get a frame, return compiled 3D vector data
createPlane.m	get plane and scales, return compiled 2D image data
createVolume.m	get plane and scales, return compiled 3D image data
makeFrameInfo.m	get frame, return frame information
readimxdemo.m	read demo files, display results

writeimx

A function for storing LaVision’s IM7/VC7 file format is added to the readimx package. The function is called ***writeimx*** and allows to store 2D/3D arrays and structs from the *readimx* (V.2) function.

Example usage:

```
>>A=rand(222,333)*1023;

>>V=rand(111,444,5)*1023;

>>S=readimx(‘testdata.vc7’)

>>writeimx(uint16(A),’myimage.im7’);    #writing a 2D image A

>>writeimx(uint16(V),’myvolume.im7’);   #writing a 3D volume V

>>writeimx(S,’mystruct.vc7’);   #writing a struct ( with changed data fields) from the readimx function
```

The following data formats for plain image and volume data or in structs are supported:

1. Double
2. Single
3. UNIT8
4. UINT16
5. UINT32
6. INT32

Sets

Since version 2.1.7, readimx allows the reading of buffer from set. The signature is extended to **Set** access and a new function for reading the number buffer in a *Set*. Example:

```
>> n = lvsetsize(‘example.set’);    % Get the number of buffers in a set
>> last = readimx(‘example.set',n); % Reads the last image of a set
>> showimx(last.Frames{1});         % Shows the image
```

Since version 2.1.8, readimx support **MultiSets.** The function ‘lvsetsize’ now returns two sizes (optional, the recent calling is still working). The first one gives the number of images in a subset and the second one gives the number of subsets. Example:

```
>> [n,m] = lvsetsize(‘example.set’);   % Get the size of a (multi) set
>> last = readimx(‘example.set’,n,m);  % Reads the last image of a (multi) set
>> showimx(last.Frames{1});            % Shows the image
```