tf.contrib.image.single_image_random_dot_stereograms

```
single_image_random_dot_stereograms(
    depth_values,
    hidden_surface_removal=None,
    convergence_dots_size=None,
    dots_per_inch=None,
    eye_separation=None,
    mu=None,
    normalize=None,
    normalize_max=None,
    normalize_min=None,
    border_level=None,
    number_colors=None,
    output_image_shape=None,
    output_data_window=None
)
```

Defined in `tensorflow/contrib/image/python/ops/single_image_random_dot_stereograms.py`.

Output a RandomDotStereogram Tensor for export via encode_PNG/JPG OP.

Given the 2-D tensor 'depth_values' with encoded Z values, this operation will encode 3-D data into a 2-D image. The output of this Op is suitable for the encode_PNG/JPG ops. Be careful with image compression as this may corrupt the encode 3-D data witin the image.

Based upon this paper.

This outputs a SIRDS image as picture_out.png:

```
img=[[1,2,3,3,2,1],
     [1,2,3,4,5,2],
     [1,2,3,4,5,3],
     [1,2,3,4,5,4],
     [6,5,4,4,5,5]]
session = tf.InteractiveSession()
sirds = single_image_random_dot_stereograms(
    img,
    convergence_dots_size=8,
    number_colors=256,normalize=True)

out = sirds.eval()
png = tf.image.encode_png(out).eval()
with open('picture_out.png', 'wb') as f:
  f.write(png)
```

Args:

- `depth_values`: A `Tensor`. Must be one of the following types: `float64`, `float32`, `int64`, `int32`. Z values of data to encode into 'output_data_window' window, lower further away {0.0 floor(far), 1.0 ceiling(near) after norm}, must be 2-D tensor
- `hidden_surface_removal`: An optional `bool`. Defaults to `True`. Activate hidden surface removal
- `convergence_dots_size`: An optional `int`. Defaults to `8`. Black dot size in pixels to help view converge image,

drawn on bottom of the image

- `dots_per_inch`: An optional `int`. Defaults to `72`. Output device in dots/inch
- `eye_separation`: An optional `float`. Defaults to `2.5`. Separation between eyes in inches
- `mu`: An optional `float`. Defaults to `0.3333`. Depth of field, Fraction of viewing distance (eg. 1/3 = 0.3333)
- `normalize`: An optional `bool`. Defaults to `True`. Normalize input data to [0.0, 1.0]
- `normalize_max`: An optional `float`. Defaults to `-100`. Fix MAX value for Normalization (0.0) - if < MIN, autoscale
- `normalize_min`: An optional `float`. Defaults to `100`. Fix MIN value for Normalization (0.0) - if > MAX, autoscale
- `border_level`: An optional `float`. Defaults to `0`. Value of bord in depth 0.0 {far} to 1.0 {near}
- `number_colors`: An optional `int`. Defaults to `256`. 2 (Black & White), 256 (grayscale), and Numbers > 256 (Full Color) are supported
- `output_image_shape`: An optional `tf.TensorShape` or list of `ints`. Defaults to shape `[1024, 768, 1]`. Defines output shape of returned image in '[X,Y, Channels]' 1-grayscale, 3 color; channels will be updated to 3 if number_colors > 256
- `output_data_window`: An optional `tf.TensorShape` or list of `ints`. Defaults to `[1022, 757]`. Size of "DATA" window, must be equal to or smaller than `output_image_shape`, will be centered and use `convergence_dots_size` for best fit to avoid overlap if possible

Returns:

A `Tensor` of type `uint8` of shape 'output_image_shape' with encoded 'depth_values'

Last updated November 2, 2017.