Proposal: Luminosity-Inverting Dark Mode for the TeXShop PDF Preview

April 2026

1 Summary

This document proposes adding an optional luminosity-inverting dark mode to TeXShop's built-in PDF preview. When macOS dark mode is active and the user enables this feature, the PDF preview would display with inverted luminosity — white pages become dark, black text becomes light — while preserving the hue of coloured content such as figures and diagrams.

The proposed implementation follows the approach used by Skim, applying `CIFilter` objects via `NSView.contentFilters` to the PDF scroll view. This is non-invasive: it does not modify the PDF data, does not interfere with text selection, annotations, or search highlighting, and requires changes to only a small number of existing TeXShop source files.

2 Motivation

TeXShop 5.x supports macOS dark mode for its editor chrome (menus, toolbars, source editor background) but renders PDF content exactly as authored — white pages on a dark UI. This creates a sharp brightness contrast that is visually uncomfortable, particularly during extended editing sessions in low-light environments.

Users currently work around this by using an external previewer such as Skim, which supports luminosity inversion in dark mode. A native solution within TeXShop would preserve the integrated edit–preview workflow that is one of TeXShop's key strengths.

3 Technical approach

3.1 Why `contentFilters`?

Apple's `PDFKit` provides no built-in API for dark mode rendering of PDF page content. The `PDFView.backgroundColor` property only controls the area around pages, not the page content itself. Four approaches were evaluated:

(a) `NSView.contentFilters` — an array of `CIFilter` objects applied to a view's rendered output. Non-invasive, works with all PDFKit features. *Recommended.*

(b) `CALayer.filters` — modifying the backing layer directly. Officially unsupported since macOS 10.13; undefined behaviour.

(c) `PDFPage` **subclass with custom drawing** — override `draw(with:to:)` to manipulate the `CGContext`. Fragile across macOS versions; interferes with annotation rendering.

(d) **Render to bitmap, apply filter, display as image** — loses all interactivity (selection, links, search). Not practical.

Approach (a) is the same one used by Skim, where it has been stable across macOS 10.14 through macOS 15. It uses only built-in `CIFilter` types (`CIGammaAdjust`, `CIColorMatrix`), which continue to work reliably even though custom `CIFilter` kernels on `CALayer` have been broken since macOS 11.

3.2 Luminosity inversion vs. naive RGB inversion

A naive RGB inversion (`CIColorInvert`) maps each channel c to $1 - c$. This turns red to cyan, blue to yellow, and so on — unacceptable for coloured figures. Luminosity inversion instead computes the perceived luminance

$$L = 0.2126\,R + 0.7152\,G + 0.0722\,B$$

using the ITU-R BT.709 coefficients and inverts only the luminance component, preserving hue. The transformation is:

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} 1 - fL_R & -fL_G & -fL_B \\ -fL_R & 1 - fL_G & -fL_B \\ -fL_R & -fL_G & 1 - fL_B \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

where $L_R = 0.2126$, $L_G = 0.7152$, $L_B = 0.0722$, and $f \approx 1.90$ is a tuning factor. With $f = 1.8972$, pure white maps to approximately `#1E1E1E` (the standard macOS dark-mode background), not pure black. This avoids excessive contrast. When the system accessibility setting "Increase contrast" is active, Skim uses $f = 1.9337$.

The inversion is performed in linear colour space by bracketing the colour matrix with gamma adjustments:

1. `CIGammaAdjust` with power 0.625 (linearise),
2. `CIColorMatrix` (luminosity inversion),
3. `CIGammaAdjust` with power 1.6 (re-apply gamma).

3.3 Appearance forcing

When the inversion filter is active, the scroll view's `NSAppearance` must be forced to `NSAppearanceNameAqua` (light mode). This ensures the PDF renders with a white background *before* the filter inverts it. Without this, the system would render a dark background which, after inversion, would become white — the opposite of the desired effect.

3.4 Limitation

The filter is applied to the entire view. Coloured images and figures are inverted along with the text. While the luminosity-preserving approach keeps hues recognisable, images will appear with altered brightness. This is the same limitation present in Skim. The commercial application Texifier addresses this by offering three modes (full inversion, images excluded, or none), but the "images excluded" mode requires a more complex rendering pipeline that identifies and masks image regions.

For a first implementation, the all-or-nothing approach is appropriate and matches user expectations from Skim.

4 Proposed changes to TeXShop

The implementation touches 8 files with minimal modifications to each. It follows existing TeXShop patterns: the toggle menu item pattern from `toggleLinkPopups:`, the preference pattern from `PdfColorMapKey`, and the dark mode detection pattern from `TSTextView`'s `viewDidChangeEffectiveAppearance`.

4.1 Overview of files

File	Class / Area	Change
globals.h	Global constants	Add preference key declaration
globals.m	Global constants	Add preference key definition
FactoryDefaults.plist	Default preferences	Add default value
MyPDFKitView.h	MyPDFKitView	Add 3 method declarations
MyPDFKitView.m	MyPDFKitView	Add filter methods + hook into setup
TSTextView.m	TSTextView	Trigger filter update on appearance change
TSPreviewWindow.h	TSPreviewWindow	Add toggle method declaration
TSPreviewWindow.m	TSPreviewWindow	Add toggle + menu validation

4.2 Step 1: Preference key

Sources/globals.h Add after BringFrontOnTypesetKey:

```
extern NSString *DarkModeInvertPreviewKey;
```

Sources/globals.m Add near DefaultDarkThemeKey:

```
NSString *DarkModeInvertPreviewKey = @"DarkModeInvertPreview";
```

Resources/FactoryDefaults.plist Add within the root <dict>:

```
<key>DarkModeInvertPreview</key>
<string>YES</string>
```

4.3 Step 2: Core filter methods in MyPDFKitView

Sources/MyPDFKitView.h Add after fixWhiteDisplay:

```
- (void)applyDarkModeInversionFilter;
- (void)removeDarkModeInversionFilter;
- (void)updateDarkModeInversion;
```

Sources/MyPDFKitView.m Add before @end:

```
#pragma mark - Dark Mode Luminosity Inversion

#define LR 0.2126
#define LG 0.7152
#define LB 0.0722

- (void)applyDarkModeInversionFilter
{
#ifdef MOJAVEORHIGHER
    NSScrollView *scrollView =
        [[self documentView] enclosingScrollView];
    if (!scrollView)
        return;
```

```
    // Force light appearance so the PDF renders with
    // a white background before we invert
    [scrollView setAppearance:
        [NSAppearance appearanceNamed:NSAppearanceNameAqua]];

    NSMutableArray *filters = [NSMutableArray array];
    CIFilter *filter;

    // Step 1: Linearise (gamma pre-correction)
    filter = [CIFilter filterWithName:@"CIGammaAdjust"
        keysAndValues:@"inputPower",
        [NSNumber numberWithDouble:0.625], nil];
    if (filter) [filters addObject:filter];

    // Step 2: Luminosity inversion via colour matrix
    // Tuning factor: white -> ~#1E1E1E
    CGFloat f = [[NSWorkspace sharedWorkspace]
        accessibilityDisplayShouldIncreaseContrast]
        ? 1.9337 : 1.8972;

    filter = [CIFilter filterWithName:@"CIColorMatrix"
        keysAndValues:
        @"inputRVector",
            [CIVector vectorWithX:1.0-LR*f
                                Y:-LG*f
                                Z:-LB*f W:0.0],
        @"inputGVector",
            [CIVector vectorWithX:-LR*f
                                Y:1.0-LG*f
                                Z:-LB*f W:0.0],
        @"inputBVector",
            [CIVector vectorWithX:-LR*f
                                Y:-LG*f
                                Z:1.0-LB*f W:0.0],
        @"inputAVector",
            [CIVector vectorWithX:0.0 Y:0.0
                                Z:0.0 W:1.0],
        @"inputBiasVector",
            [CIVector vectorWithX:1.0 Y:1.0
                                Z:1.0 W:0.0],
        nil];
    if (filter) [filters addObject:filter];

    // Step 3: Re-apply gamma
    filter = [CIFilter filterWithName:@"CIGammaAdjust"
        keysAndValues:@"inputPower",
        [NSNumber numberWithDouble:1.6], nil];
    if (filter) [filters addObject:filter];

    [scrollView setContentFilters:filters];
#endif
}

- (void)removeDarkModeInversionFilter
{
#ifdef MOJAVEORHIGHER
    NSScrollView *scrollView =
```

```
        [[self documentView] enclosingScrollView];
    if (!scrollView)
        return;

    [scrollView setAppearance:nil];
    [scrollView setContentFilters:@[]];
#endif
}

- (void)updateDarkModeInversion
{
#ifdef MOJAVEORHIGHER
    if (@available(macOS 10.14, *)) {
        BOOL isDark = [[self effectiveAppearance]
            bestMatchFromAppearancesWithNames:
            @[NSAppearanceNameDarkAqua]] != nil;
        BOOL userWants =
            [SUD boolForKey:DarkModeInvertPreviewKey];

        if (isDark && userWants)
            [self applyDarkModeInversionFilter];
        else
            [self removeDarkModeInversionFilter];
    }
#endif
}
```

Hook into existing setup methods. In the `setup` method (after `initializeDisplay`) and in `showForSecond` (after `initializeDisplay`), add:

```
[self updateDarkModeInversion];
```

4.4 Step 3: Respond to system appearance changes

`Sources/TSTextView.m` In `viewDidChangeEffectiveAppearance`, after each `[self.document changeColors:]` call, add:

```
[self.document.myPDFKitView updateDarkModeInversion];
[self.document.myPDFKitView2 updateDarkModeInversion];
```

This ensures the filter is applied or removed when the user switches between light and dark mode at the system level.

4.5 Step 4: Menu toggle

`Sources/TSPreviewWindow.h` Add after `toggleLinkPopups:`:

```
- (void)toggleDarkModeInversion:(id)sender;
```

`Sources/TSPreviewWindow.m` Add the toggle action (following the `toggleLinkPopups:` pattern):

```
- (void)toggleDarkModeInversion:(id)sender
```

```
{
    BOOL current = [SUD boolForKey:DarkModeInvertPreviewKey];
    [SUD setBool:!current forKey:DarkModeInvertPreviewKey];
    [SUD synchronize];

    MyPDFKitView *view1 = self.myPDFKitView;
    MyPDFKitView *view2 = self.myPDFKitView2;
    [view1 updateDarkModeInversion];
    [view2 updateDarkModeInversion];
}
```

In `validateMenuItem:`, add after the `toggleLinkPopups:` block:

```
if ([anItem action] ==
    @selector(toggleDarkModeInversion:)) {
    if ([SUD boolForKey:DarkModeInvertPreviewKey])
        [anItem setState:NSOnState];
    else
        [anItem setState:NSOffState];
    return YES;
}
```

Menu item. The simplest approach is to add the item programmatically in `TSAppDelegate.m` within `applicationDidFinishLaunching:`, after the existing Copy Format menu setup:

```
NSMenuItem *darkInvertItem = [[NSMenuItem alloc]
    initWithTitle:NSLocalizedString(
        @"Invert Preview Colors",
        @"Invert Preview Colors")
    action:@selector(toggleDarkModeInversion:)
    keyEquivalent:@""];
[darkInvertItem setTarget:nil];
// Insert into Preview menu at appropriate index
```

This avoids editing the 12+ localised MainMenu nib files.

5 Skim source reference

Skim's implementation, which this proposal follows, is open source and available at https://github.com/scris/skim-pdf. The relevant files are:

- `NSGraphics_SKExtensions.m` — the `SKColorEffectFilters()` function containing the full filter chain.
- `SKBasePDFView.m` — where `contentFilters` is applied to the scroll view, appearance is forced, and `viewDidChangeEffectiveAppearance` is handled.

6 Testing considerations

1. Verify inversion activates only when both macOS dark mode is on *and* the preference is enabled.
2. Verify toggling the menu item applies/removes the filter immediately.

3. Verify switching between light and dark mode at the system level updates the preview correctly.
4. Verify text selection, search highlighting, and link clicking work normally with the filter active.
5. Verify split-view mode (both `myPDFKitView` and `myPDFKitView2`) inverts correctly.
6. Verify coloured figures are recognisable (hues preserved, luminosity inverted).
7. Test across macOS versions: 10.14 (Mojave) through the current release.
8. Verify the "Increase contrast" accessibility setting produces a slightly stronger inversion.

7 Future enhancements

- **Sepia tone and white point adjustment.** Skim also supports sepia tone (`CISepiaTone`) and white point adjustment (`CIWhitePointAdjust`) as additional filters. These could be exposed as preferences for users who prefer a warm-tinted preview.
- **Image exclusion.** A more advanced implementation could identify image regions in the PDF and exclude them from inversion, similar to Texifier's "images excluded" mode. This would require parsing the PDF content stream to locate image XObjects and masking those regions from the filter — significantly more complex but achievable.
- **Keyboard shortcut.** Assign a default key equivalent (e.g., `Cmd+Shift+I`) for quick toggling.