```
// Extract layer a from the 3D wave
MatrixOp destWave = wave3d[][][a]
```

You can also extract layers from a 3D wave starting with layer a and increasing the layer number up to layer b using increments c:

```
MatrixOp destWave = wave3d[][][a,b,c]
```

a, b and c must be scalars. Layers are clipped to the valid range and c must be a positive integer.

If you need to loop over rows, columns, layers or chunks it is often useful to extract the relevant data using the MatrixOp functions `row`, `col`, `layer` and `chunk`. You can extract matrix diagonals using `getDiag`. You can use `subRange` to extract more than one row or more than one column.

The **Rotate** operation works on 1D waves. MatrixOp extends this to higher dimensions with the functions: `rotateRows`, `rotateCols`, `rotateLayers`, and `rotateChunks`.

MatrixOp `transposeVol` is similar to **ImageTransform** `transposeVol` but it supports complex data types.

The MatrixOp `redimension` function is designed to convert 1D waves into 2D arrays but it can also extract data from higher-dimensional waves.

MatrixOp Data Promotion Policy

MatrixOp chooses the data type of the destination wave based on the data types and operations used in the expression on the RHS. Here are some examples:

```
Make/O/B/U wave2, wave3             // Create two unsigned byte waves

MatrixOp/O wave1 = wave2            // wave1 is unsigned byte

MatrixOp/O wave1 = wave2 + wave3    // wave1 is unsigned word (16 bit)

MatrixOp/O wave1 = wave2 * wave3    // wave1 is unsigned word (16 bit)

MatrixOp/O wave1 = wave2 / wave3    // wave1 is SP

MatrixOp/O wave1 = magsqr(wave2)    // wave1 is SP
```

The examples show the destination wave's data type changing to represent the range of possible results of the operation. MatrixOp does not inspect the data in the waves to determine if the promotion is required for the specific waves.

Igor variables are usually treated as double precision data tokens. MatrixOp applies special rules for variables that contain integers. Here is an example:

```
Make/O/B/U wave2                    // Create unsigned byte wave
Variable vv = 2.1                   // Variable containing DP value
MatrixOp/O wave1 = wave2*vv         // wave1 is DP
```

Now change the variable to store various integers:

```
vv = 2
MatrixOp/O wave1 = wave2*vv         // wave1 is unsigned word (16 bit)
vv = 257
MatrixOp/O wave1 = wave2*vv         // wave1 is unsigned integer (32 bit)
vv = 65538

MatrixOp/O wave1 = wave2*vv         // wave1 is DP
```

These examples show that MatrixOp represents the variable `vv` as a token with a data type that fits it contents. If the variable does not contain an integer value the token is treated as DP.