1 System: Statistical Image

```python
import numpy as np
from numba.experimental import jitclass
from numba import int64
integer = int64
```

```python
__all__ = ['StatisticalImage']
@jitclass([
    ('I0', integer[:]),
    ('I',  integer[:]),
    ('N',  integer),
    ('M',  integer),
    ('E',  integer),
    ('Ev', integer),
    ('dE', integer),
    ('dx', integer),
    ('i',  integer)
])
class StatisticalImage:
    def __init__(self, I0, I, M):
        if len(I0) != len(I):
            raise ValueError('Ground image I0 and current image I should have the same length.')
        if M < 0:
            raise ValueError('Maximum site value must be nonnegative.')
        self.I0 = I0
        self.I  = I
        self.N  = len(I0)
        self.M  = M
        self.E  = self.energy()
        self.Ev = self.E
        self.dE = 0
        self.dx = 0
        self.i  = 0
    def state(self):
        return self.I0.copy(), self.I.copy(), self.M
    def state_names(self):
        return 'I0', 'I', 'M'
    def copy(self):
        return StatisticalImage(*self.state())
    def energy_bins(self):
        E0 = 0
        Ef = np.sum(np.maximum(self.I0, self.M - self.I0))
        ΔE = 1
        return np.arange(E0, Ef + ΔE + 1, ΔE)
    def energy(self):
        return np.sum(np.abs(self.I - self.I0))
    def propose(self):
        i = np.random.randint(self.N)
        self.i = i
        x0 = self.I0[i]
        x = self.I[i]
        r = np.random.randint(2)
        if x == 0:
            dx = r
        elif x == self.M:
            dx = -r
        else:
            dx = 2*r - 1
        dE = np.abs(dx) if x0 == x else (dx if x0 < x else -dx)
        self.dx = dx
        self.dE = dE
        self.Ev = self.E + dE
    def accept(self):
        self.I[self.i] += self.dx
        self.E = self.Ev
```