Late August 2013 scale-space re-organization
(motivated by confusion about differences in representation and communication of SS info
between gage (and its command-line tools) versus pull and meet (and its command-line tools)

The newly organized gageStackBlurParm:

```
typedef struct {
  unsigned int num;      /* # of pre-computed blurring scales == allocated
                            length of the "scale" vector below */
  double sigmaRange[2];  /* sigma range for image blurring */
  int sigmaSampling;     /* from gageSigmaSampling* enum: how to sample the
                            range from sigmaRange[0] to sigmaRange[1] */
  double *sigma;         /* when-non-NULL, the sigma parameter for each
                            blurring level */
  NrrdKernelSpec *kspec; /* the kernel with which we do blurring. */
  int renormalize;       /* renormalize kernel weights (associated with the
                            kernel);  passed to nrrdResampleRenormalizeSet */
  NrrdBoundarySpec *bspec; /* what do to at image boundaries */
  int oneDim,            /* for experimental/debugging purposes: blur *only*
                            along the first (fastest) axis */
    needSpatialBlur,     /* always do blurring in the spatial domain, even
                            if frequency space blurring is possible */
    verbose;             /* verbosity level */
  double dgGoodSigmaMax; /* The same info as communicated by
                            nrrdKernelDiscreteGaussianGoodSigmaMax, but
                            allowing it to be different. With this limit on
                            the sigma we pass to nrrdKernelDiscreteGaussian,
                            instead of doing the blurring in one step, the
                            diffusion is done iteratively, with steps in
                            diffusion time of goodSigmaMax^2 */
} gageStackBlurParm;
```

old command-line options for {p,v}probe
these will not be augmented for {p,v}probe
(gprobe has shiny new -sbp option)

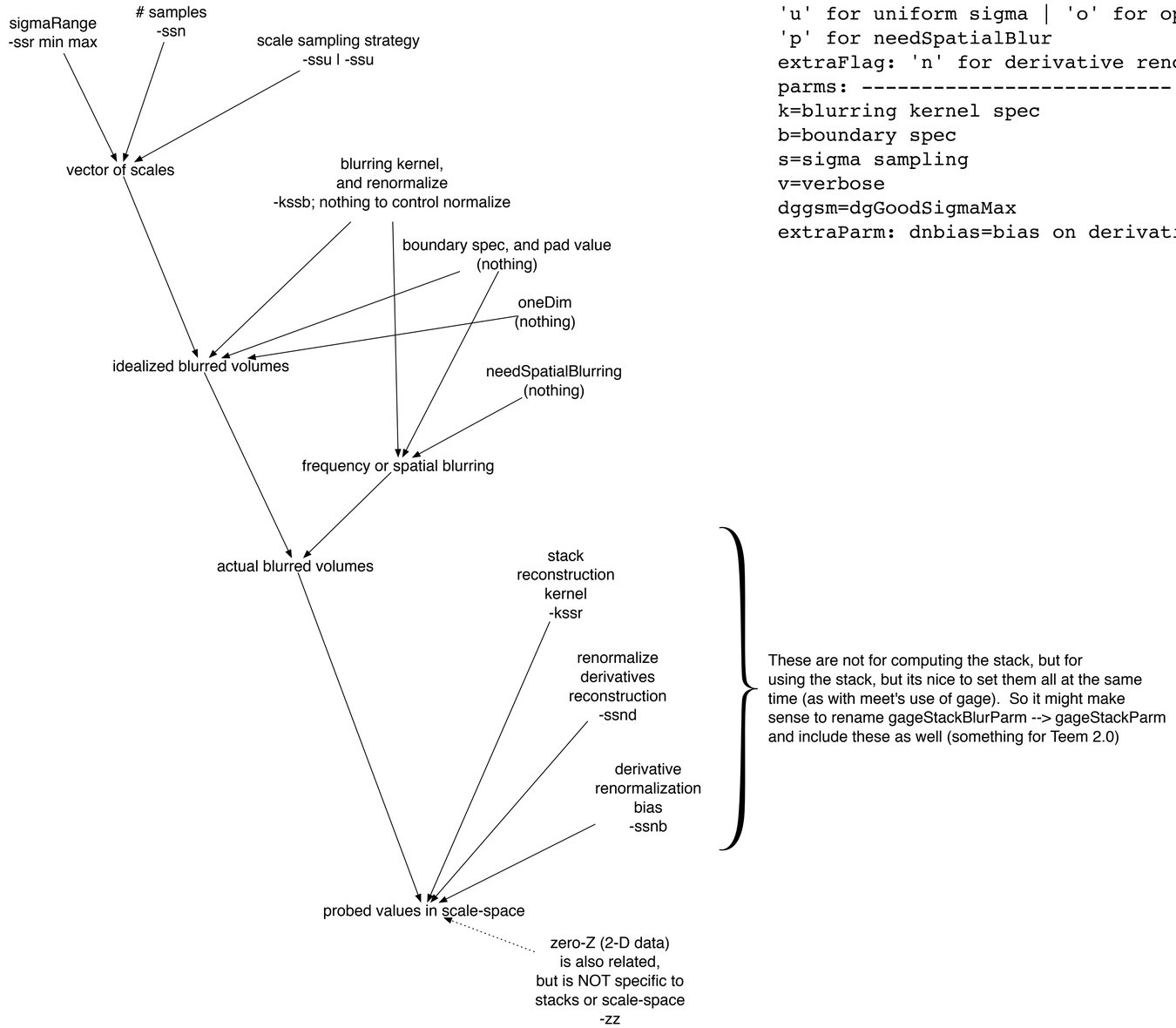
```
-ssn <# samples>
-ssr <minScale> <maxScale>
-ssu: uniform (in sigma) | -sso: optimal
* MISSING: general means of specifying strategy

-kssb <blurring kernel>
* MISSING: control blurring kernel spatial renormalization
* MISSING: boundary, and padValue, if needed
* MISSING: oneDim
* MISSING: needSpatialBlurring

-kssr <scale recon kernel>
-ssnd: normalize derivatives
-ssnb: scale derivative normalization bias

-zz: zeroZ
```

should put the new gageStackBlurParm in the pullVolume?

```
typedef struct {
  int verbose;                /* blah blah blah */
  char *name;                 /* how the volume will be identified
                                 (like its a variable name) */
  const gageKind *kind;
  const Nrrd *ninSingle;      /* don't own */
  const Nrrd *const *ninScale; /* don't own;
                                  NOTE: only one of ninSingle and ninScale
                                  can be non-NULL */
  unsigned int scaleNum;      /* number of scale-space samples (volumes) */
  double *scalePos;           /* location of all samples in scale */
  int scaleDerivNorm;         /* normalize derivatives based on scale */
  double scaleDerivNormBias;  /* bias on derivative normalization by scale */
  NrrdKernelSpec *ksp00,      /* for sampling tensor field */
    *ksp11,                   /* for gradient of mask, other 1st derivs */
    *ksp22,                   /* for 2nd derivatives */
    *kspSS;                   /* for reconstructing from scale-space
                                 samples */
  gageQuery pullValQuery;     /* if this is a pullValGageKind volume,
                                 then we don't have a real gageContext,
                                 and we have to manage our own query */
  gageContext *gctx;          /* do own, and set based on info here */
  gagePerVolume *gpvl,        /* stupid gage API . . . */
    **gpvlSS;                 /* stupid gage API . . . */
  int seedOnly,               /* volume only required for seeding, for
                                 either pullInfoSeedThresh or
                                 pullInfoSeedPreThresh */
    forSeedPreThresh;         /* we learn pullInfoSeedPreThresh from this */
} pullVolume;
```

Actually, no: pull is not the place to be worried about how a stack is created;
it just receives a stack that someone else computes.

4.5) contrive a meetPullVol ctest
5) recompute optimal 3D L2 L2 sigmas
6) fix pull/constraints.c if (0 && point->pos[3]) code; still have to rationalize logic of how constrEps is used



New format for gageStackBlurParm specification string:

```
<minSigma>-<#samples>-<maxSigma>[-flags][/parm[/parm...]]
flags: ------------------------
'1' for one-dim
'r' turn OFF blurring kernel renormalization
'u' for uniform sigma | 'o' for optimalL2L2
'p' for needSpatialBlur
extraFlag: 'n' for derivative renormalization
parms: --------------------------
k=blurring kernel spec
b=boundary spec
s=sigma sampling
v=verbose
dggsm=dgGoodSigmaMax
extraParm: dnbias=bias on derivative normalization
```