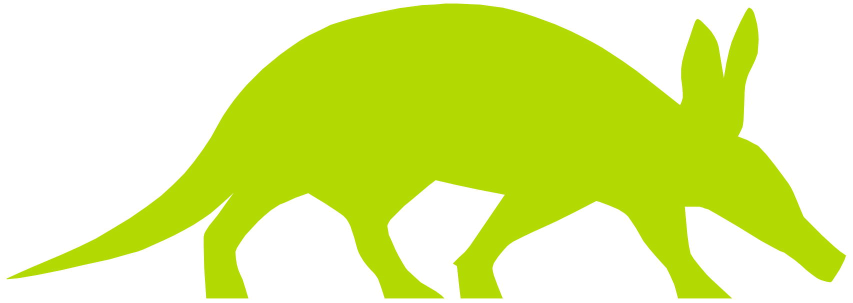


AARDVARK

Tensors & Images
or
How To Avoid Writing Loops

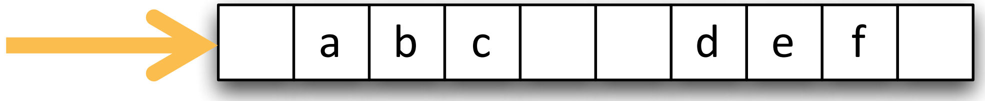
Robert F. Tobler

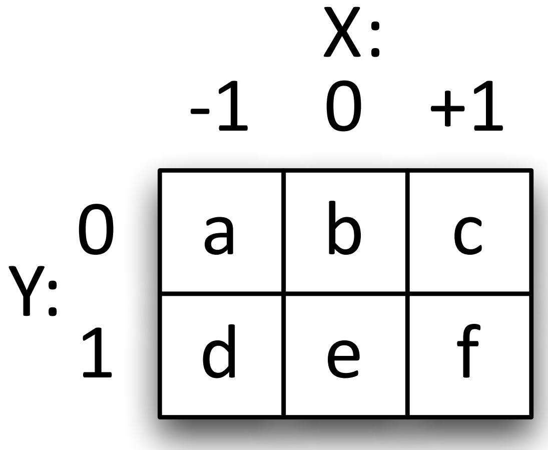
VRVis Research Center
Vienna, Austria

v r vis

The Generic Matrix Data Type

```
struct Matrix<T>
{
    T[]           Data;
    MatrixInfo    Info;
}

struct MatrixInfo
{
    long Origin; // 2
    V2l  Size;   // [3, 2]
    V2l  Delta;  // [1, 5]
    V2l  First;  // [-1, 0]
}
```





Properties of the Generic Matrix Data Type

- does not own the underlying data array: it is just a specific view of the data stored in the array (i.e. it is a Façade)
- can handle various different data layouts (row/column)
  - e.g.: a transposed version can easily be constructed by swapping the X- and Y- components of the **`Size`**, **`Delta`**, and **`First`** fields
- a sub-matrix is of the very same type
- the **`First`** field allows matrices with lowest coordinates that are different from **`[0, 0]`**, e.g. filters with are symmetrical about the origin
  - if this functionality is not used the **`First`** field is **`[0, 0]`**

Other Tensors: Vector, Matrix, and Tensor4

Analogous construction to Matrix

```
Vector<T>:    long Size, Delta, First;
Matrix<T>:    V2l  Size, Delta, First;
Volume<T>:    V3l  Size, Delta, First;
Tensor4<T>:   V4l  Size, Delta, First;
```

Default Memory Layout of Tensors

															…
X:	0	1	2	0	1	2	0	1	2	0	1	2	0	1	…
Y:	0	0	0	1	1	1	0	0	0	1	1	1	0	0	…
Z:	0	0	0	0	0	0	1	1	1	1	1	1	2	2	…

-
  ```
  vol.ForeachZYX(
     (z) => { }                        // pre-plane action
     (z,y) => { }                      // pre-line action
     (z,y,x,i) => {…vol[i]…}           // element action
     (z,y) => { }                      // post-line action
     (z) => { });                      // post-plane action
  ```

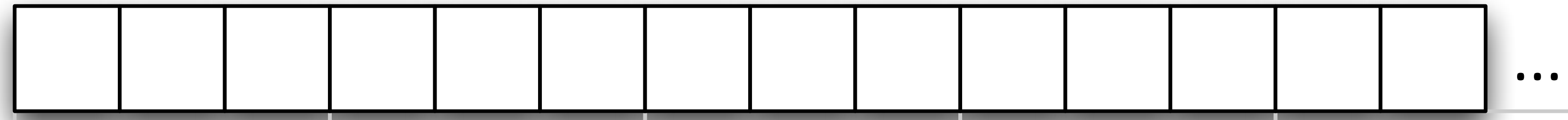
Matrix with different Data and View Types

```
struct Matrix<T>
{
  T[]  Data;
  T    this[long x, long y] { … }
}

struct Matrix<TData, TView>
{
  TData[] Data;
  TView   this[long, long y] { … }

  Func<TData[], long, TView> Getter;
  Action<TData[], long, TView> Setter;
}
```

An Image Class

```
class PixImage
{
  Col.Format Format; // enum, e.g. RGB, RGBA
}

class PixImage<T> : PixImage
{
  Volume<T> Volume;  // T ... channel type
}
```

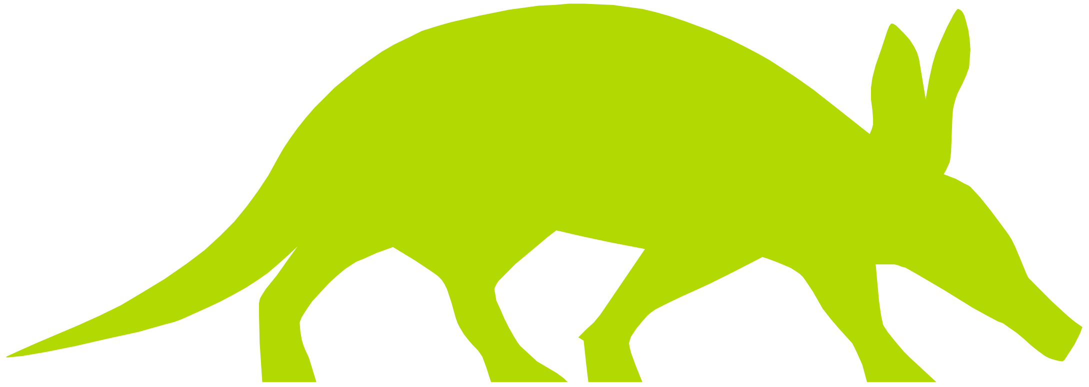
Default Layout of Image Volumes



Z:	R	G	B	R	G	B	R	G	B	R	G	B	R	G	...
X:	0	0	0	1	1	1	0	0	0	1	1	1	0	0	...
Y:	0	0	0	0	0	0	1	1	1	1	1	1	2	2	...

- X, Y are natural image coordinates
- `var matrix = image.GetChannel(Col.Channel.Red);`

  gets a matrix referencing the red channel **without copying**
  - Delta.X = 3, cannot be directly used with Ippi methods
    use `.ToImage()` to obtain matrix that can be used with Ippi



AARDVARK

Thank you for your attention!

Please visit us at

http://www.VRVis.at/

vrvis