```
using Images,TestImages
```

```
begin
using Base: OneTo, tail
using OffsetArrays


struct PaddedView{T,N,I,A} <: AbstractArray{T,N}
    fillvalue::T
    data::A
    indices::I
    function PaddedView{T,N,I,A}
(fillvalue,data,indices::NTuple{N,AbstractUnitRange}) where {T,N,I,A}
        new{T,N,I,A}(fillvalue, data, indices)
    end
end

function PaddedView(fillvalue::FT, data::AbstractArray{T}, args...) where {FT, T}
    PaddedView{filltype(FT, T)}(fillvalue, data, args...)
end

function PaddedView{FT}(fillvalue,data::AbstractArray{T,N},
            sz::Tuple{Integer,Vararg{Integer}},dims=1) where {FT,T,N}
    if(dims==0)
      val=copy(sz[1])
     new_tuple=(size(data)[2],val)
    elseif(dims==1)
      val=copy(sz[2])
     new_tuple=(val,size(data)[1])
    end
    inds = map(OneTo, new_tuple)
    dims=1
    PaddedView{FT,N,typeof(inds),typeof(data)}(convert(FT, fillvalue), data, inds)
end

filltype(::Type, ::Type{T}) where T = T
filltype(::Type{FT}, ::Type{T}) where {FT<:Union{Nothing, Missing}, T} = Union{FT,
T}
filltype(::Type{FT}, ::Type{T}) where {FT, T<:Union{Nothing, Missing}} = Union{FT,
T}
# ambiguity patch
filltype(::Type{FT}, ::Type{T}) where {FT<:Union{Nothing, Missing},
T<:Union{Nothing, Missing}} = Union{FT, T}

Base.axes(A::PaddedView) = A.indices
@inline Base.axes(A::PaddedView, d::Integer) = d <= ndims(A) ? A.indices[d] :
default_axes(A.indices)
default_axes(::NTuple{N,I}) where {N,I<:AbstractUnitRange} = convert(I, OneTo(1))
default_axes(::Tuple{}) = OneTo(1)
default_axes(::Any) = OneTo(1)

Base.size(A::PaddedView) = map(length, axes(A))

Base.parent(A::PaddedView) = A.data

Base.@propagate_inbounds function Base.getindex(A::PaddedView{T,N},
i::Vararg{Int,N}) where {T,N}
    @boundscheck checkbounds(A, i...)
    if Base.checkbounds(Bool, A.data, i...)
        return convert(T, A.data[i...])
    end
    return A.fillvalue
end
```

```
function Base.showarg(io::IO, A::PaddedView, toplevel)
    print(io, "PaddedView(", A.fillvalue, ", ")
    Base.showarg(io, parent(A), false)
    print(io, ", (", join(A.indices, ", "))
    print(io, ndims(A) == 1 ? ",))" : "))")
    toplevel && print(io, " with eltype ", eltype(A))
end

end
```

```
PaddedView(-1, pyramid[2], (512,512),0) # 0 if we want update in rows direction, and 1 if we want to update in columns direction
```

(a vector displayed as a row to save space)

```
begin
img_source = testimage("woman_darkhair")
pyramid = gaussian_pyramid(img_source, 5, 2, 1)
end
```

```
Enter cell code...
```