JOURNAL OF
APPLIED
CRYSTALLOGRAPHY

Volume 55 (2022)

Supporting information for article:

Combination of an inject-and-transfer system for serial femtosecond crystallography

Keondo Lee, Jihan Kim, Sangwon Baek, Jaehyun Park, Sehan Park, Jong-Lam Lee, Wan Kyun Chung, Yunje Cho and Ki Hyun Nam

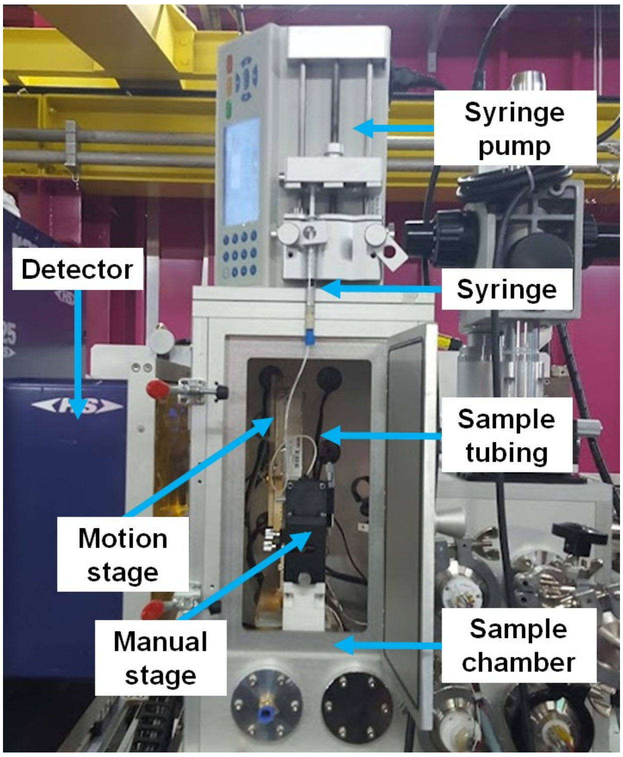

Figure S1 Photo of experimental setup of the BITS at PAL-XFEL..

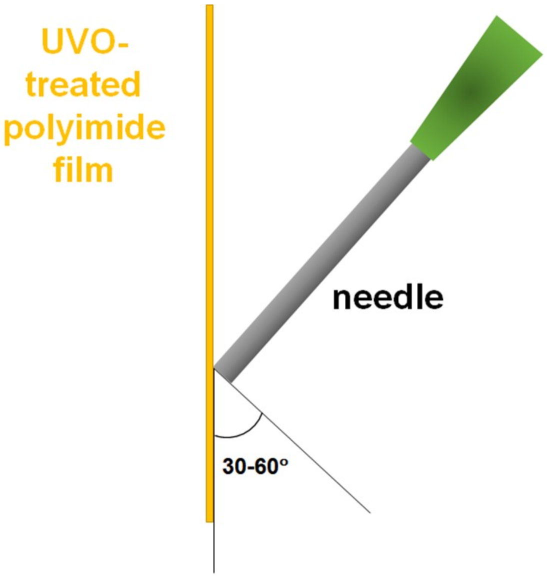

Figure S2 Schematic drawing of the injection needle attached to the UVO-treated polyimide film.

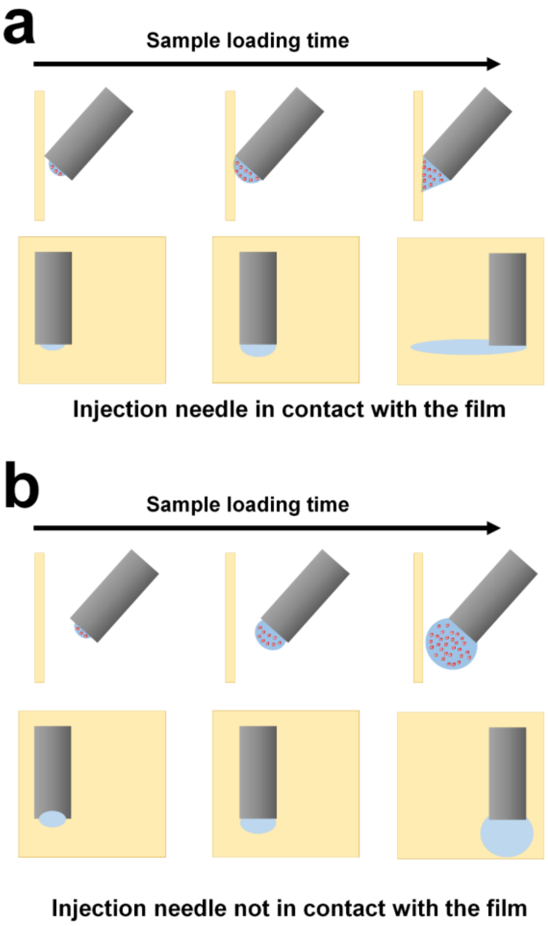
a
Sample loading time
Injection needle in contact with the film
b
Sample loading time
Injection needle not in contact with the film

Figure S3 Schematic drawing of the difference in sample deposition according to injection needle (a) contact and (b) non-contact with UVO-treated polyimide film.

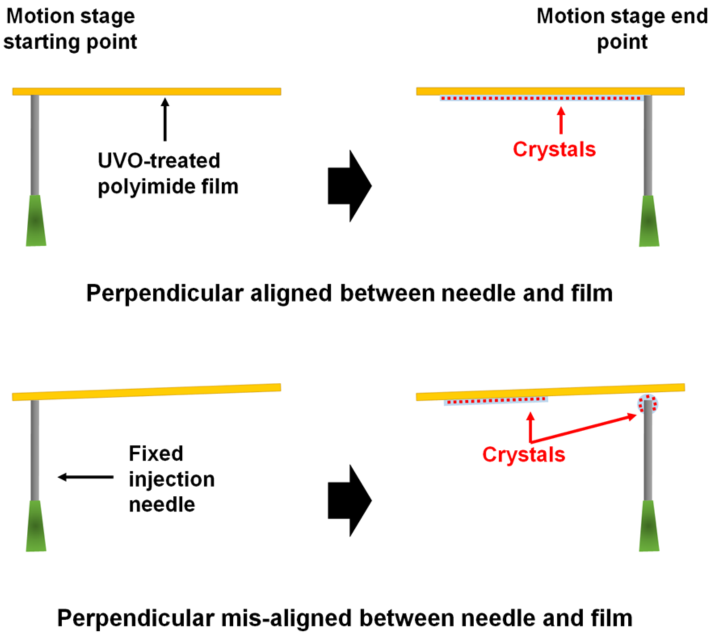
Motion stage starting point
Motion stage end point
UVO-treated polyimide film
Crystals
Perpendicular aligned between needle and film
Fixed injection needle
Crystals
Perpendicular mis-aligned between needle and film

Figure S4 Schematic drawing of differences in sample deposition according to perpendicular (a) aligned and (b) misaligned UVO-treated polyimide film with the injection needle.

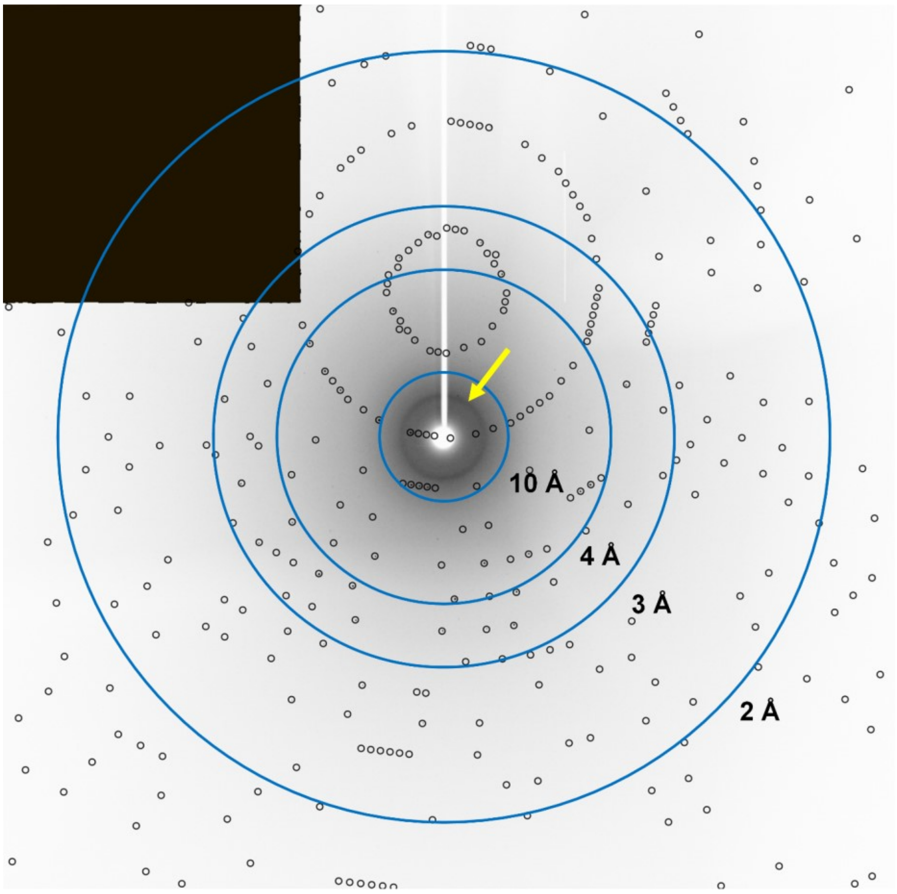

Figure S5 Typical diffraction image of lysozyme embedded in lard material delivered by BITS. The background scattering of lard material is indicated by the yellow arrow. The circles indicate the Bragg peak poinsions predicted by CrytFEL.

Table S1. The translation code used in BITS system.

```
cv::Point2d start_pt;
int step_size_x = 50; // um
int step_size_y = 5000; // um
int sub_step_size_x = 50; // um
int sub_step_size_y = 150; // um
int chip_size_x = 30000; // X-axis scan range [um]
int chip_size_y = step_size_y*5;  // Y-axis scan range [um]
int sub_step_x = (int)chip_size_x / sub_step_size_x;
double waitingTime; // Timer
LARGE_INTEGER clockStart, clockEnd, clockFreq; // Timer


// Get Start Location
start_pt.x = static_cast<double>(smaract_mcs2->get_position(smaract_ch_x));
start_pt.y = static_cast<double>(smaract_mcs2->get_position(smaract_ch_y));


//================= BITS Scan code =================//
for (int i = 0; i < step_y && SCANNING_ON; i++) {
   // Start time measurement
   QueryPerformanceFrequency(&clockFreq); // Timer
   QueryPerformanceCounter(&clockStart); // Timer


   // Set the travel speed of Smaract stage
   smaract_mcs2->set_velocity(smaract_ch_x, step_size_x * 30);
   smaract_mcs2->set_velocity(smaract_ch_y, step_size_x * 30);


   //------------- Scan code -------------
   //   ┌─┐ ┌─┐ ┌─┐ ┌─┐ ┌─┐ ┌─┐ ┌─┐ ┌─┐
   //   | | | | | | | | | | | | | | | |
   //   │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │
   //────┘ └──┘ └──┘ └──┘ └──┘ └──┘ └──┘ └──┘ └──→
   for (int k = 0; k < sub_step_x && SCANNING_ON; k++) {
       smaract_mcs2->absolute_move(smaract_ch_x, start_pt.x + sub_step_size_x * k);
       smaract_mcs2->wait_for_moving(smaract_ch_x);
       smaract_mcs2->absolute_move(smaract_ch_y, start_pt.y + step_size_y * i + sub_step_size_y * (k %
2));
       smaract_mcs2->wait_for_moving(smaract_ch_y);
   }

   // If SCANNING_ON is false, stop the thread
   if (!SCANNING_ON) {
       break;
   }

   // Set the travel speed of Smaract stage
   smaract_mcs2->set_velocity(smaract_ch_x, chip_size_x*0.6);
   smaract_mcs2->set_velocity(smaract_ch_y, step_size_y*0.6);

   // Move to the next line
   //───→
   // ↖
   //   ↖
   //     ↖
   //       ↖
   //         ↖
   //           ↖
   //             ↖
   //             ↖

   //              ↖

   //               ↖

   //                ↖

   //                 ↖

   //                  ↖

   //                   ↖

   //                    ↖

   //  ┌─┐ ┌─┐ ┌─┐ ┌─┐ ┌─┐ ┌─┐ ┌─┐↖┌─┐

   //  │ │ │ │ │ │ │ │ │ │ │ │ │ │↖│

   //  │ │ │ │ │ │ │ │ │ │ │ │ │ │ │ │↖

   //──┘ └─┘ └─┘ └─┘ └─┘ └─┘ └─┘ └─┘ └─→

   smaract_mcs2->absolute_move(smaract_ch_x, start_pt.x);

   smaract_mcs2->absolute_move(smaract_ch_y, start_pt.y + step_size_y * (i + 1));

   smaract_mcs2->wait_for_moving(smaract_ch_x);

   smaract_mcs2->wait_for_moving(smaract_ch_y);


   // Stop time measurement

   QueryPerformanceCounter(&clockEnd); // Timer

   waitingTime = ((double)(clockEnd.QuadPart - clockStart.QuadPart)) / clockFreq.QuadPart; // Timer [s]


   // Monitoring the scan progress

   std::stringstream ss;

   ss << "[" << i + 1 << "/" << step_y << "] " << (int)round((waitingTime * (step_y - i - 1)) / 60) << "
minutes remain...";

   scanning_status = ss.str();

   std::cout << scanning_status << std::endl;
   zmq_send(publisher, scanning_status.data(), scanning_status.size(), ZMQ_NOBLOCK);

}
```