GraXpert
Sample-Free Background

Algorithm, block diagram, mathematics, and native preview design

Technical note • 16 October 2026 • Standalone macOS application

What this document describes

The installed **/Applications/GraXpert.app** contains a deterministic Sample-Free method adapted from Siril. It models gradients using robust rejection and smooth interpolation across excluded structures. No user-placed background samples are needed. This is a separate method from GraXpert's learned AI background estimator.

With the default simplified option enabled, the model is the sum of a robust low-degree polynomial and a multiscale model fitted to the polynomial-subtracted image. Each color channel is estimated separately. A background pedestal is retained when applying subtraction or division.

Evidence and version boundaries

The installed bundle reports an old beta display label, while its modules explicitly cite Siril's simplified-model revision of 3 September 2026. The label alone is not a reliable algorithm version. The implementation examined is **graxpert.sample_free_background** plus **graxpert.sample_free_onnx**, recovered read-only from the installed PyInstaller module archive and inspected with Python 3.14. It was exercised directly in numerical tests.

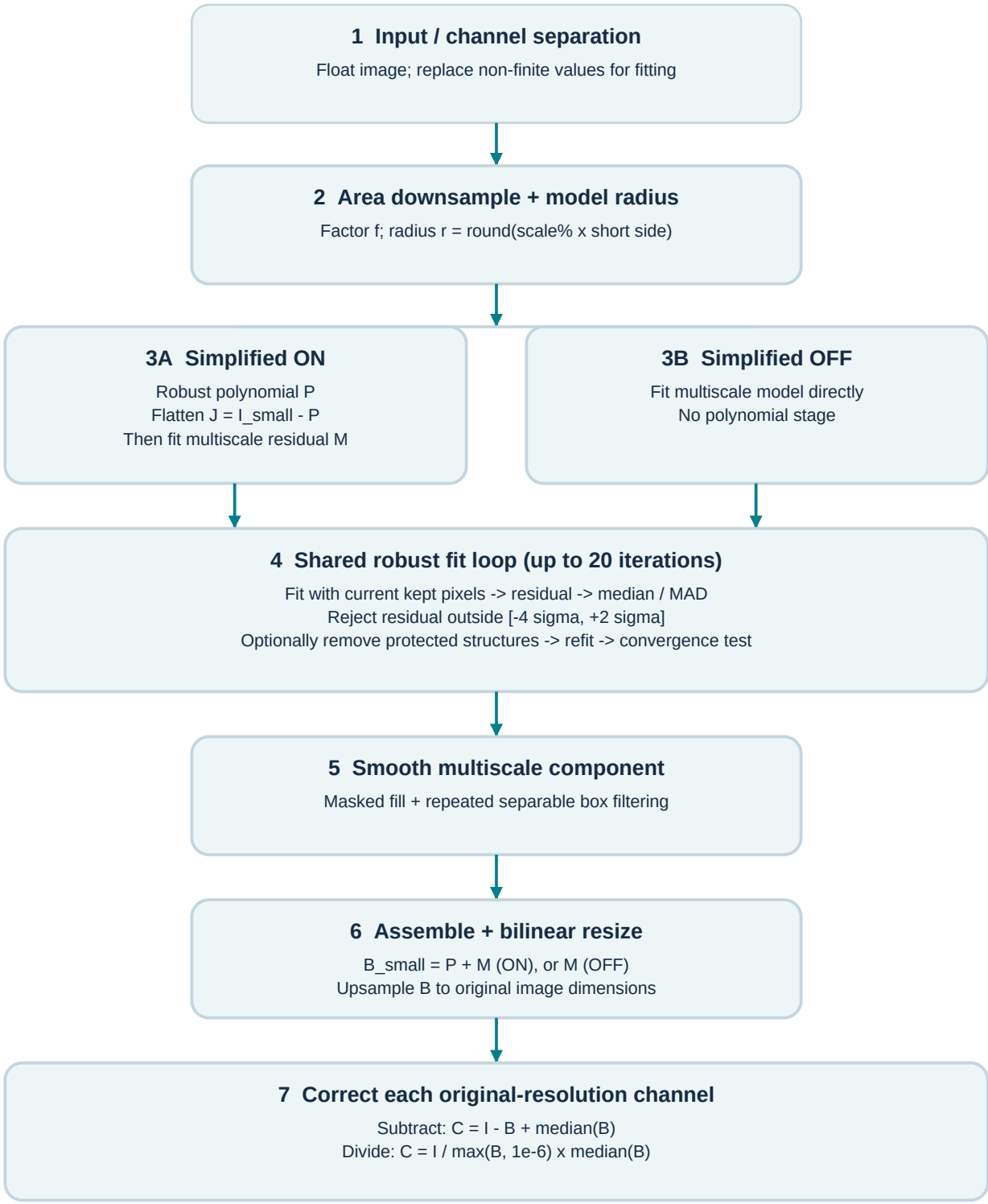
The official upstream GraXpert main branch checked for comparison was commit **c3bc2211f8a4bee864d301987e9a154ad4480969** (1 January 2025). It has AI background extraction but no Sample-Free module. Accordingly, this report makes no claim that the installed build is the latest official upstream release.

Notation

Symbol	Meaning
I, C, B	Input channel, corrected channel, background model
J, P, M	Working image, polynomial surface, multiscale surface
K, R	Boolean keep mask and residual image
f, r, s	Downsample factor, model radius, smoothing multiplier
G	Low-pass operator; the installed port uses repeated box passes

Equations use explicit programming-style notation so the steps translate directly into an implementation. Multiplication is written with *, powers with ^, and pixel indices in brackets.

Pipeline block diagram



The robust-fit loop is used once for the direct model, or independently in both stages of simplified mode. Background and polynomial components can be viewed separately. The ONNX graph is used inside the low-pass operation; it is not a learned semantic segmentation or background predictor.

1-2 Prepare pixels and establish the scale

Channel handling and finite pixels

The core accepts a 2-D monochrome image or an H x W x C array. It validates dimensions and parameters, converts the channel working copy to float64, and fits each channel independently. For fitting, any non-finite values are replaced by the median of finite pixels. A channel with no finite pixels raises an error. The corrected output restores NaN at input positions that were non-finite.

```
I_fit[p] = I[p] if finite(I[p])
I_fit[p] = median(I[q] for finite q) otherwise
```

The GraXpert image loader converts integer images to normalized floating point. Floating data outside [0,1] can be rescaled by that loader. This happens before the core algorithm and matters because the protection threshold is an absolute intensity threshold. A 0.05 threshold is not a 5% probability or a sigma threshold.

Area downsampling

For factor f in {1,2,4,8}, the installed port averages nonoverlapping f x f blocks. It uses floor-sized dimensions and crops any incomplete bottom or right block. If either reduced dimension would be less than two, the factor falls back to one.

```
h = floor(H/f); w = floor(W/f)
J[v,u] = (1/f^2) * sum I_fit[f*v+j, f*u+i]
i,j = 0,...,f-1
```

The original full-resolution image is retained for correction. Only the estimated background is reconstructed. Downsampling therefore accelerates the fit without applying a reduced-resolution output image, although it can change what structures the estimator sees.

Map the Scale control to a radius

```
r = max(1, floor((scale/100)*min(h,w) + 0.5))
```

Scale is a percentage of the shorter working-image dimension. At 4000 x 3000 pixels, factor 4 gives 1000 x 750 and default scale 5 produces radius 38. Increasing scale produces a coarser model that is less able to follow small structures. It does not select a polynomial degree.

Precision and edges

Block means are accumulated in float64 and returned as float32. Cropped remainders are not separate fit samples. Reconstruction spans the entire original extent, so the final edge background is extrapolated from the available reduced grid through the resize mapping. Inspect borders on images with strong vignetting or stacking artifacts.

3 Low-pass smoothing and masked filling

The installed smoothing primitive

The port expresses the intended blur size through a variance-equivalent Gaussian sigma. With p box passes and integer radius r, one 1-D box has variance r(r+1)/3. The variance of repeated passes adds:

```
sigma(r,p) = sqrt(p*r*(r+1)/3)
r_eff = round((-1 + sqrt(1 + 12*sigma^2/3))/2)
```

The installed _ag_gauss routine maps sigma back to an effective radius using AG_PASSES = 3, then runs the bundled ONNX graph. The NumPy reference path performs three horizontal-plus-vertical box passes. For sigma below 0.25 or radius below one, it returns a float32 copy.

```
Box_x(A)[y,x] = sum A[y, clamp(x+k)] / (2*r_eff+1)
k = -r_eff,...,+r_eff
G(A) = (Box_y o Box_x)^3 (A)
```

Indices are clamped at the image boundary, equivalent to replicating edge pixels. For calls that request two variance-equivalent passes, sigma is computed with p=2, but the filter implementation still quantizes to a three-pass effective radius. These details matter for reproducing the installed output.

Fill holes before estimating the smooth background

The keep mask K marks pixels that are still considered background. Rejected holes are initialized with the mean of kept pixels. Ten iterations smooth the current fill and then restore the known pixels. A final low-pass returns the model:

```
mu_K = sum(K*J) / sum(K)
F_0 = K*J + (1-K)*mu_K
F_(t+1) = K*J + (1-K)*G(F_t), t = 0,...,9
M = G(F_10)
```

If every pixel is kept, the model is simply G(J). If no pixel is kept, the fill routine returns zeros. This is a finite iterative smooth-fill construction, not an exact solution to a Poisson or harmonic boundary-value problem. Large masked regions may be weakly constrained by real background data.

What ONNX means here

The installed module creates a cached ONNX Runtime session with CPUExecutionProvider and passes the image and radius to a fixed filtering graph. This acceleration is distinct from the learned GraXpert AI model. The hardware acceleration toggle for learned AI operations does not change this Sample-Free session.

4 Robust fitting and iterative rejection

Each robust-fit call starts with all pixels kept, fits a first model F(K), then repeats the following loop up to AG_N_ITER = 20. F is either the polynomial fit on page 7 or masked smooth filling on page 4. Residual statistics are measured over the previous iteration's kept pixels; if that subset is empty, the whole residual is used.

```
R = J - model
m = median(R[K])
MAD = median(abs(R[K] - m))
sigma = 1.4826*MAD + 1e-12
```

Asymmetric rejection

The algorithm permits a larger negative residual than positive residual. Bright stars and nebulosity are therefore excluded earlier than moderately dark deviations. The scale estimate is robust to a minority of outliers, but still depends on whether genuine background remains in the frame.

```
low = m - 4*sigma; high = m + 2*sigma
K_new = (R >= low) AND (R <= high)
if protect: K_new = K_new AND NOT structure_mask(R-m)
```

The keep mask is recomputed from all residual pixels each iteration; it is not required to shrink monotonically. A previously rejected pixel can re-enter the fit. Structure protection adds a spatial exclusion criterion, described on the next page.

Minimum-support safeguard

```
n_min = max(16, int(0.02*N))
if count(K_new) < n_min:
k = min(n_min, N-1)
t = kth_smallest(R.flatten(), k)
K_new = (R <= t)
```

Here k is a zero-based partition index. Ties may keep more pixels than requested. The fallback uses a one-sided low-residual subset and replaces the preceding rejection/protection result. It prevents severe starvation but does not establish that the selected pixels are astrophysical background.

Refit and stop

```
model = F(K_new)
delta = abs(count(K_new)-previous_count) / N
K = K_new; previous_count = count(K_new)
stop if iteration > 0 AND delta < 0.0001
```

The convergence criterion compares counts, not the identity of retained pixels or a model-error norm. Equal-size masks can have different membership. A faithful port must preserve this criterion rather than silently replacing it with a different optimization objective.

5 Structure protection and final smoothness

Detect bright structure in centered residuals

Protection examines R-m, where m is the residual median from the robust loop. Let threshold t be protect_threshold and amount a be protect_amount. The detector is binary and one-sided:

```
D[p] = 1 if (R[p]-m) > t, else 0
r_grow = max(1, round(r*(0.5+a)))
S = G_sigma(r_grow,2)(D)
cutoff = (1-a)*0.5 + 0.001
protected[p] = S[p] > cutoff
```

If no residual exceeds t, the structure mask is entirely false. Otherwise the detector is low-passed and thresholded to form a coherent region. The amount control changes both the growth radius and cutoff. It is not a linear blend between an original and corrected image.

Interpret the controls

Lower threshold: more positive residuals qualify as structures. **Higher amount:** a larger growth radius and lower cutoff generally expand protection. The exact mask can change discretely because radius is rounded and the detector is binary. Dark structures are not directly selected by this positive detector; the asymmetric residual clipping remains relevant.

Protection is applied during the fit. It does not reconstruct or add back signal after subtraction. Broad nebulosity with intensity and scale similar to a gradient can still enter the model, particularly if it fills most of the field or falls below the absolute threshold.

Smoothness is a final model operation

After a multiscale robust fit, if smoothness s is positive, the fitted model receives another low-pass. This is skipped for a polynomial fit. A zero value skips this extra smoothing, but the multiscale fit itself remains smooth because its fill operation uses low-pass filtering.

```
if s > 0:
r_smooth = max(1, round(r*s))
M_final = G_sigma(r_smooth,3)(M)
else: M_final = M
```

Practical review

Compare the background against the original. Galaxy arms, dust lanes, nebular edges, or star halos visible in the background may indicate signal absorption. Increase scale, adjust protection, or compare simplified mode on and off. These are diagnostic experiments; no setting guarantees a unique physical separation of sky glow from diffuse astronomical emission.

6 Simplified mode: robust polynomial + residual

Polynomial design matrix

Coordinates on the reduced grid are mapped to [-1,1] in each direction. For degree d, include every monomial x^i y^j with i+j <= d. There are T=(d+1)(d+2)/2 coefficients. At default degree one, P is a plane; degree two adds curvature.

```
x_n = linspace(-1,1,w); y_n = linspace(-1,1,h)
P(x,y) = sum a_ij*x_n^i*y_n^j, i+j <= d
A[p,k] = x_n[p]^i_k * y_n[p]^j_k
T = (d+1)*(d+2)/2
```

Least-squares solve over kept pixels

The port builds normal-equation accumulators in row blocks, using only rows where K is true. It adds a tiny ridge term and uses Cholesky factorization and triangular solves. The implementation has fallback handling for insufficient support or a failed numerical solve; such cases should be treated as degraded fits rather than evidence for a high-degree background.

```
H = A_K^T*A_K + 1e-9*Identity(T)
b = A_K^T*J_K
H = L*L^T
a = solve(L^T, solve(L,b))
P = A*a
```

Degree six has 28 terms. A higher degree increases flexibility, conditioning sensitivity, and the possibility that real diffuse structure is absorbed into the background. The normal equations square the condition number relative to the original design matrix, which is one reason normalized coordinates matter.

Two independent robust stages

```
P = RobustFit(J, polynomial=True)
J_flat = J - P
M = RobustFit(J_flat, polynomial=False)
B_small = P + M
```

The two stages each run the rejection/protection logic. The multiscale stage begins a new keep mask on the flattened image. The smoothness control acts on M, not P. The optional polynomial preview shows P alone; it is not the full background model. With simplified mode off, B_small is a direct multiscale robust fit of J.

Removing a stiff global trend first changes the residual statistics and structure detector used by the second stage. This can help when a large gradient otherwise competes with nebulosity, but the polynomial does not mathematically guarantee preservation of extended objects.

7 Reconstruct the background and correct the image

Bilinear reconstruction

The installed resize routine uses endpoint-aligned coordinates: linspace(0,w-1,W) and linspace(0,h-1,H). For each output location, interpolate between the four neighboring reduced-grid samples. This differs from libraries that use half-pixel-center mappings.

```
u = x*(w-1)/(W-1); v = y*(h-1)/(H-1)
u0=floor(u); v0=floor(v); alpha=u-u0; beta=v-v0
B[y,x] = (1-alpha)*(1-beta)*Bsmall[v0,u0]
+ alpha*(1-beta)*Bsmall[v0,u1]
+ (1-alpha)*beta*Bsmall[v1,u0]
+ alpha*beta*Bsmall[v1,u1]
```

Neighbor indices u1 and v1 are clamped to the reduced extent. Each channel background is reconstructed to the original dimensions before correction. The optional polynomial component follows the same resize path.

Subtraction: additive gradients

```
level_c = median(B_c)
C_c = I_c - B_c + level_c
```

This removes spatial variation while retaining a channel-specific background pedestal. It does not force the background to zero and does not remove every source of background noise. A constant background model leaves the image unchanged apart from floating-point rounding.

Division: multiplicative gradients

```
C_c = I_c / max(B_c, 1e-6) * median(B_c)
```

The denominator floor prevents division by zero but does not make near-zero or negative fitted backgrounds physically valid. Division may strongly amplify noise where the denominator is small. Use subtraction for an additive sky glow model and division only when a multiplicative interpretation is appropriate.

Output semantics

The direct Sample-Free core returns a background and corrected array; it preserves non-finite source positions as NaN in the corrected image. The native wrapper uses that direct result rather than imposing the legacy AI path's [0,1] clipping. It preserves float32 values on 32-bit export. A 16-bit export necessarily clips/quantizes to [0,1]. Display stretch and saturation do not change the exported linear array.

8 Controls and preview behavior

Control	Default / range	Mathematical effect
Scale	5; 1-10	Radius as % of shorter reduced dimension
Smoothness	1; nonnegative	Extra low-pass radius round(r*s); app range 0-5
Protect	On	Add structure exclusion to each robust iteration
Threshold	0.05; 0-1	Absolute threshold on residual minus its median
Amount	0.5; 0-1	Growth radius and structure-mask cutoff
Simplified	On	Polynomial followed by multiscale residual fit
Degree	1; 1-6	Total polynomial degree; used in simplified mode
Downsample	4; 1,2,4,8	Area reduction before fitting
Correction	Subtraction	Subtract with pedestal or divide with floor

Native application interaction

Controls are on the left and the image canvas is on the right. Release a slider to submit a calculation; double-click it to restore its default. The app serializes calculations, queues the newest settings, and discards results from obsolete requests. Switching between original, final, background, and polynomial uses cached display images. Fit and zoom controls are separate from processing.

The initial window is inset within NSScreen.visibleFrame, which excludes the visible Dock and menu bar. File selection accepts GraXpert-supported astronomical formats. Save computes a fresh full-resolution result from the selected settings, independent of preview resolution.

Preview is an approximation

For responsiveness, preview input is limited to 1024, 2048, or 4096 pixels on its longest side, followed by the selected algorithm downsample factor. Changing the preview size can alter clipping, structure masks, and fitted results. Full-resolution export can therefore differ. Check a 4096-pixel preview or the saved full-resolution result for subtle gradients and small objects.

Select active image from PixInsight

After an observed PixInsight crash during interface investigation, the final app avoids externally launching the PixInsight command-line executable. The button creates a transfer request and asks the user to run the bundled ExportActiveImage.js inside PixInsight. That helper exports a temporary 32-bit XISF copy, and the app loads it when the reply arrives. The transfer path is prepared but has not been verified in a live PixInsight session. No PI_Scripts repository integration is included.

9 Verification, reproducibility, and sources

Checks completed

Synthetic tests exercised odd-sized monochrome inputs, constant RGB input, source immutability, finite backgrounds, and both correction equations. Preview runs passed for Sample-Free, AI background extraction, RBF, splines, and kriging. Export checks reopened 16-bit and 32-bit TIFF, FITS, and XISF files and verified dimensions and sample types. Denoising and deconvolution are wired to installed GraXpert functions, but matching local models were unavailable for numerical validation.

The macOS app is built from SwiftUI/AppKit source and uses the installed GraXpert bundle through a small Python adapter. It does not bundle or replace the original GraXpert engine. It expects the inspected Python 3.14 Sample-Free build; another GraXpert package may require an updated adapter. The desktop automation service failed, so visual interaction and the safe PixInsight transfer require user verification.

Algorithm provenance

[1] Installed implementation. /Applications/GraXpert.app; modules graxpert.sample_free_background, graxpert.sample_free_onnx, background_extraction, astroimage, denoising, and deconvolution. Read-only archive inspection and direct execution, 16 October 2026. Sample-Free constants: HIGH_K=2, LOW_K=4, N_ITER=20, PASSES=3. The installed module identifies its adaptation as GPL-3.0-or-later.

[2] Siril source cited by the installed module. background_extraction.c. The local source retrieval on 16 October 2026 was used for comparison. Current Siril uses its own Gaussian implementation, so exact numerical equivalence to the installed box-filter ONNX port is not asserted.

[3] Simplified-model revision cited by the installed module. Siril commit 5555df299f96923551bcc093c0e5e1a022648061, identified in the module docstring as the revised model of 3 September 2026. The citation comes from the installed module; the commit page itself was not independently retrievable through the web reader.

[4] Official GraXpert comparison. Upstream source snapshot, including background_extraction.py and README.md. This reference explains the learned AI, RBF, spline, and kriging methods, not the newer installed Sample-Free port.

Difference from learned AI background extraction

Upstream AI resizes the image to a padded 256 x 256 model input, normalizes channels by their median and MAD, invokes a learned ONNX background predictor, reverses normalization, smooths, crops padding, and resizes the predicted background. Sample-Free instead fits a deterministic background using pixel rejection and low-pass filling. Both need no manual sample points, but their mathematics, controls, models, and failure modes differ.

Future PI_Scripts work should pin an engine/source version, preserve the full correction and normalization contract, and test output parity on representative linear monochrome and RGB astronomical images. That integration is deliberately outside this deliverable.