# IsoResampleImageStack (Calls: 1, Time: 30.883 s)

*Generated 03-Jul-2024 13:21:21 using performance time.*
Function in file C:\Will\Matlab\Current\MichaelPalmer\NucleusRadialModelClassification\ImagePreprocessing\IsoResampleImageStack.m
Copy to new window for comparing multiple runs

## Parents (calling functions)

No parent

## Lines that take the most time

| Line Number | Code | Calls | Total Time (s) | % Time | Time Plot |
|---|---|---|---|---|---|
| 57 | `J = interp1(zs, I, zq);` | 1 | 22.064 | 71.4% | |
| 62 | `J = cast(J, ogclass);` | 1 | 4.997 | 16.2% | |
| 58 | `J = permute(J, [2 3 1]);` | 1 | 1.810 | 5.9% | |
| 56 | `I = permute(I, [3 1 2]);` | 1 | 1.298 | 4.2% | |
| 28 | `I = single(I);` | 1 | 0.536 | 1.7% | |
| All other lines | | | 0.178 | 0.6% | |
| Totals | | | 30.883 | 100% | |

## Children (called functions)

| Function Name | Function Type | Calls | Total Time (s) | % Time | Time Plot |
|---|---|---|---|---|---|
| interp1 | Function | 1 | 22.063 | 71.4% | |
| Self time (built-ins, overhead, etc.) | | | 8.820 | 28.6% | |
| Totals | | | 30.883 | 100% | |

## Code Analyzer results

No Code Analyzer messages.

## Coverage results

Show coverage for parent folder

| | |
|---|---|
| Total lines in function | 63 |
| Non-code lines (comments, blank lines) | 33 |
| Code lines (lines that can run) | 30 |
| Code lines that did run | 17 |
| Code lines that did not run | 13 |
| Coverage (did run/can run) | 56.67 % |

## Function listing

```
Time    Calls    Line
                   1  function J = IsoResampleImageStack(I, dx, dz, memoryeff)
                   2  % RESAMPLEIMAGESTACK will efficiently resample an image stack. By default,
                   3  % this function is optimized to use minimal memory but if the user has
```

```
% sufficient memory and prefers computational efficiency instead, the
% function will permute the array to interpolate along z and then permute
% the dimensions back.
%
% William A. Ramos, Kumar Lab @ MBL - July 2024

    if nargin < 4 || isempty(memoryeff)
        memoryeff = true;
    end

    % Getting dimensions to resample
    [M, N, Z] = size(I, [1 2 3]);

    % Figuring out number of new z slices
    dz_dx = dz/dx;
    Z2  = ceil(Z*dz_dx);

    % Original sample points
    zs = 1:dz_dx:Z2;
    zq = 1:Z2;

    % Resampling on a row by row basis
    ogclass = class(I);
    % Memory efficient
    I       = single(I);

    % The function will branch in terms of how it interpolates depending on
    % whether the user wants to be memory efficient
    if memoryeff
        % Init array to assign to
        J       = zeros(M, N, Z2, 'single');
        for r = 1:M
            % Pull out the first row
            Im = I(r, :, :);

            % Faster when only operating on rows
            Im = squeeze(Im);
            Im = interp1(zs, Im', zq);
            Im = Im';
            Im = reshape(Im, 1, N, Z2);

            % Assignment on a row by row basis
            J(r,:,:) = Im;

            % Reporting on progress
            pct      = (r/M)*100;
```

| | | |
|---|---|---|
| < 0.001 | 1 | 10 |
| < 0.001 | 1 | 12 |
| < 0.001 | 1 | 15 |
| < 0.001 | 1 | 18 |
| < 0.001 | 1 | 19 |
| < 0.001 | 1 | 22 |
| < 0.001 | 1 | 23 |
| < 0.001 | 1 | 26 |
| 0.536 | 1 | 28 |
| < 0.001 | 1 | 32 |

```
                50              msg      = ['Percent resampled: ' num2str(pct, '%.2f') '%'];
                51              disp(msg)
                52          end
< 0.001         1   53      else
                54          % If memory is not a concern, then the user can easily interpolate
                55          % the entire stack
1.298           1   56          I = permute(I, [3 1 2]);
22.064          1   57          J = interp1(zs, I, zq);
1.810           1   58          J = permute(J, [2 3 1]);
< 0.001         1   59      end
                60
                61      % Cast at end for efficiency
4.997           1   62      J = cast(J, ogclass);
0.177           1   63  end
```